macromedia®

# FLASH® 8

Aprendizaje de ActionScript 2.0 en Flash

**Agradecimientos**

Dirección del proyecto: Sheila McGinn

Redacción: Jen deHaan; Peter deHaan, Joey Lott

Directora de edición: Rosana Francescato

Redactora jefe: Lisa Stanziano

Edición: Linda Adler, Geta Carlson, Evelyn Eldridge, John Hammett, Mary Kraemer, Noreen Maher, Jessie Wood, Anne Szabla

Dirección de la producción: Patrice O'Neill, Kristin Conradi, Yuko Yagi

Producción y diseño multimedia: Adam Barnett, Aaron Begley, Paul Benkman. John Francis, Geeta Karmarkar, Masayo Noda, Paul Rangel, Arena Reed, Mario Reynoso

Reconocimiento especial a Jody Bleyle, Mary Burger, Lisa Friendly, Stephanie Gowin, Bonnie Loo, Mary Ann Walsh, Erick Vera, Jorge G. Villanueva, responsables de las pruebas realizadas a la versión beta, y a la totalidad de los equipos de diseño y control de calidad de Flash y Flash Player.

Primera edición: septiembre de 2005

Macromedia, Inc.
601 Townsend St.
San Francisco, CA 94103, EE.UU.

# Contenido

# Introducción

Macromedia Flash Basic 8 y Macromedia Flash Professional 8 son las herramientas estándar de edición profesional para la creación de publicaciones Web de gran impacto. ActionScript es el lenguaje que deberá utilizar para añadir interactividad a aplicaciones Flash, tanto si las aplicaciones son simples archivos SWF de animación como si son complejas aplicaciones de Internet. Para utilizar Flash, no es necesario utilizar ActionScript, pero si desea ofrecer a los usuarios interacción básica o compleja, trabajar con objetos que no sean los incorporados en Flash (como por ejemplo, botones y clips de película) o convertir un archivo SWF en una experiencia de usuario más fiable, es posible que desee utilizar este lenguaje.

Para más información, consulte los temas siguientes:



## Destinatarios

En este manual se presupone que el usuario ya ha instalado Flash Basic 8 o Flash Professional 8 y sabe cómo utilizarlo. Deberá saber cómo colocar objetos en el escenario y cómo manipularlos en el entorno de edición de Flash. Si ha utilizado lenguajes de creación de scripts anteriormente, ActionScript le resultará familiar. No obstante, aunque no tenga experiencia en programación, le resultará fácil aprender a utilizar ActionScript. Se puede empezar con comandos muy simples y aumentar el grado de complejidad conforme se va avanzando. Puede añadir gran cantidad de interactividad a sus archivos sin tener que aprender (ni escribir) gran cantidad de código.

# Requisitos del sistema

Los componentes de ActionScript 2.0 no exigen ningún requisito del sistema adicional a los de Flash 8.

En este manual se presupone que el usuario utiliza la configuración de publicación predeterminada para los archivos Flash: Flash Player 8 y ActionScript 2.0. Si cambia alguno de estos valores, es posible que las explicaciones y los ejemplos de código que se muestran en la documentación no sean válidos. Si desarrolla aplicaciones para versiones anteriores de Flash Player, consulte el Apéndice D, “Escritura de scripts para versiones anteriores de Flash Player”, en la página 833.

# Actualización de archivos XML de Flash

Es importante tener siempre instalados los archivos XML más recientes de Flash. Macromedia incorpora en ocasiones nuevas funciones en subediciones (versiones secundarias) de Flash Player. Cuando se encuentre disponible una versión de este tipo, deberá actualizar su versión de Flash para obtener los archivos XML más recientes. En caso contrario, el compilador de Flash 8 podría generar errores si utiliza nuevas propiedades o métodos que no estaban disponibles en la versión de Flash Player suministrada con la instalación de Flash.

Por ejemplo, Flash Player 7 (7.0.19.0) contenía un método nuevo para el objeto System, `System.security.loadPolicyFile`. Para acceder a este método, debe utilizar el programa de instalación de Player Updater para actualizar todos los Flash Players instalados con Flash. De no hacerlo, el compilador de Flash mostrará errores.

Recuerde que puede instalar un Player Updater que sea una o varias versiones principales superior a la versión de Flash de que dispone. Al hacerlo, obtendrá los archivos XML necesarios sin que se produzcan errores de compilador al publicar en versiones anteriores de Flash Player. En ocasiones, los nuevos métodos o propiedades están disponibles para versiones anteriores, por lo que, si dispone de los archivos XML más recientes, se minimizan los errores de compilador que aparecen al intentar acceder a métodos o propiedades anteriores.

# Documentación

Este manual proporciona información general sobre la sintaxis de ActionScript y cómo utilizar ActionScript cuando se trabaja con distintos tipos de objetos. Para obtener detalles sobre la sintaxis y el uso de cada uno de los elementos de lenguaje, consulte *Referencia del lenguaje ActionScript 2.0*.

Para más información, consulte los temas siguientes:

- "Información general sobre el libro Aprendizaje de ActionScript 2.0" en la página 11
- "Archivos de muestra" en la página 15
- "Términos utilizados en este documento" en la página 14
- "Copiar y pegar código" en la página 14

## Información general sobre el libro Aprendizaje de ActionScript 2.0

En la siguiente lista se resume el contenido de este manual:

- El Capítulo 1, "Novedades de Flash 8 ActionScript", describe las funciones nuevas de ActionScript, los cambios aplicados al compilador y al depurador, así como el nuevo modelo de programación para el lenguaje ActionScript 2.0.
- El Capítulo 2, "Escritura y edición en ActionScript 2.0", describe las funciones del editor de ActionScript incluido en Flash que facilita la escritura de código.
- El Capítulo 3, "ActionScript", explica en qué consiste el lenguaje ActionScript y proporciona detalles que permiten elegir la versión de ActionScript que debe utilizar.
- El Capítulo 4, "Principios básicos de la sintaxis y el lenguaje", describe la terminología y los conceptos básicos del lenguaje ActionScript. Estos conceptos se utilizan en todo el manual.
- El Capítulo 5, "Funciones y métodos", explica cómo escribir distintos tipos de funciones y métodos, así como la forma de utilizarlos en la aplicación.
- El Capítulo 6, "Clases", describe la forma de crear objetos y clases personalizadas en ActionScript. Este capítulo enumera también las clases integradas en ActionScript y ofrece una breve descripción de cómo utilizarlas para obtener acceso a potentes funciones de ActionScript.
- El Capítulo 7, "Herencia", describe la herencia en el lenguaje ActionScript, así como la forma de ampliar clases incorporadas o personalizadas.
- El Capítulo 8, "Interfaces", describe la forma de crear objetos y trabajar con interfaces en ActionScript.

- El Capítulo 9, “Gestión de eventos”, explica varias maneras distintas de controlar los eventos: métodos de controlador de eventos, detectores de eventos y controladores de eventos de clip de película y de botón.
- El Capítulo 10, “Datos y tipos de datos”, describe la terminología y los conceptos básicos de datos, tipos de datos y variables. Estos conceptos se utilizan en todo el manual.
- El Capítulo 11, “Trabajo con clips de película”, describe clips de película y el código ActionScript que puede utilizar para controlarlos.
- El Capítulo 12, “Utilización de texto y cadenas”, examina las distintas maneras en las que se puede controlar texto y cadenas en Flash e incluye información sobre el formato de texto y FlashType (representación de texto avanzada, como el texto suavizado).
- El Capítulo 13, “Animaciones, filtros y dibujos”, explica cómo crear imágenes y animación basadas en código, añadir filtros a objetos y dibujar con ActionScript.
- El Capítulo 14, “Creación de interacciones con ActionScript”, describe algunos métodos sencillos para crear aplicaciones más interactivas, incluido el control de reproducción de archivos SWF, la creación de punteros personalizados y la creación de controles de sonido.
- El Capítulo 15, “Utilización de imágenes, sonido y vídeo”, describe la forma de importar a las aplicaciones Flash archivos multimedia externos, como imágenes de mapa de bits, archivos MP3, archivos de Flash Video (FLV) y otros archivos SWF. En este capítulo se incluye también información general sobre cómo trabajar con vídeo en las aplicaciones, así como la forma de crear una barra de progreso que carga las animaciones.
- El Capítulo 16, “Trabajo con datos externos”, describe cómo procesar datos de fuentes externas utilizando scripts de servidor o de cliente en las aplicaciones. En este capítulo se explica cómo integrar datos con las aplicaciones.
- El Capítulo 17, “Aspectos básicos de la seguridad”, explica la seguridad en Flash Player, que se aplica al trabajo con archivos SWF situados localmente en el disco duro. En este capítulo se indican también los problemas de seguridad entre dominios y cómo cargar datos de servidores o entre dominios.
- El Capítulo 18, “Depuración de aplicaciones”, describe el depurador de ActionScript incluido en Flash que facilita la escritura de aplicaciones.
- El Capítulo 19, “Recomendaciones y convenciones de codificación para ActionScript 2.0”, explica cuáles son las prácticas recomendadas a la hora de utilizar Flash y escribir ActionScript. En este capítulo se enumeran también las convenciones de codificación estándar, como la asignación de nombre a las variables, además de otras convenciones.
- El Apéndice A, “Mensajes de error”, contiene la lista de mensajes de error que el compilador de Flash puede generar.
- El Apéndice B, “Operadores de Flash 4 no admitidos”, enumera todos los operadores de Flash 4 y su asociatividad.

- El Apéndice C, “Teclas del teclado y valores de códigos de tecla”, muestra todas las teclas de un teclado estándar y los valores correspondientes del código de tecla ASCII utilizados para identificar las teclas en ActionScript.
- El Apéndice D, “Escritura de scripts para versiones anteriores de Flash Player”, proporciona las directrices que le ayudarán a escribir scripts con una sintaxis correcta para la versión de reproductor que desea utilizar.
- El Apéndice E, “Programación orientada a objetos con ActionScript 1.0”, ofrece información sobre la utilización del modelo de objetos de ActionScript 1.0 para escribir scripts.
- El Apéndice F, “Terminología”, incluye una lista de la terminología que se utiliza con mayor frecuencia al trabajar con el lenguaje ActionScript y ofrece descripciones de los distintos términos.

En este manual se explica cómo utilizar el lenguaje ActionScript. Para obtener información sobre los elementos del lenguaje propiamente dichos, consulte la *Referencia del lenguaje ActionScript 2.0*.

## Convenciones tipográficas

En este manual se utilizan las convenciones tipográficas siguientes:

- La `fuente de código` indica que se trata de código de ActionScript.
- La `fuente de código en negrita`, que normalmente aparece dentro de un procedimiento, indica código que debe modificar o añadir al código que ya ha incorporado al archivo FLA. En algunos casos, puede utilizarse para resaltar código que se desea comprobar.
- El **texto en negrita** indica datos que debe escribir en la interfaz de usuario, como el nombre de un archivo o un nombre de instancia.
- El *texto en cursiva* indica un término nuevo definido en el texto que sigue. En una ruta de archivo, puede indicar un valor que debe sustituirse (por ejemplo, por el nombre de un directorio de su propio disco duro).

## Términos utilizados en este documento

En este manual se utilizan los términos siguientes:

- *Usted* hace referencia al desarrollador que escribe un script o una aplicación.
- *El usuario* hace referencia a la persona que ejecutará los scripts y las aplicaciones.
- La *fase de compilación* es la fase en la que se publica, exporta, prueba o depura un documento.
- El *tiempo de ejecución* es el espacio de tiempo en el que se ejecuta un script en Flash Player.

Los términos de ActionScript como *método* y *objeto* se definen en el Apéndice F, "Terminología", en la página 851.

## Copiar y pegar código

Al pegar código ActionScript desde el panel Ayuda al archivo FLA o ActionScript, se debe tener cuidado con los caracteres especiales, que incluyen las comillas dobles especiales (también denominadas comillas curvas). Estos caracteres no los interpreta el editor de ActionScript, por lo que el código genera un error si intenta compilarlo en Flash.

Se puede determinar que los caracteres de comillas son caracteres especiales cuando no presentan el código de color correcto. Es decir, si todas las cadenas no cambian de color en el editor de código, debe sustituir los caracteres especiales por las comillas rectas normales. Si se escriben las comillas simples o dobles directamente en el editor de ActionScript, se introducen siempre como comillas rectas. El compilador (al probar o publicar un archivo SWF) genera un error y le indica si en el código hay un tipo erróneo de caracteres (comillas especiales o curvas).

NOTA

También pueden aparecer las comillas especiales cuando se realiza la operación de pegar código ActionScript desde otras ubicaciones, como una página Web o un documento de Microsoft Word.

Controle que los saltos de línea sean los correctos al copiar y pegar código. Cuando se pega código desde otras ubicaciones, puede que la línea de código salte a la línea siguiente en una ubicación incorrecta. Asegúrese de que el código de color de la sintaxis es el correcto en el editor de ActionScript si piensa que los saltos de línea podrían ser un problema. Puede comparar el código del panel Acciones con el del panel Ayuda para ver si coinciden. Pruebe a activar la función Ajustar texto en el editor de ActionScript para ayudar a resolver el exceso de saltos de líneas del código (seleccione Ver > Ajustar palabra en la ventana Script, o bien Ajustar texto en el menú emergente del panel Acciones.)

# Recursos adicionales

Además de este manual sobre ActionScript, hay otros manuales sobre temas de Flash, como componentes y Macromedia Flash Lite. Se puede acceder a cada manual del panel Ayuda (Ayuda > Ayuda de Flash) desde la Tabla de contenido predeterminada. Haga clic en el botón Borrar para ver cada uno de los manuales disponibles; para más información, consulte "Dónde encontrar documentación sobre otros temas" en la página 18.

Para más información sobre los demás recursos disponibles, consulte los temas siguientes:



## Archivos de muestra

Hay disponibles numerosos archivos de muestra basados en ActionScript que se instalan con Flash. Estos archivos muestran cómo funciona el código en un archivo FLA y constituyen con frecuencia una herramienta de aprendizaje muy útil. En los capítulos de este manual se suele hacer referencia a estos archivos, pero se recomienda que también se consulte la carpeta de archivos de muestra del disco duro.

Los archivos de muestra incluyen archivos FLA de aplicaciones que utilizan funcionalidad común de Flash instalada con Flash. Estas aplicaciones se han diseñado para ayudar a los nuevos desarrolladores de Flash a adentrarse en las posibilidades de las aplicaciones Flash, así como para mostrar a los desarrolladores avanzados el modo en que funcionan las funciones de Flash en su contexto.

Los archivos de origen de muestra centrados en ActionScript se pueden encontrar en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/.

Puede que encuentre útil los siguientes archivos de muestra centrados en componentes, ya que contienen gran cantidad de código ActionScript. También se encuentran en la carpeta Samples del disco duro:

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\Components\.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/Components/.

También se pueden encontrar archivos de muestra adicionales para descargar en Internet. La siguiente página Web contiene vínculos y descripciones de archivos de muestra adicionales: www.macromedia.com/go/flash_samples_es/

## Dónde encontrar archivos PDF o documentación impresa

Si prefiere leer documentación en formato impreso, puede descargar las versiones en PDF de cada manual de la Ayuda. Vaya a www.macromedia.com/support/documentation/es/ y seleccione el producto en el que esté interesado. Puede ver o descargar el archivo PDF o el vínculo a la versión de LiveDocs del manual.

Normalmente, también puede adquirir la documentación impresa. Para obtener información actualizada, vaya al sitio de soporte de documentación y seleccione Flash Basic 8 o Flash Professional 8.

## LiveDocs

Se puede acceder a la documentación en el sitio Web de LiveDocs, además de poder hacerlo desde el panel Ayuda. El sitio Web de LiveDocs contiene todas las páginas de la Ayuda de Flash y podría contener comentarios que aclarasen, actualizasen o corrigiesen partes de la documentación. Haga clic en Ver comentarios en LiveDocs en la parte inferior de una página del panel Ayuda para ver la página equivalente en el sitio Web de LiveDocs. Vaya a http://livedocs.macromedia.com para ver una lista de la documentación disponible en formato de LiveDocs.

Escritores técnicos se encargan de supervisar el sitio Web de LiveDocs. Una de las ventajas de LiveDocs es la posibilidad de ver comentarios que aclaran aspectos de la documentación o corrigen cualquier errata o problema que haya surgido después de publicar la versión de software. LiveDocs no es el lugar apropiado para solicitar ayuda como formular preguntas sobre código que no funciona, realizar comentarios sobre problemas con el software y la instalación o preguntar cómo crear algo con Flash. Este sitio es para ofrecer comentarios sobre la documentación (por ejemplo, si se advierte que una frase o párrafo podría aclararse mejor).

Al hacer clic en el botón para añadir un comentario a LiveDocs, existen varios aspectos sobre los tipos de comentarios que son aceptables en el sistema. Lea atentamente estas directrices, ya que su comentario podría eliminarse del sitio Web.

Si desea realizar alguna pregunta sobre Flash, hágalo en los foros Web de Macromedia: www.macromedia.com/go/flash_forum_es. Los foros Web constituyen el mejor lugar para formular preguntas, ya que están supervisados por numerosos empleados de Macromedia, voluntarios de Team Macromedia, administradores y miembros de grupos de usuarios de Macromedia e incluso escritores técnicos.

Los ingenieros no supervisan el sistema de LiveDocs pero sí controlan la lista de Flash Wish. Si cree que ha encontrado un error en el software o le gustaría solicitar una mejora en Flash, rellene el formulario en www.macromedia.com/go/wish. Si informa del error o solicita una mejora en LiveDocs, no se añadirá oficialmente a la base de datos de errores. Debe utilizar el formulario específico (wishform) si desea que un ingeniero considere el error o la solicitud.

No olvide tener cuidado con los caracteres especiales y los saltos de línea al pegar desde la Web, incluido desde LiveDocs. Macromedia se ha esforzado al máximo por eliminar todos los caracteres especiales de los ejemplos de código, pero si tiene problemas al pegar código, consulte "Copiar y pegar código" en la página 14.

## Recursos en línea adicionales

Existen diversos recursos en línea que ofrecen formación, ayuda y asesoramiento a la hora de aprender a utilizar Macromedia Flash 8. Visite los siguientes sitios Web regularmente para obtener la información más reciente:

**El sitio Web Centro de desarrollo de Macromedia** (www.macromedia.com/go/developer_fl_es) se actualiza regularmente con la última información sobre Flash, además de consejos de usuarios expertos, temas más complejos, ejemplos, sugerencias, tutoriales (de incluso varias partes) y otras actualizaciones. Visite el sitio Web regularmente para conocer las últimas noticias sobre Flash y cómo sacar el máximo partido del programa.

**El centro de servicio técnico de Macromedia Flash** (www.macromedia.com/go/flash_support_es) proporciona notas técnicas, documentación actualizada y vínculos a otros recursos de la comunidad Flash.

**El sitio Web Macromedia Weblogs** (http://weblogs.macromedia.com) ofrece una lista de weblogs (también conocidos como *blogs*) tanto para el personal de Macromedia como para la comunidad de usuarios.

**Los foros Web de Macromedia** (http://webforums.macromedia.com) ofrecen numerosos foros en los que formular preguntas específicas sobre Flash, sus aplicaciones o el lenguaje ActionScript. Los foros están supervisados por voluntarios de Team Macromedia y es visitado con frecuencia por empleados de Macromedia. Si no está seguro de adónde acudir o cómo resolver un problema, visite un foro de Flash.

**El sitio Web Comunidad de Macromedia** (www.macromedia.com/community) suele albergar Macrochats, una serie de presentaciones en directo sobre diversos temas realizadas por empleados de Macromedia y miembros de la comunidad. Consulte con frecuencia el sitio Web para obtener la información más reciente y registrarse en los macrochats.

## Dónde encontrar documentación sobre otros temas

Los siguientes manuales ofrecen información adicional sobre temas asociados frecuentemente con ActionScript 2.0:

- Para obtener información sobre los elementos que componen el lenguaje ActionScript, consulte la *Referencia del lenguaje ActionScript 2.0*.
- Para obtener información sobre cómo trabajar en el entorno de edición de Flash, consulte *Cómo utilizar la Ayuda*.
- Si desea información sobre los componentes, consulte *Utilización de componentes*.

CAPÍTULO 1

# Novedades de Flash 8 ActionScript

1

Macromedia Flash Basic 8 y Macromedia Flash Professional 8 proporcionan varias mejoras que facilitan la escritura de scripts más robustos con el lenguaje ActionScript. Las nuevas funciones, que se describen más adelante en este capítulo, incluyen nuevos elementos de lenguaje (consulte "Elementos añadidos al lenguaje ActionScript" en la página 22), herramientas de edición mejoradas (consulte "Cambios de edición de ActionScript" en la página 28), cambios en el modelo de seguridad y otras mejoras relacionadas con el código ActionScript en la herramienta de edición.

Para más información, consulte los siguientes temas:



## Nuevas funciones en ActionScript 2.0 y Flash 8

El lenguaje ActionScript ha crecido y se ha desarrollado desde su introducción hace algunos años. Con cada nueva versión de Flash se han añadido a ActionScript palabras clave, objetos, métodos y otros elementos de lenguaje adicionales. Asimismo se han incluido mejoras relativas a ActionScript en los entornos de edición de Flash 8. Flash Basic 8 y Flash Professional 8 presentan nuevos elementos de lenguaje para mejorar la expresividad, como filtros y modos de mezcla, y el desarrollo de aplicaciones, como integración con JavaScript (ExternalInterface) y entrada y salida de archivos (FileReference y FileReferenceList).

Esta sección proporciona información general sobre los elementos y clases del lenguaje ActionScript que son nuevos o han cambiado en Flash 8, así como las mejoras relativas a ActionScript en la herramienta de edición. Para obtener una lista específica de las funciones añadidas a ActionScript 2.0, consulte "Elementos añadidos al lenguaje ActionScript" en la página 22. Para utilizar cualquiera de los nuevos elementos del lenguaje en los scripts, debe utilizar Flash Player 8 (el predeterminado) al publicar los documentos.

Las funciones siguientes se añadieron tanto a Flash Basic 8 como a Flash Professional 8 (a menos que se indique lo contrario):

- Las mejoras del editor de ActionScript permiten mostrar los caracteres ocultos en los scripts. Para más información, consulte "Visualización de caracteres ocultos" en la página 57.
- Las opciones de depuración se encuentran ahora disponibles en la ventana Script, así como el panel Acciones, para los archivos ActionScript.
- Se ha reorganizado el directorio Configuration que incluye los archivos XML y de clases. Consulte "Archivos de configuración que se instalan con Flash 8" en la página 69 para obtener más detalles.
- Puede definir una preferencia para volver a cargar archivos de script modificados mientras trabaja en una aplicación, lo que le ayuda a evitar trabajar con versiones anteriores de archivos de script y sobrescribir los que sean más nuevos. Para más información, consulte "Preferencias de ActionScript" en la página 45.
- La función de la ventana Script se encuentra disponible en Flash Basic 8 y Flash Professional 8, lo que indica que puede crear ahora un archivo de ActionScript en cada programa.
- El asistente de script (similar al modo Normal en las versiones anteriores de Flash) le ayuda a codificar sin necesidad de entender la sintaxis. Para más información sobre el asistente de script, consulte "Asistente de script" en la página 62.
- Puede cargar nuevos tipos de archivos de imagen en tiempo de ejecución, que incluyen imágenes JPEG progresivas, así como archivos GIF y PNG no animados. Si carga un archivo animado, aparece el primer fotograma de la animación.
- Puede asignar identificadores de vinculación a los archivos de mapas de bits y de sonido almacenados en la biblioteca, lo que significa que puede asociar imágenes al escenario o trabajar con estos elementos en bibliotecas compartidas.
- La caché de mapa de bits permite mejorar el rendimiento de las aplicaciones en tiempo de ejecución al almacenar en caché una representación de mapa de bits de las instancias. Puede utilizar código ActionScript para acceder a esta propiedad. Para más información, consulte "Caché de mapa de bits, desplazamiento y rendimiento" en la página 513.
- La escala de 9 divisiones permite escalar instancias de clip de película sin ampliar los trazos que forman el contorno del clip. Puede utilizar código ActionScript para acceder a esta función en Flash Basic 8 y Flash Professional 8, o bien en la herramienta de edición de Flash 8. Para más información, consulte "Utilización de escala en 9 divisiones en ActionScript" en la página 592. Para obtener información sobre el acceso a la escala de 9 divisiones en la herramienta de edición, consulte "La escala en 9 divisiones y los símbolos de clip de película" en la página 90 en *Utilización de Flash*.

- Ahora puede añadir información de metadatos a los archivos FLA en el cuadro de diálogo Configuración de publicación. Con el cuadro de diálogo se puede añadir un nombre y una descripción al archivo FLA para ayudar a aumentar la visibilidad en la búsqueda en línea.
- Se ha mejorado el panel Cadenas a fin de admitir texto con varias líneas en el campo Cadena y un archivo XML de idioma. Para más información, consulte "Panel Cadenas" en la página 482.
- Se ha incorporado en Flash Player un nuevo recolector de datos innecesarios, que utiliza un recolector incremental para mejorar el rendimiento.
- Se ha mejorado el flujo de trabajo para crear aplicaciones accesibles. Flash Player 8 ya no precisa que los desarrolladores añadan todos los objetos al índice de tabulación para que el lector de pantalla lea correctamente el contenido. Para más información sobre el índice de tabulación, consulte `tabIndex (Button.tabIndex property)`, `tabIndex (MovieClip.tabIndex property)` y `tabIndex (TextField.tabIndex property)` en *Referencia del lenguaje ActionScript 2.0*.
- Flash Player ha mejorado la seguridad de archivos local, con seguridad adicional al ejecutar archivos SWF en el disco duro. Para obtener información sobre la seguridad de archivos local, consulte "Seguridad de archivos local y Flash Player" en la página 717.
- Con el código ActionScript, puede utilizar la API de dibujo para controlar el estilo de línea de los trazos que dibuje. Para más información sobre los nuevos estilos de línea, consulte "Utilización de estilos de línea" en la página 581.
- Con el código ActionScript, puede utilizar la API de dibujo para crear degradados más complejos con los que rellenar formas. Para obtener información sobre los rellenos de degradados, consulte "Utilización de rellenos con degradado complejos" en la página 580.
- Puede utilizar código ActionScript para aplicar muchos filtros a objetos en el escenario (como instancias de clip de película). Para obtener información sobre los filtros y ActionScript, consulte "Utilización de filtros con código ActionScript" en la página 540.
- Puede utilizar la API de FileReference y FileReferenceList para cargar archivos en el servidor. Para más información, consulte "Carga y descarga de archivos" en la página 682.
- Puede utilizar código ActionScript para acceder a nuevas y avanzadas formas de aplicar y manipular los colores. Para más información, consulte "Configuración de valores de color" en la página 606 y `ColorTransform (flash.geom.ColorTransform)` en *Referencia del lenguaje ActionScript 2.0*.
- Se han aplicado numerosas mejoras al manejo del texto, incluidas nuevas opciones, propiedades y parámetros en las clases TextField y TextFormat. Para más información, consulte `TextField` y `TextFormat` en *Referencia del lenguaje ActionScript 2.0*.

- El código ActionScript se puede utilizar para acceder a funciones de suavizado avanzado (FlashType). Para más información, consulte "Representación de fuentes y texto suavizado" en la página 434.
- Puede eliminar archivos ASO al probar la aplicación. Seleccione Control > Eliminar archivos ASO o Control > Eliminar archivos ASO y probar película en la herramienta de edición. Para más información, consulte "Utilización de archivos ASO" en la página 254.

Para obtener una lista de clases, elementos de lenguaje, métodos y propiedades específicas añadidas a ActionScript 2.0 en Flash 8, consulte "Elementos añadidos al lenguaje ActionScript" en la página 22.

## Elementos añadidos al lenguaje ActionScript

En esta sección se enumeran los nuevos elementos y clases del lenguaje ActionScript que son nuevos o se han modificado en Flash 8. Las siguientes clases y elementos de lenguaje son nuevas adiciones o acaban de admitirse en Flash Player 8.

Las siguientes clases se añadieron a ActionScript 2.0 en Flash 8:

- La clase BevelFilter (en el paquete flash.filters) permite añadir efectos de bisel a los objetos.
- La clase BitmapData (en el paquete flash.display) permite crear y manipular imágenes de mapas de bits transparentes u opacas de tamaño arbitrario.
- La clase BitmapFilter (en el paquete flash.display) es una clase base para todos los efectos de filtro.
- La clase BlurFilter permite aplicar desenfoques a los objetos de Flash.
- La clase ColorMatrixFilter (en el paquete flash.filters) permite aplicar transformaciones a los valores de colores ARGB y alfa.
- La clase ColorTransform (en el paquete flash.geom) permite ajustar valores de colores en los clips de película. La clase Color deja de admitirse en favor de esta clase.
- La clase ConvolutionFilter (en el paquete flash.filters) permite aplicar efectos de filtro de convolución de matrices.
- La clase DisplacementMapFilter (en el paquete flash.filters) permite utilizar valores de píxel de un objeto BitmapData para realizar el desplazamiento de un objeto.
- La clase DropShadowFilter (en el paquete flash.filters) permite añadir efectos de sombras sesgadas a los objetos.
- La clase ExternalInterface (en el paquete flash.external) permite comunicarse mediante ActionScript con el contenedor de Flash Player (el sistema que contiene la aplicación Flash, como un navegador con JavaScript o la aplicación de escritorio).

- La clase FileReference (en el paquete flash.net) permite cargar y descargar archivos entre el equipo del usuario y un servidor.
- La clase FileReferenceList (en el paquete flash.net) permite seleccionar uno o más archivos para cargar.
- La clase GlowFilter (en el paquete flash.filters) permite añadir efectos de iluminación a los objetos.
- La clase GradientBevelFilter (en el paquete flash.filters) permite añadir efectos de biseles degradados a los objetos.
- La clase GradientGlowFilter (en el paquete flash.filters) permite añadir efectos de iluminación degradada a los objetos.
- La clase IME (en la clase System) permite manipular el editor de método de entrada (IME) del sistema operativo en Flash Player.
- La clase Locale (en el paquete mx.lang) permite controlar cómo aparece texto en varios idiomas en un archivo SWF.
- La clase Matrix (en el paquete flash.geom) representa una matriz de transformación que determina cómo asignar puntos de un espacio de coordenadas a otro.
- La clase Point (en el paquete flash.geom) representa una ubicación en un sistema de coordenadas bidimensional (*x* representa el eje horizontal e *y* el eje vertical).
- La clase Rectangle (en el paquete flash.geom) permite crear y modificar objetos Rectangle.
- La clase TextRenderer (en el paquete flash.text) proporciona la funcionalidad para el suavizado de las fuentes incorporadas.
- La clase Transform (en el paquete flash.geom) recopila datos sobre las transformaciones de color y manipulaciones de coordenadas que se aplican a una instancia MovieClip.

| NOTA | Se ha añadido compatibilidad oficial para la clase AsBroadcaster en Flash 8. |
|---|---|

Los nuevos elementos de lenguaje, métodos y funciones añadidas a las clases existentes en ActionScript incluyen:

- La función global `showRedrawRegions` proporciona la capacidad al reproductor depurador para que perfile las regiones de la pantalla que se están dibujando de nuevo (es decir, las regiones sucias que se están actualizando). La función dispone del reproductor que muestra lo que se ha vuelto a dibujar, pero no le permite controlar estas regiones.
- La propiedad `blendMode` de la clase Button, que establece el modo de mezcla para la instancia de botón.
- La propiedad `cacheAsBitmap` de la clase Button, que permite guardar en caché el objeto como una representación interna de mapa de bits de la instancia.

- La propiedad `filters` de la clase Button, que es una matriz indexada que contiene cada objeto de filtro asociado al botón.
- La propiedad `scale9Grid` de la clase Button, que es la región rectangular que define nueve regiones de escala de la instancia.
- La propiedad `hasIME` de la clase System.capabilities, que indica si el sistema tiene un editor de método de entrada (IME) instalado.
- La propiedad `getUTCYear` de la clase Date, que devuelve el año de esta fecha, según la hora universal.
- El método `isAccessible()` de la clase Key devuelve un valor booleano que indica si otros archivos SWF pueden acceder a la última tecla pulsada, según las restricciones de seguridad.
- El controlador de eventos `onHTTPStatus` de la clase LoadVars devuelve el código de estado que se devuelve del servidor (por ejemplo, el valor 404 para la página no encontrada). Para más información, consulte `onHTTPStatus (LoadVars.onHTTPStatus handler)` en *Referencia del lenguaje ActionScript 2.0*.
- El método `attachBitmap()` de la clase MovieClip, que asocia una imagen de mapa de bits a un clip de película. Para más información, consulte `BitmapData (flash.display.BitmapData)` en *Referencia del lenguaje ActionScript 2.0*.
- El método `beginBitmapFill()` de la clase MovieClip, que asocia un clip de película a una imagen de mapa de bits.
- Los parámetros `spreadMethod`, `interpolationMethod` y `focalPointRatio` del método `beginGradientFill()` en la clase MovieClip. Este método rellena un área de dibujo con una imagen de mapa de bits y el mapa de bits se puede repetir o incluir en un mosaico para rellenar el área.
- La propiedad `blendMode` de la clase MovieClip, que permite establecer el modo de mezcla para la instancia.
- La propiedad `cacheAsBitmap` de la clase MovieClip, que permite guardar en caché el objeto como una representación interna de mapa de bits de la instancia.
- La propiedad `filters` de la clase MovieClip, que es una matriz indexada que contiene cada objeto de filtro asociado actualmente a la instancia.
- El método `getRect()` de la clase MovieClip, que devuelve las propiedades que son los valores mínimo y máximo de las coordenadas de la instancia especificada.
- El método `lineGradientStyle()` de la clase MovieClip, que especifica un estilo de línea de degradado que utiliza Flash al dibujar una ruta.

- Los parámetros `pixelHinting`, `noScale`, `capsStyle`, `jointStyle` y `miterLimit` del método `lineStyle()` en la clase MovieClip. Estos parámetros especifican tipos de estilos de línea que se pueden utilizar al dibujar líneas.
- La propiedad `opaqueBackground` de la clase MovieClip, que establece el color del fondo opaco (no transparente) del clip de película con el color que especifica el valor hexadecimal RGB.
- La propiedad `scale9Grid` de la clase MovieClip, que es la región rectangular que define nueve regiones de escala de la instancia.
- La propiedad `scrollRect` de la clase MovieClip, que permite desplazar rápidamente el contenido del clip de película y abrir una ventana que muestre mayor cantidad de contenido.
- La propiedad `transform` de la clase MovieClip, que permite fijar la configuración en relación a la matriz, transformación de color y límites de píxel de un clip de película. Para más información, consulte `Transform (flash.geom.Transform)` en *Referencia del lenguaje ActionScript 2.0*.
- El parámetro `status` del controlador de eventos `MovieClipLoader.onLoadComplete` devuelve el código de estado que se devuelve del servidor (por ejemplo, el valor 404 para la página no encontrada). Para más información, consulte `onLoadComplete (MovieClipLoader.onLoadComplete event listener)` en *Referencia del lenguaje ActionScript 2.0*.
- El controlador de eventos `onLoadError` de la clase MovieClipLoader se invoca cuando no se carga un archivo cargado con `MovieClipLoader.loadClip()`.
- El parámetro `secure` del método `SharedObject.getLocal()` determina si el acceso a este objeto compartido se restringe a los archivos SWF transmitidos a través de una conexión HTTPS. Para más información, consulte `getLocal (SharedObject.getLocal method)` en *Referencia del lenguaje ActionScript 2.0*.
- La propiedad sandboxType de la clase System.security indica el tipo de modelo de seguridad del entorno local en el que está funcionando el archivo SWF que realiza la llamada. Para más información, consulte `sandboxType (security.sandboxType property)` en *Referencia del lenguaje ActionScript 2.0*.
- La propiedad `antiAliasType` de la clase TextField, que establece el tipo de suavizado que utiliza para la instancia TextField.
- La propiedad `filters` de la clase TextField, que es una matriz indexada que contiene cada objeto de filtro asociado actualmente a la instancia TextField.

- La propiedad `gridFitType` de la clase TextField, que establece el tipo de ajuste de cuadrícula que utiliza para la instancia. Para obtener información sobre el ajuste de cuadrícula y TextField.gridFitType, consulte `gridFitType (TextField.gridFitType property)` en *Referencia del lenguaje ActionScript 2.0*.
- La propiedad `sharpness` de la clase TextField, que establece la nitidez de los bordes de glifo para la instancia TextField. Debe establecer el método `antiAliasType()` como avanzado si utiliza esta propiedad.
- La propiedad `thickness` de la clase TextField, que establece el grosor de los bordes de glifo en la instancia TextField. Debe establecer el método `antiAliasType()` como avanzado si utiliza esta propiedad.
- El valor `justify` para la propiedad `align` de la clase TextFormat, que permite justificar un párrafo especificado.
- La propiedad `indent` de la clase TextFormat, que permite utilizar valores negativos.
- La propiedad `kerning` de la clase TextFormat, que permite activar o desactivar el ajuste entre caracteres para el objeto TextFormat.
- La propiedad `leading` de la clase TextFormat, que permite utilizar el interlineado negativo, de modo que el espacio entre líneas es menor que la altura del texto. De este modo, puede aproximar las líneas de texto en las aplicaciones.
- La propiedad `letterSpacing` de la clase TextFormat, que permite especificar la cantidad de espacio distribuido de forma uniforme entre los caracteres.
- La propiedad `_alpha` de la clase Video, que es la cantidad especificada de transparencia para el objeto de vídeo.
- La propiedad `_height` de la clase Video, que indica la altura de la instancia de vídeo.
- La propiedad `_name` de la clase Video, que indica el nombre de la instancia de vídeo.
- La propiedad `_parent` de la clase Video, que indica la instancia de clip de película u objeto que contiene la instancia de vídeo.
- La propiedad `_rotation` de la clase Video, que permite establecer la cantidad de rotación en grados de la instancia de vídeo.
- La propiedad `_visible` de la clase Video class, que permite establecer la visibilidad de una instancia de vídeo.
- La propiedad `_width` de la clase Video, que permite establecer la anchura de la instancia de vídeo.
- La propiedad `_x` de la clase Video, que permite establecer la coordenada *x* de la instancia de vídeo.
- La propiedad `_xmouse` de la clase Video, que permite establecer la coordenada *x* de la posición del puntero del ratón.

- La propiedad `_xscale` de la clase Video, que permite establecer el porcentaje de la escala horizontal de la instancia de vídeo.
- La propiedad `_y` de la clase Video, que permite establecer la coordenada *y* de la instancia de vídeo.
- La propiedad `_ymouse` de la clase Video, que permite establecer la coordenada *y* de la posición del puntero del ratón.
- La propiedad `_yscale` de la clase Video, que permite establecer el porcentaje de la escala vertical de la instancia de vídeo.
- El controlador de eventos `onHTTPStatus` de la clase XML devuelve el código de estado que se devuelve del servidor (por ejemplo, el valor 404 para la página no encontrada). Para más información, consulte `onHTTPStatus (XML.onHTTPStatus handler)` en *Referencia del lenguaje ActionScript 2.0*.
- La propiedad `localName` de la clase XMLNode, que devuelve el nombre completo del objeto de nodo XML (incluidos tanto el prefijo como el nombre local).
- La propiedad `namespaceURI` de la clase XMLNode, que lee el URI del espacio de nombres en el que se resuelve el prefijo del nodo XML. Para más información, consulte `namespaceURI (XMLNode.namespaceURI property)` en *Referencia del lenguaje ActionScript 2.0*.
- La propiedad `prefix` de la clase XMLNode, que lee el prefijo del nombre del nodo.
- El método `getNamespaceForPrefix()` de la clase XMLNode, que devuelve el URI del espacio de nombres asociado al prefijo especificado para el nodo.
- El método `getPrefixForNamespace` de la clase XMLNode, que devuelve el prefijo asociado al URI del espacio de nombres especificado para el nodo.

## Elementos de lenguaje no admitidos

Algunos elementos de lenguaje no se admiten en Flash Player 8. Para obtener una lista de los elementos de lenguaje no admitidos y alternativas que pueden utilizarse en Flash Player 8, consulte las siguientes secciones en *Referencia del lenguaje ActionScript 2.0*:

- `Deprecated Class summary`
- `Deprecated Function summary`
- `Deprecated Property summary`
- `Deprecated Operator summary`

# Cambios de edición de ActionScript

El editor de ActionScript en el panel Acciones y la ventana Script se ha actualizado de varias formas para que resulte más robusto y fácil de utilizar que las versiones anteriores de la herramienta. En esta sección se ofrece un resumen de dichos cambios.

**Ver caracteres ocultos** Ahora puede utilizar el menú emergente Opciones de los paneles Script, Depurador y Salida para ver u ocultar caracteres cuando escribe archivos de script en el panel Acciones o la ventana Script. Para obtener información sobre esta función, consulte "Visualización de caracteres ocultos" en la página 57.

**Asistente de script añadido al panel Acciones** En las versiones anteriores de Flash, podía trabajar en el panel Acciones en *modo Normal* (se especificaban las opciones y los parámetros para crear el código) o en *modo Experto* (se añadían comandos directamente en el panel Script). Estas opciones no estaban disponibles en Flash MX 2004 ni en Flash MX Professional 2004. Sin embargo, en Flash Basic 8 y Flash Professional 8, puede trabajar en el *modo de asistente de script*, que es similar al modo normal existente en versiones anteriores de Flash (pero más robusto). Para obtener información sobre el asistente de script, consulte el Capítulo 13, "Escritura de ActionScript con el asistente de script" en *Utilización de Flash*. Para realizar un tutorial sobre el asistente de script, consulte el Capítulo 13, "Creación de un evento startDrag/stopDrag con el Asistente de script" en *Utilización de Flash*.

**Volver a cargar archivos modificados** Puede volver a cargar archivos de script modificados al trabajar en una aplicación. Aparece un mensaje de advertencia en el que se le solicita que vuelva a cargar los archivos de script modificados asociados con la aplicación con la que está trabajando. Esta función resulta especialmente beneficiosa para equipos que trabajan en aplicaciones al mismo tiempo, ya que ayuda a evitar que se trabaje con scripts desactualizados o se sobrescriban versiones más recientes de un script. Si se ha movido o eliminado un archivo de script, aparece un mensaje de advertencia en el que se le indica que guarde los archivos si es necesario. Para más información, consulte "Preferencias de ActionScript" en la página 45.

# Cambios en el modelo de seguridad para archivos SWF instalados localmente

Flash Player 8 cuenta con un nuevo modelo de seguridad mejorado en el que las aplicaciones de Flash y los archivos SWF de un equipo local pueden comunicarse con Internet y el sistema de archivos local en vez de ejecutarse desde un servidor Web remoto. Al desarrollar una aplicación de Flash, debe indicar si se permite que el archivo SWF se comunique con una red o con un sistema de archivos local.

| NOTA | En esta descripción, un *archivo SWF local* debe entenderse como un archivo SWF instalado localmente en el equipo de un usuario, no como archivo servido desde un sitio Web, y que no incluye archivos de proyector (EXE). |
|---|---|

En las versiones anteriores de Flash Player, los archivos SWF locales podían interactuar con otros archivos SWF y cargar datos de otro equipo local o remoto sin establecer la configuración de seguridad. En Flash Player 8, un archivo SWF no puede realizar conexiones con el sistema de archivos local y la red (como Internet) en la misma aplicación sin que se haya establecido un parámetro de seguridad. Se trata de una medida de seguridad; de este modo, un archivo SWF no puede leer archivos del disco duro ni enviar el contenido de dichos archivos por Internet.

Esta restricción de seguridad afecta a todo el contenido implementado localmente, tanto si es contenido antiguo (un archivo FLA creado en una versión anterior de Flash) como creado en Flash 8. Con Flash MX 2004 o una herramienta de edición anterior, podía probar una aplicación de Flash que se ejecutara localmente y también acceder a Internet. En Flash Player 8, esta aplicación solicita ahora al usuario que dé su permiso para que pueda comunicarse con Internet.

Cuando se prueba un archivo en el disco duro, hay varios pasos para determinar si el archivo es un documento local de confianza (seguro) o un documento que potencialmente no es de confianza (no seguro). Si crea el archivo en el entorno de edición de Flash (por ejemplo, cuando selecciona Control > Probar película), el archivo es de confianza porque se encuentra dentro del entorno de prueba.

En Flash Player 7 y en versiones anteriores, los archivos SWF locales tenían permisos para acceder al sistema de archivos local y a la red. En Flash Player 8, los archivos SWF locales tienen tres niveles de permiso:

- Acceder al sistema de archivos local solamente (el nivel predeterminado). El archivo SWF local puede leer desde las rutas de red de convención de nomenclatura universal (UNC) y sistema de archivos local y no se puede comunicar con Internet.

- Acceder sólo a la red. El archivo SWF local puede acceder sólo a la red (como Internet) y no al sistema de archivos local donde está instalado el archivo SWF.
- Acceder tanto al sistema de archivos local como a la red. El archivo SWF local puede leer del sistema de archivos local en el que está instalado el archivo, leer y escribir en cualquier servidor que le conceda permiso y usar scripts con otros archivos SWF de la red o del sistema de archivos local que le conceda permiso.

Para más detalles sobre cada nivel de permiso, consulte "Seguridad de archivos local y Flash Player" en la página 717.

Existen también leves cambios en System.security.allowDomain y mejoras en System.security.allowInsecureDomain. Para más información sobre la seguridad de archivos local, consulte el Capítulo 17, "Aspectos básicos de la seguridad".

CAPÍTULO 2

# Escritura y edición en ActionScript 2.0

2

Al escribir código ActionScript en Macromedia Flash Basic 8 o Macromedia Flash Professional 8, se utiliza el panel Acciones o la ventana Script. El panel Acciones y la ventana Script contienen un editor de código completo (denominado *editor de ActionScript*) que incluye sugerencias y consejos para el código, coloreado y aplicación de formato del código, resaltado de sintaxis, revisión de sintaxis, depuración, números de línea, ajuste de texto y compatibilidad con Unicode en dos vistas distintas. Para más información sobre el editor de ActionScript, consulte "Utilización del panel Acciones y la ventana Script" en la página 36.

Puede utilizar uno de dos métodos existentes para escribir código ActionScript en Flash. Puede escribir scripts que forman parte del documento de Flash (es decir, scripts incorporados en el archivo FLA), o bien escribir scripts externos (scripts o clases almacenados en archivos externos). Sin embargo, no puede utilizar el panel Acciones para escribir scripts externos.

Al escribir scripts en un archivo FLA, se utiliza el editor de ActionScript del panel Acciones. Este panel contiene el editor de ActionScript en un panel Script y herramientas para facilitar la escritura de scripts. Entre estas herramientas, se incluye la caja de herramientas Acciones, que le proporciona acceso rápido a los principales elementos del lenguaje ActionScript, el navegador de scripts, que le ayuda a desplazarse por los scripts del documento, y el modo de asistente de script, que le solicita el elemento necesario para crear scripts. Para más información sobre el panel Acciones, consulte "Panel Acciones" en la página 37. Para más información sobre el asistente de script, consulte "Asistente de script" en la página 62.

Cuando necesite crear un script externo, deberá utilizar la ventana Script del editor de ActionScript para crear un nuevo archivo ActionScript. (También puede utilizar su editor de texto favorito para crear un archivo AS externo.) En la ventana Script, el editor de ActionScript incluye funciones de ayuda para el código, como sugerencias y colores de código o revisión de la sintaxis, entre otras, del mismo modo que el panel Acciones. Para más información sobre la ventana Script, consulte "Ventana Script" en la página 38.

Flash proporciona ayuda adicional para crear scripts a través de comportamientos. Los comportamientos son funciones predefinidas de ActionScript que puede asociar a los objetos del documento de Flash y que evitan que tenga que crear el código ActionScript manualmente. Para más información sobre comportamientos, consulte "Comportamientos" en la página 65.

Para más información sobre el control de eventos, consulte las secciones siguientes:



# ActionScript y eventos

En Macromedia Flash Basic 8 y Macromedia Flash Professional 8, el código ActionScript se ejecuta cuando se produce un evento: por ejemplo, cuando se carga un clip de película, cuando se entra en un fotograma clave de la línea de tiempo o cuando el usuario hace clic en un botón. Los eventos puede accionarlos el usuario o el sistema. Los usuarios hacen clic en botones y presionan teclas; el sistema acciona eventos cuando se cumplen condiciones específicas o finalizan procesos (se carga el archivo SWF, la línea de tiempo llega a un determinado fotograma, termina de cargarse un gráfico, etc.).

Deberá escribir un *controlador de evento* que responda con una acción cuando tenga lugar el evento. Comprender cuándo y dónde tienen lugar los eventos le ayudará a determinar cómo y dónde deberá responder al evento con una acción y qué herramientas de ActionScript utilizará en cada caso. Para más información, consulte "Escritura de scripts para controlar eventos" en la página 35.

Los eventos pueden agruparse en varias categorías: eventos de ratón y de teclado, que tienen lugar cuando un usuario interactúa con la aplicación Flash a través del ratón y el teclado; eventos de clip, que se producen en los clips de película y los eventos de fotograma, que tienen lugar en los fotogramas de la línea de tiempo.

Para obtener información sobre los tipos de scripts que puede escribir para controlar eventos, consulte "Escritura de scripts para controlar eventos" en la página 35.

## Eventos de ratón y teclado

Un usuario que interactúa con la aplicación o el archivo SWF desencadena eventos de ratón y de teclado. Por ejemplo, cuando el usuario pasa el puntero por encima de un botón, tiene lugar el evento `Button.onRollOver` o `on (rollOver)`; cuando hace clic en un botón, tiene lugar el evento `Button.onRelease`; si se presiona una tecla del teclado, tiene lugar el evento `on (keyPress)`. Puede escribir código en un fotograma o asociar scripts a una instancia para controlar estos eventos y añadir toda la interactividad que desee.

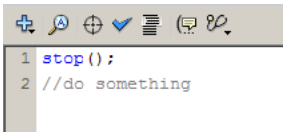
## Eventos de clip

Dentro de un clip de película, puede reaccionar a una serie de eventos de clip que se desencadenan cuando el usuario entra o sale de la escena o interactúa con la misma mediante el ratón o el teclado. Por ejemplo, podría cargar un archivo SWF o de imagen JPG externo en el clip de película cuando el usuario entrase en la escena o permitir que los movimientos del ratón del usuario cambiasen de posición los elementos de la escena.

## Eventos de fotograma

En una línea de tiempo principal o de clip de película, un evento de sistema tiene lugar cuando la cabeza lectora entra en un fotograma clave (esto se conoce como *evento de fotograma*). Los eventos de fotograma son útiles para desencadenar acciones en función del paso del tiempo (el avance por la línea de tiempo) o para interactuar con elementos que actualmente están visibles en el escenario. Al añadir un script a un fotograma clave, el script se ejecuta cuando se llega al fotograma clave durante la reproducción. Un script asociado a un fotograma se denomina *script de fotograma*.

Uno de los usos más habituales de los scripts de fotograma es para detener la reproducción cuando se alcanza un determinado fotograma clave. Esto se consigue mediante la función `stop()`. Se selecciona un fotograma clave y luego se añade la función `stop()` como elemento de script en el panel Acciones.

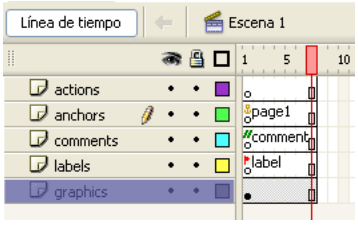



Una vez que haya detenido el archivo SWF en un determinado fotograma clave, deberá realizar alguna acción. Por ejemplo, podría utilizar un script de fotograma para actualizar dinámicamente el valor de una etiqueta, para gestionar la interacción de los elementos del escenario, etc.

# Organización de código ActionScript

Puede asociar scripts a fotogramas clave y a instancias de objetos (clips de película, botones y otros símbolos). Sin embargo, si dispersa el código ActionScript entre muchos fotogramas clave e instancias de objetos, la depuración de la aplicación resultará mucho más compleja. Además, resultará imposible que varias aplicaciones de Flash compartan el mismo código. Por consiguiente, es importante seguir las recomendaciones de codificación al crear código ActionScript en Flash.

En lugar de asociar los scripts a elementos como fotogramas clave, clips de película y botones, debe responder a los eventos llamando a funciones que residan en una ubicación central. Uno de los métodos consiste en adjuntar código ActionScript incorporado al primer o al segundo fotograma de la línea de tiempo cuando sea posible, de manera que no tenga que buscar por todo el archivo FLA para localizar todo el código. Otra práctica habitual consiste en crear una capa llamada *actions* y colocar el código ActionScript en ella.




Al asociar todos los scripts a elementos individuales, está incorporando todo el código en el archivo FLA. Si es importante que el código puedan compartirlo otras aplicaciones de Flash, utilice la ventana Script o su editor de texto preferido para crear un archivo ActionScript (AS) externo.

Al crear un archivo externo, consigue que el código sea más modular y esté mejor organizado. Conforme crezca el proyecto, observará cómo este enfoque resulta mucho más útil de lo que en un principio pudiera imaginar. Un archivo externo facilita la depuración y el control del código fuente si trabaja en un proyecto con otros desarrolladores.

Para utilizar el código ActionScript contenido en un archivo AS externo, deberá crear un script dentro del archivo FLA y luego utilizar la sentencia `#include` para acceder al código que ha guardado externamente, como se muestra en el ejemplo siguiente:

```
#include "../core/Functions.as"
```

También puede utilizar ActionScript 2.0 para crear clases personalizadas. Debe almacenar las clases personalizadas en archivos AS externos y utilizar sentencias `import` en un script para que las clases se exporten al archivo SWF, en lugar de utilizar sentencias `#include`. Para más información sobre la escritura de archivos de clases, consulte "Escritura de archivos de clases personalizadas" en la página 205 y "Importación de archivos de clases" en la página 209 sobre la importación de archivos de clases. También puede utilizar componentes (clips de película creados previamente) para compartir código y funcionalidad como, por ejemplo, scripts y elementos de la interfaz de usuario.

| NOTA | El código ActionScript de los archivos externos se compila en un archivo SWF cuando se publica, se exporta o se depura un archivo FLA. Por lo tanto, si realiza modificaciones en un archivo externo deberá guardarlo y volver a compilar todos los archivos FLA que lo utilizan. |
|---|---|

Al escribir código ActionScript en Flash 8, se utiliza el panel Acciones, la ventana Script, o ambos. La conveniencia de uno u otro vendrá determinada por la forma en que responda a los eventos, cómo organice el código y, por encima de todo, las prácticas de codificación más recomendables.

Para más información sobre prácticas recomendadas y convenciones de codificación, consulte "Convenciones de codificación de ActionScript" en la página 790.

Al utilizar comportamientos, que son funciones de ActionScript predefinidas (consulte "Comportamientos" en la página 65), se deben tener en cuenta otros problemas de flujo de trabajo y organización del código.

## Escritura de scripts para controlar eventos

La escritura de código de eventos se puede dividir en dos categorías principales: los eventos que tienen lugar en la línea de tiempo (en fotogramas clave) y los que tienen lugar en instancias de objeto (clips de película, botones y otros componentes). La interactividad de la aplicación o archivo SWF puede distribuirse entre los numerosos elementos que integran el proyecto y es posible que tenga la tentación de añadir scripts directamente a estos elementos. Sin embargo, Macromedia recomienda no añadir scripts directamente a estos elementos (fotogramas clave y objetos). Por el contrario, debe responder a los eventos llamando a funciones que residen en una ubicación central, como se describe en "Organización de código ActionScript".

# Utilización del panel Acciones y la ventana Script

Para crear scripts en un archivo FLA, se introduce código ActionScript directamente en el panel Acciones. Para crear scripts externos que incluirá o importará a la aplicación, puede utilizar la ventana Script (Archivo > Nuevo y seleccione Archivo ActionScript) o su editor de texto preferido.

Cuando se utiliza el panel Acciones o la ventana Script, se utilizan las funciones del editor de ActionScript para escribir, aplicar formato y editar el código. Tanto el panel Acciones como la ventana Script cuentan con un panel Script (que es donde se escribe el código) y la caja de herramientas Acciones. El panel Acciones ofrece algunas funciones más de ayuda para el código que la ventana Script. Flash ofrece estas funciones en el panel Acciones porque resultan especialmente útiles en el contexto de edición de ActionScript dentro de un archivo FLA.

### Para mostrar el panel Acciones, realice uno de los siguientes procedimientos:

- Seleccione Ventana > Acciones.
- Presione F9.

### Para mostrar la ventana Script, realice uno de los siguientes procedimientos:

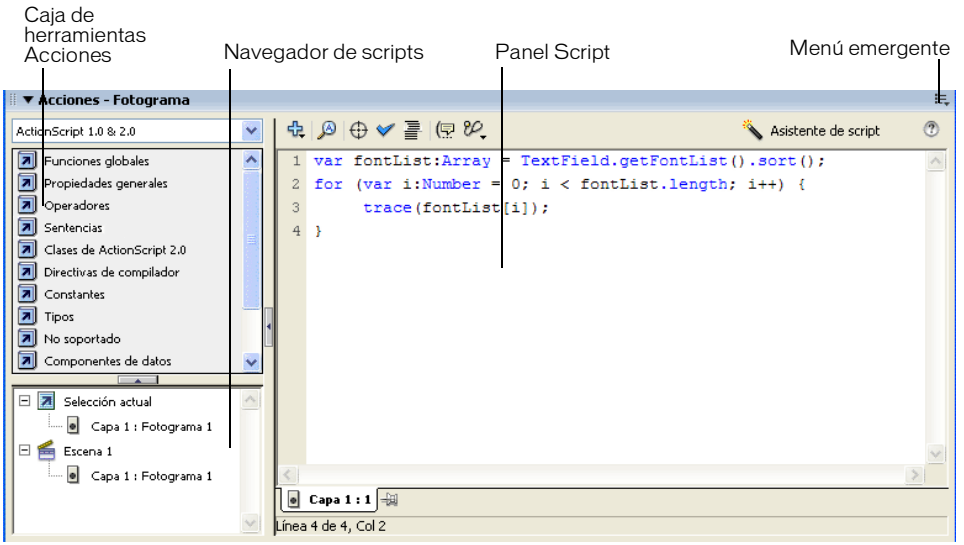
- Para empezar a escribir un nuevo script, seleccione Archivo > Nuevo y elija Archivo ActionScript.
- Para abrir un script existente, seleccione Archivo > Abrir y, a continuación, abra un archivo AS.
- Para editar un script que ya está abierto, haga clic en la ficha de documento que muestra el nombre del script.

Para más información, consulte los temas siguientes:


- "Panel Acciones" en la página 37
- "Ventana Script" en la página 38

# Panel Acciones

El panel Acciones permite crear código ActionScript en un documento de Flash (archivo FLA). Este panel consta de tres paneles, cada uno de los cuales le facilita la creación y gestión de los scripts.




**Caja de herramientas Acciones** Se utiliza para examinar una lista de elementos del lenguaje ActionScript ordenados por categorías (funciones, clases, tipos, etc.) y luego insertarlos en el panel Script. Para insertar un elemento de script en el panel Script, haga doble clic en él o arrástrelo directamente al panel Script. También puede añadir elementos del lenguaje a los scripts mediante el botón Añadir (+) de la barra de herramientas del panel Acciones. Para más información, consulte "Barras de herramientas del panel Acciones y la ventana Script" en la página 41.

**Navegador de scripts** Muestra una lista jerárquica de elementos de Flash (clips de película, fotogramas y botones) que contienen scripts. Utilice el navegador de scripts para desplazarse rápidamente por todos los scripts del documento de Flash.

Si hace clic en un elemento del navegador de scripts, el script asociado con ese elemento aparecerá en el panel Script y la cabeza lectora se desplazará a esa posición en la línea de tiempo. Si hace doble clic en un elemento del navegador de scripts, el script quedará *fijado* (bloqueado en su sitio). Para más información, consulte "Fijación de scripts en el panel Acciones" en la página 63.

**Panel Script** En este panel es donde se escribe el código. El panel Script ofrece herramientas para crear scripts en un editor completo (denominado *editor de ActionScript*) que incluye aplicación de formato y revisión de la sintaxis del código, sugerencias o consejos sobre códigos, colores para el código y otras funciones que simplifican la creación de scripts. Para más información, consulte "Utilización del panel Acciones y la ventana Script" en la página 36.

Para obtener información sobre cada uno de los botones de la barra de herramientas del panel Acciones, consulte "Codificación en el panel Acciones y la ventana Script" en la página 40. Para más información sobre las funciones del panel Acciones, consulte los temas siguientes:



# Ventana Script

Puede escribir y editar código ActionScript en la ventana Script al crear un nuevo archivo ActionScript, Flash Communication o Flash JavaScript. La ventana Script se utiliza para escribir y editar archivos de script externos e incluye color para la sintaxis, sugerencias o consejos sobre códigos y otras opciones del editor.

En la ventana Script, puede crear código ActionScript externo, comunicación de ActionScript y archivos JavaScript de Flash. En función del tipo de archivo de script externo que cree, la caja de herramientas Acciones ofrece la lista completa de elementos del lenguaje disponibles para cada uno de ellos.

Al utilizar la ventana Script, observará que algunas de las funciones de ayuda para el código, como el navegador de scripts, el asistente de script y los comportamientos, no están disponibles. Esto se debe a que estas funciones sólo son útiles en el contexto de creación de un documento de Flash, no al crear un archivo de script externo.

También observará que muchas de las opciones disponibles en el panel Acciones no lo están en la ventana Script. La ventana Script admite las siguientes opciones de editor: la caja de herramientas Acciones, buscar y reemplazar, revisión de sintaxis, aplicación de formato automático, sugerencias o consejos sobre códigos y opciones de depuración (sólo en archivos ActionScript). Además, la ventana Script permite la visualización de números de línea y caracteres ocultos y el ajuste de texto.

### Para ver la ventana Script:

1. Seleccione Archivo > Nuevo.
2. Seleccione el tipo de archivo externo que desee crear (archivo ActionScript, Flash Communication o Flash JavaScript).

Puede tener varios archivos externos abiertos al mismo tiempo; los nombres de archivo se muestran en las fichas situadas en la parte superior de la ventana Script. Para más información sobre las funciones de la ventana Script, consulte los temas siguientes:


- "Barras de herramientas del panel Acciones y la ventana Script" en la página 41
- "Opciones de edición de ActionScript" en la página 43
- "Sugerencias o consejos sobre códigos en Flash" en la página 47
- "Aplicación de formato al código" en la página 53
- "Utilización del resaltado de sintaxis" en la página 55
- "Utilización de números de línea y ajuste de texto" en la página 56
- "Utilización de las teclas de método abreviado de Esc" en la página 56
- "Visualización de caracteres ocultos" en la página 57
- "Utilización de la herramienta Buscar" en la página 58
- "Comprobación de la sintaxis y la puntuación" en la página 59
- "Importación y exportación de scripts" en la página 60

# Codificación en el panel Acciones y la ventana Script

El panel Script, donde se edita el código, es el elemento principal tanto del panel Acciones como de la ventana Script. El panel Acciones y la ventana Script ofrecen funciones básicas de edición de scripts y de ayuda para el código, como sugerencias o consejos sobre códigos, coloreado, aplicación de formato automático, etc.

Se puede acceder a las funciones que ayudan a editar código desde la barra de herramientas del panel Acciones o la ventana Script, a través del sistema de menús y en el propio panel Acciones.

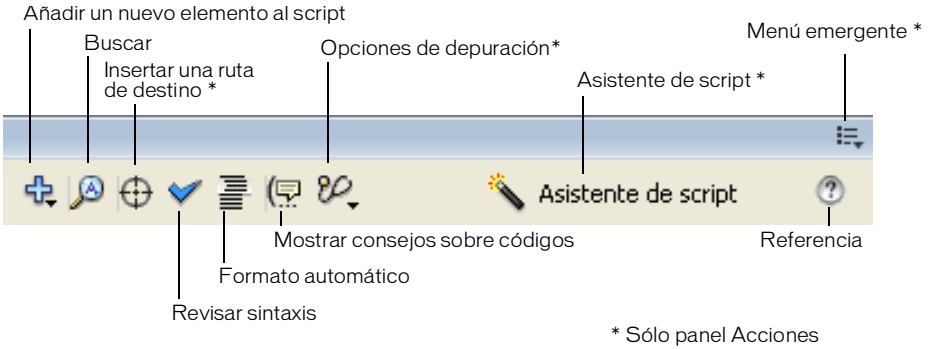
En los siguientes temas se presentan las numerosas funciones que ofrece el editor de ActionScript (panel Acciones y ventana Script):



Para obtener información sólo de las funciones del panel Acciones, como la fijación de script y el navegador de script, consulte "Funciones del panel Acciones" en la página 62.

# Barras de herramientas del panel Acciones y la ventana Script

Las barras de herramientas del panel Acciones y la ventana Script contienen vínculos a las funciones de ayuda para el código que le ayudan a simplificar y agilizar la codificación en ActionScript. Estas barras de herramientas son distintas en función de si utiliza el editor de ActionScript del panel Acciones o el panel Script. En la siguiente imagen se muestran las funciones incluidas en el editor de ActionScript. Las opciones marcadas sólo están disponibles en el panel Acciones.




Las funciones de la barra de herramientas se describen con detalle en "Utilización del panel Acciones y la ventana Script" en la página 36. A continuación, se ofrece un resumen rápido de los botones que encontrará en las barras de herramientas tanto del panel Acciones como de la ventana Script.

NOTA

Algunas de las opciones siguientes se encuentran solamente en el panel Acciones. Estas funciones están marcadas como *Sólo panel Acciones*.

**Añadir un nuevo elemento al script** Muestra todos los elementos del lenguaje que también se encuentran en la caja de herramientas de ActionScript. Al elegir un elemento de la lista de categorías de elementos del lenguaje, éste se añade al script.

**Buscar** Busca y reemplaza texto del código ActionScript. Para más información, consulte "Utilización de la herramienta Buscar" en la página 58.

**Insertar una ruta de destino** *Sólo panel Acciones.* Le ayuda a la hora de establecer una ruta de destino absoluta o relativa para una acción del script. Para más información, consulte "Inserción de rutas de destino" en la página 65.

**Revisar sintaxis** Comprueba los errores de sintaxis existentes en el script actual. Los errores de sintaxis se muestran en el panel Salida. Para más información, consulte "Comprobación de la sintaxis y la puntuación" en la página 59.

**Formato automático** Aplica formato al script para lograr una sintaxis de código correcta y mejorar la legibilidad. Puede establecer preferencias de aplicación de formato automático en el cuadro de diálogo Preferencias, disponible desde el menú Edición o desde el menú emergente del panel Acciones. Para más información, consulte "Aplicación de formato al código" en la página 53.

**Mostrar consejos sobre códigos** Si ha desactivado los consejos automáticos sobre códigos, puede utilizar este comando con el fin de mostrar manualmente un consejo para la línea de código en la que está trabajando. Para más información, consulte "Asistente de script" en la página 62.

**Opciones de depuración** Establece y quita puntos de corte en el script, de forma que, al depurar el documento de Flash, pueda detener y luego avanzar línea a línea por el script. Las opciones de depuración se encuentran ahora disponibles en la ventana Script, así como en el panel Acciones, pero sólo para los archivos ActionScript. Esta opción se desactiva para los archivos de comunicación de ActionScript y Flash JavaScript. Para obtener más información sobre la depuración de los documentos de Flash, consulte "Depuración de los scripts" en la página 754. Para obtener información sobre el establecimiento y eliminación de elementos, consulte "Establecimiento y eliminación de puntos de corte" en la página 763.

**Asistente de script** *Sólo panel Acciones.* El modo de asistente de script le solicita la introducción de los elementos necesarios para crear scripts. Para más información, consulte "Asistente de script" en la página 62.

**Referencia** Muestra un tema de ayuda de referencia para el elemento del lenguaje ActionScript seleccionado en el panel Script. Por ejemplo, si hace clic en una sentencia `import` y luego selecciona Referencia, aparece en el panel Ayuda el tema relativo a `import`.

**Menú emergente** *Sólo panel Acciones.* Contiene los diversos comandos y preferencias que se aplican al panel Acciones o a la ventana Script. Por ejemplo, puede establecer los números de línea y el ajuste de texto en el editor de ActionScript, acceder a las preferencias de ActionScript e importar o exportar scripts. Para más información, consulte "Opciones de edición de ActionScript" en la página 43.

# Opciones de edición de ActionScript

La ventana Script y el panel Acciones ofrecen una gran variedad de funciones de ayuda para el código (herramientas que facilitan la escritura y el mantenimiento de los scripts). Estas opciones están disponibles en la barra de herramientas del panel Acciones o la ventana Script y en el menú emergente de Acciones. Al editar código ActionScript en la ventana Script, estas opciones están disponibles en la barra de herramientas y en el sistema de menús de Flash.

El panel Acciones ofrece más opciones de las que se encuentran disponibles en la ventana Script. Ello se debe a que las opciones adicionales son útiles en el contexto de creación de código ActionScript incorporado en un documento de Flash (no al escribir archivos de ActionScript externos). Para obtener información sobre cuáles de estas opciones están disponibles en la ventana Script, consulte "Ventana Script" en la página 38.

Las opciones que están disponibles en la ventana Script y el panel Acciones se describen en "Barras de herramientas del panel Acciones y la ventana Script" en la página 41.

Las siguientes opciones están disponibles en el menú emergente del panel Acciones y a través de diversos menús de la ventana Script.

NOTA

Algunas de las opciones siguientes se encuentran solamente en el panel Acciones. Estas funciones están marcadas como *Sólo panel Acciones*.

**Volver a cargar sugerencias de código** *Sólo panel Acciones.* Si personaliza el modo de asistente de script mediante la escritura de métodos personalizados, puede volver a cargar sugerencias de código sin reiniciar Flash 8.

**Fijar script** *Sólo panel Acciones.* Fija (bloquea en un lugar) el script actualmente mostrado en el panel Script. Para más información, consulte "Fijación de scripts en el panel Acciones" en la página 63.

**Cerrar script** *Sólo panel Acciones.* Cierra el script actualmente abierto.

**Cerrar todos los script** *Sólo panel Acciones.* Cierra todos los scripts actualmente abiertos.

**Ir a línea** Localiza y resalta la línea especificada del panel Script.

**Buscar y reemplazar** Busca y reemplaza texto dentro de los scripts del panel Script. Para más información, consulte "Utilización de la herramienta Buscar" en la página 58.

**Buscar otra vez** Repite la acción de búsqueda de la última cadena introducida en la herramienta Buscar. Para más información, consulte "Utilización de la herramienta Buscar" en la página 58.

**Importar script** Le permite importar un archivo de script (ActionScript) al panel Script. Para más información, consulte "Preferencias de importación y exportación" en la página 61.

**Exportar script** Exporta el script actual a un archivo ActionScript (AS) externo. Para más información, consulte "Preferencias de importación y exportación" en la página 61.

**Teclas de método abreviado de Esc** Introduce rápidamente elementos y estructuras sintácticas habituales del lenguaje en los scripts. Por ejemplo, al presionar Esc+g+p en el panel Script, se inserta la función `gotoAndPlay()` en el script. Al seleccionar la opción Teclas de método abreviado de Esc del menú emergente del panel Acciones, aparecen todas las teclas de método abreviado de Esc en la caja de herramientas Acciones. Para más información, consulte "Utilización de las teclas de método abreviado de Esc" en la página 56.

**Caracteres ocultos** Permite ver los caracteres ocultos del script. Los caracteres ocultos son espacios, tabuladores y saltos de línea. Para más información, consulte "Visualización de caracteres ocultos" en la página 57.

**Números de línea** Muestra los números de línea en el panel Script. Para más información, consulte "Utilización de números de línea y ajuste de texto" en la página 56.

**Preferencias** *Sólo panel Acciones.* Muestra el cuadro de diálogo de preferencias de ActionScript. Para más información, consulte "Preferencias de ActionScript" en la página 45.

**Ajustar texto** Para ajustar las líneas del script que superan el tamaño actual de la ventana Script, seleccione Ajustar texto del menú emergente del panel Acciones. Si utiliza la ventana Script, seleccione Ajustar texto en el menú Ver. Para más información, consulte "Utilización de números de línea y ajuste de texto" en la página 56.

**Agrupar Acciones con** *Sólo panel Acciones.* Le permite agrupar el panel Acciones (que incluye la caja de herramientas Acciones y el navegador de scripts) con otros paneles dentro del entorno de edición de Flash.

Además, el menú emergente del panel Acciones incluye los comandos Imprimir, Ayuda y de cambio de tamaño del panel.

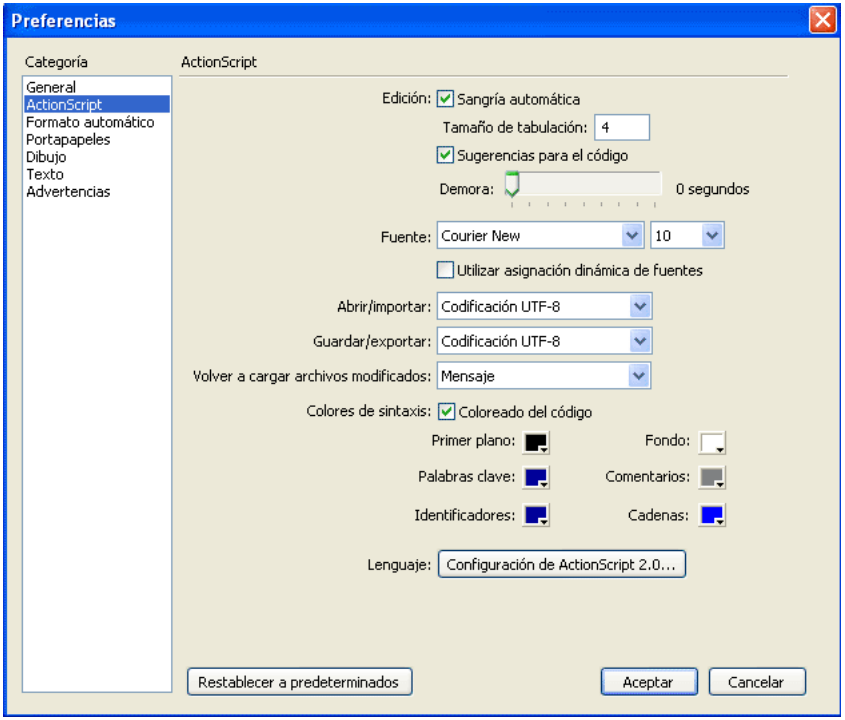
# Preferencias de ActionScript

Tanto si edita código en el panel Acciones como si lo hace en la ventana Script, puede establecer y modificar un único conjunto de preferencias. Por ejemplo, puede controlar la sangría automática, las sugerencias (consejos) y el color del código y diversas funciones básicas de edición de código.

### Para acceder a las preferencias de ActionScript:

1. Para acceder a las preferencias de ActionScript en un archivo FLA con el panel Acciones, seleccione Preferencias en el menú emergente, o bien Edición > Preferencias (Windows) o Flash > Preferencias (Macintosh) y haga clic en ActionScript en la lista Categoría.
2. Para acceder a las preferencias de ActionScript en la ventana Script, seleccione Edición > Preferencias y haga clic en ActionScript (Windows) o Flash > Preferencias y haga clic en ActionScript (Macintosh).

En la siguiente imagen se muestra la configuración de ActionScript que puede cambiar en Flash 8.



Puede establecer las siguientes preferencias:

**Sangría automática** Cuando la sangría automática está activada, el texto que se escribe después del paréntesis de apertura [(] o llave inicial ({) se sangra de forma automática según el valor de Tamaño de tabulación establecido en las preferencias de ActionScript. Para más información, consulte "Aplicación de formato al código" en la página 53.

**Tamaño de tabulación** Especifica el número de caracteres que debe desplazarse una nueva línea cuando está activada la sangría automática.

**Sugerencias para el código** Activa las sugerencias (consejos) para el código en el panel Script. Para más información sobre el uso de sugerencias o consejos de códigos, consulte "Sugerencias o consejos sobre códigos en Flash" en la página 47.

**Demora** Especifica el retardo (en segundos) hasta que se muestran las sugerencias o consejos sobre códigos.

**Fuente** Especifica la fuente utilizada en el panel Script.

**Utilizar asignación dinámica de fuentes** Comprueba que la familia de fuentes seleccionada dispone de los glifos necesarios para representar todos los caracteres. De no ser así, Flash sustituye la familia de fuentes por otra que contenga los caracteres necesarios. Para más información, consulte "Aplicación de formato al código" en la página 53.

**Codificación** Especifica la codificación de caracteres utilizada al abrir, guardar, importar y exportar archivos de ActionScript. Para más información, consulte "Importación y exportación de scripts" en la página 60.

**Volver a cargar archivos modificados** Permite seleccionar cuándo se verán advertencias si se modifica, traslada o elimina un archivo de script. Seleccione Siempre, Nunca o Mensaje.

- **Siempre** No se muestra ninguna advertencia cuando se detecta un cambio y el archivo se vuelve a cargar automáticamente.
- **Nunca** No se muestra ninguna advertencia cuando se detecta un cambio y el archivo permanece en el estado actual.
- **Mensaje** (Predeterminado) Se muestra una advertencia cuando se detecta un cambio y puede elegir entre volver a cargar o no el archivo.

Al crear aplicaciones que incluyen archivos de script externos, esta función ayuda a evitar sobrescribir un script que haya modificado un miembro del equipo desde que abrió la aplicación o publicar la aplicación con versiones anteriores de scripts. Las advertencias permiten cerrar automáticamente un script y volver a abrir la versión modificada más actual.

**Colores de sintaxis** Especifica los colores para el coloreado de código en los scripts. Con el coloreado de código activado, puede seleccionar los colores que se mostrarán en el panel Script.

**Lenguaje** Abre el cuadro de diálogo Configuración de ActionScript. Para más información, consulte "Modificación de la ruta de clases" en la página 68.

# Sugerencias o consejos sobre códigos en Flash

Cuando utilice el panel Acciones o la ventana Script, podrá emplear diversas funciones que le ayudarán a escribir código sintácticamente correcto. Los consejos sobre códigos le ayudan a escribir código de forma rápida y precisa. Los consejos sobre códigos incluyen sugerencias que contienen la sintaxis correcta y menús que permiten seleccionar nombres de métodos y propiedades. En las siguientes secciones se muestra cómo escribir código que utilice estas funciones.



## Activación de las sugerencias para el código

Al trabajar en el panel Acciones o en la ventana Script, Flash puede detectar la acción que está introduciendo y mostrar una *sugerencia para el código*. Los dos estilos diferentes de sugerencias para el código son una sugerencia que contiene la sintaxis completa de la acción en cuestión y un menú emergente en el que se enumeran nombres de los métodos y propiedades posibles (lo que en ocasiones denominados *capacidad para completar código*). Aparece un menú emergente para los parámetros, propiedades y eventos cuando utiliza strict typing o asigna un nombre a los objetos, como se explica en esta sección.

Las sugerencias para el código también aparecen al hacer doble clic en un elemento de la caja de herramientas Acciones o en Añadir (+) situado en la barra de herramientas del panel Acciones o la ventana Script para añadir acciones al panel Script. Para más información sobre cómo utilizar las sugerencias para el código cuando aparecen, consulte "Utilización de las sugerencias para el código" en la página 48.

| NOTA | Las sugerencias para el código están activadas automáticamente para clases nativas que no requieren que se cree y asigne un nombre a una instancia de la clase, como Math, Key, Mouse, etc. |
|---|---|

Para asegurarse de que estén activados las sugerencias para el código, la opción Sugerencias para el código debe estar seleccionada en el cuadro de diálogo Preferencias de ActionScript. Para más información, consulte "Panel Acciones" en la página 37.

## Utilización de las sugerencias para el código

Las sugerencias para el código están desactivadas de forma predeterminada. Al definir las preferencias, puede desactivar las sugerencias para el código o determinar la rapidez con la que aparecen. Aunque las sugerencias para el código estén desactivadas en las preferencias, puede ver una sugerencia para un comando específico.

### Para especificar la configuración de las sugerencias para el código automáticas, realice una de las acciones siguientes:

- En el panel Acciones o en la ventana Script, seleccione Edición > Preferencias (Windows) o Flash > Preferencias (Macintosh), haga clic en ActionScript de la lista Categoría y active o desactive Sugerencias para el código.
- En el panel Acciones, seleccione Preferencias en el menú emergente (en la parte superior derecha del panel Acciones) y active o desactive Sugerencias para el código en las preferencias de ActionScript.

Si activa las sugerencias para el código, también puede especificar los segundos que deben transcurrir antes de que aparezcan dichas sugerencias. Por ejemplo, si no tiene experiencia con ActionScript, seguramente preferirá que no haya ninguna demora, ya que así las sugerencias para el código aparecen inmediatamente. No obstante, si suele saber lo que desea escribir y tan sólo necesita sugerencias cuando utiliza elementos de lenguaje con los que no está familiarizado, puede especificar una demora para que las sugerencias para el código no aparezcan cuando no desee utilizarlas.

### Para especificar una demora en las sugerencias para el código:

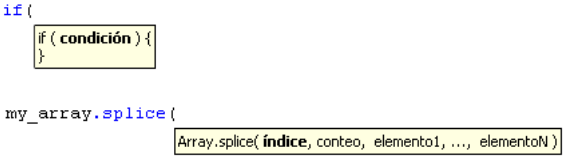
1. En el panel Acciones o en la ventana Script, seleccione Edición > Preferencias (Windows) o Flash > Preferencias (Macintosh) desde el menú principal.
2. Haga clic en ActionScript de la lista Categoría.
3. Utilice el deslizador para seleccionar una cantidad de retraso.

   Esta cantidad es en segundos.

### Para trabajar con sugerencias para el código con formato de información sobre herramientas:

1. Visualice las sugerencias para el código escribiendo un paréntesis de apertura `[(]` después de un elemento que requiere paréntesis (por ejemplo, después de un nombre de método, un comando del tipo `if` o `do..while`, etc.).

Aparece la sugerencia para el código.



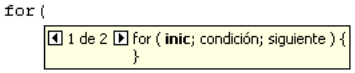

**NOTA**

Si no aparece la sugerencia para el código, asegúrese de que no ha desactivado Sugerencias para el código en las preferencias de ActionScript (Edición › Preferencias (Windows) o Flash › Preferencias (Macintosh) y haga clic en ActionScript de la lista Categoría). Si desea visualizar sugerencias para el código para una variable u objeto que ha creado, asegúrese de que ha asignado un nombre correcto a la variable o al objeto (consulte "Utilización de sufijos para activar las sugerencias para el código" en la página 51), o que utiliza strict typing para la variable u objeto (consulte "Strict typing de objetos para activar las sugerencias para el código" en la página 51).

2. Introduzca un valor para el parámetro.

   Si hay presente más de un parámetro, separe los valores con comas. En el caso de funciones o sentencias, como el bucle `for`, separe los parámetros con punto y coma.

   Los comandos sobrecargados (funciones o métodos que pueden invocarse con diferentes conjuntos de parámetros), como `gotoAndPlay()` o `for`, muestran un indicador que permite seleccionar el parámetro que desea establecer. Haga clic en los botones de flecha o presione Ctrl+Flecha izquierda o Ctrl+Flecha derecha para seleccionar el parámetro.

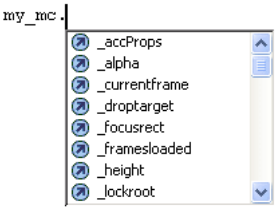



3. Para cerrar la sugerencia para el código, utilice uno de los procedimientos siguientes:
   - Escriba un paréntesis de cierre [`)`].
   - Haga clic fuera de la sentencia.
   - Presione Esc.

### Para trabajar con las sugerencias para el código con formato de menú:

1. Visualice la sugerencia para el código escribiendo un punto después del nombre de objeto o variable.

   Aparece el menú de sugerencias para el código.




| NOTA | Si no aparece la sugerencia para el código, asegúrese de que no ha desactivado Sugerencias para el código en las preferencias de ActionScript (Edición › Preferencias (Windows) o Flash › Preferencias (Macintosh) y haga clic en ActionScript de la lista Categoría). Si desea visualizar sugerencias para el código para una variable u objeto que ha creado, asegúrese de que ha asignado un nombre correcto a la variable o al objeto (consulte "Utilización de sufijos para activar las sugerencias para el código" en la página 51), o que utiliza strict typing para la variable u objeto (consulte "Strict typing de objetos para activar las sugerencias para el código" en la página 51). |
|---|---|

2. Para desplazarse por las sugerencias para el código, utilice las teclas de flecha arriba y flecha abajo.
3. Para seleccionar un elemento del menú, presione Intro o el tabulador, o haga doble clic en el elemento.
4. Para cerrar la sugerencia para el código, utilice uno de los procedimientos siguientes:
   - Seleccione uno de los elementos del menú.
   - Haga clic por encima o por debajo de la ventana de menú.
   - Escriba un paréntesis de cierre [)] si ya ha utilizado un paréntesis de apertura [(].
   - Presione Esc.

### Para mostrar manualmente una sugerencia para el código:

1. Haga clic en una ubicación de código en la que puedan aparecer sugerencias para el código, como en las siguientes ubicaciones:
   - Después del punto (.) que aparece a continuación de una sentencia o comando, donde debe especificarse una propiedad o un método
   - Entre paréntesis [()] en un nombre de método

2. Siga uno de estos procedimientos:

- Haga clic en Mostrar consejos sobre códigos en la barra de herramientas del panel Acciones o ventana Script.
- Presione Ctrl+barra espaciadora (Windows) o Comando+barra espaciadora (Macintosh).
- Si está utilizando el panel Acciones, seleccione Mostrar consejos sobre códigos en el menú emergente.

## Strict typing de objetos para activar las sugerencias para el código

Cuando se utiliza ActionScript 2.0, se puede utilizar strict typing para una variable basada en una clase incorporada, como Botón, Array, etc. De este modo, el panel Script muestra las sugerencias para el código de la variable. Por ejemplo, suponga que escribe el código siguiente:

```
var names:Array = new Array();
names.
```

En cuanto escriba el punto (.), Flash mostrará una lista de métodos y propiedades disponibles para los objetos Array en un menú emergente, ya que ha definido la variable como una matriz. Para más información sobre la escritura de datos, consulte “Asignación de tipos de datos y “strict data typing”” en la página 337. Para más información sobre cómo utilizar las sugerencias para el código cuando aparecen, consulte “Utilización de las sugerencias para el código” en la página 48.

## Utilización de sufijos para activar las sugerencias para el código

Si utiliza ActionScript 1 o desea visualizar las sugerencias para el código de los objetos que crea sin strict typing (consulte “Strict typing de objetos para activar las sugerencias para el código” en la página 51), debe añadir un sufijo especial al nombre de cada objeto cuando lo cree. Por ejemplo, los sufijos que activan las sugerencias para el código para la clase Array y la clase Camera son `_array` y `_cam`, respectivamente. Por ejemplo, si escribe el código siguiente:

```
var my_array = new Array();
var my_cam = Camera.get();
```

puede introducir cualquiera de las siguientes posibilidades (el nombre de la variable seguido de un punto):

```
my_array.
my_cam.
```

Aparecerán sugerencias para el código de los objetos Array y Camera.

Para los objetos que aparecen en el escenario, utilice el sufijo del cuadro de texto Nombre de instancia del inspector de propiedades. Por ejemplo, para ver las sugerencias para el código de objetos MovieClip, utilice el inspector de propiedades para asignar nombres de instancia con el sufijo `_mc` a todos los objetos MovieClip. Así, siempre que escriba el nombre de instancia seguido de un punto, se mostrarán las sugerencias para el código.

Aunque los sufijos no son necesarios para activar las sugerencias para el código cuando se utiliza el strict typing de un objeto, su utilización ayuda a comprender los scripts.

En la tabla siguiente se enumeran los sufijos necesarios para que se presenten sugerencias para el código automáticamente.

| Tipo de objeto | Sufijo de la variable |
|---|---|
| Array | _array |
| Button | _btn |
| Camera | _cam |
| Color | _color |
| ContextMenu | _cm |
| ContextMenuItem | _cmi |
| Date | _date |
| Error | _err |
| LoadVars | _lv |
| LocalConnection | _lc |
| Microphone | _mic |
| MovieClip | _mc |
| MovieClipLoader | _mcl |
| PrintJob | _pj |
| NetConnection | _nc |
| NetStream | _ns |
| SharedObject | _so |
| Sound | _sound |
| String | _str |
| TextField | _txt |
| TextFormat | _fmt |
| Video | _video |

| Tipo de objeto | Sufijo de la variable |
|---|---|
| XML | _xml |
| XMLNode | _xmlnode |
| XMLSocket | _xmlsocket |

Para más información sobre cómo utilizar las sugerencias para el código cuando aparecen, consulte "Utilización de las sugerencias para el código" en la página 48.

## Utilización de comentarios para activar las sugerencias para el código

También puede utilizar comentarios de ActionScript para especificar una clase de objeto para las sugerencias para el código. En el ejemplo siguiente se indica a ActionScript que la clase de la instancia `theObject` es Object y así sucesivamente. Si especificara `mc` seguido de un punto después de estos comentarios, aparecerían las sugerencias para el código que muestran la lista de métodos y propiedades de MovieClip. Si introdujese `theArray` seguido de un punto, aparecería un menú con una lista de los métodos y propiedades de Array, y así sucesivamente.

```
// Object theObject;
// Array theArray;
// MovieClip theMc;
```

No obstante, Macromedia recomienda que, en lugar de esta técnica, se utilice "strict data typing" (consulte "Strict typing de objetos para activar las sugerencias para el código" en la página 51) o sufijos (consulte "Utilización de sufijos para activar las sugerencias para el código" en la página 51), ya que dichas técnicas activan automáticamente las sugerencias para el código y hacen que el código sea más comprensible. Para más información sobre el uso de sugerencias para el código, consulte "Utilización de las sugerencias para el código" en la página 48.

# Aplicación de formato al código

Puede especificar una configuración para determinar si se debe aplicar formato y sangrías automática o manualmente al código. Además, puede seleccionar si debe utilizarse asignación dinámica de fuentes, lo que garantiza que se utilicen las fuentes correctas cuando se trabaja con textos multilingües.

### Para establecer las opciones de formato:

1. En el panel Acciones, seleccione Preferencias del menú emergente (en la parte superior derecha del panel Acciones). En el cuadro de diálogo Preferencias, seleccione Formato automático.

De forma alternativa, en la ventana Script, seleccione Edición > Preferencias (Windows) o Flash > Preferencias (Macintosh). En el cuadro de diálogo Preferencias, seleccione Formato automático.

2. Seleccione cualquiera de las opciones de Formato automático.

   Para ver el efecto de cada selección, examine el panel Vista previa.

Una vez establecidas las opciones de Formato automático, la configuración se aplicará automáticamente al código que escriba, pero no al código que ya exista; deberá aplicar manualmente la configuración al código ya existente. Debe aplicar formato manualmente a un código al que se aplicó formato con una configuración diferente, un código importado de otro editor, etc.

### Para aplicar formato al código según la configuración de Formato automático, realice una de las acciones siguientes:

- Haga clic en Formato automático del panel Acciones o en la barra de herramientas de la ventana Script.
- En el panel Acciones, seleccione Formato automático en el menú emergente.
- Presione Ctrl+Mayús+F (Windows) o Comando+Mayúsculas+F (Macintosh).
- En la ventana Script, seleccione Herramientas > Formato automático.

### Para utilizar la asignación dinámica de fuentes:

- Para activar o desactivar la asignación dinámica de fuentes, seleccione o anule la selección de Utilizar asignación dinámica de fuentes en el cuadro de diálogo Preferencias.

  La asignación dinámica de fuentes se encuentra desactivada de forma predeterminada, ya que así aumenta el tiempo de rendimiento al crear scripts. Si está trabajando con textos multilingües, deberá activar la asignación dinámica de fuentes, ya que de este modo, se asegura de que se utilizan las fuentes correctas.

### Para utilizar una sangría automática:

- Para activar y desactivar la sangría automática, seleccione o anule la selección de Sangría automática en el cuadro de diálogo Preferencias.

  Cuando la sangría automática está activada, el texto que se escribe después del paréntesis de apertura [(] o llave inicial ({) se sangra de forma automática según el valor de Tamaño de tabulación establecido en las preferencias de ActionScript.

  En los scripts, puede aplicar sangría a una línea seleccionándola y presionando la tecla Tabulador. Para quitar la sangría, seleccione la línea y presione Mayús+Tab.

## Utilización del resaltado de sintaxis

En ActionScript, como en cualquier lenguaje, la *sintaxis* es la manera en la que los elementos se agrupan para crear un significado. Si utiliza una sintaxis de ActionScript incorrecta, los scripts no funcionarán.

Cuando escribe scripts en Flash Basic 8 y Flash Professional 8, los comandos no admitidos por la versión del reproductor que esté utilizando se muestran en amarillo en la caja de herramientas Acciones. Por ejemplo, si la versión del archivo SWF de Flash Player está establecida en Flash 7, el código de ActionScript que sólo se admite en Flash Player 8 aparece en color amarillo en la caja de herramientas Acciones. (Para obtener información sobre la configuración de la versión de archivos SWF de Flash Player, consulte Capítulo 17, "Establecimiento de las opciones de publicación para el formato de archivo SWF de Flash" en *Utilización de Flash*.)

También puede definir una preferencia para que Flash codifique por colores las distintas partes de los scripts a medida que los escribe, con el fin de resaltar los errores de escritura. Por ejemplo, supongamos que establece la preferencia de coloreado de la sintaxis de tal modo que las palabras clave se muestren en color azul oscuro. Mientras escribe código, si escribe `var`, la palabra `var` aparecerá en azul. No obstante si, por error, escribe `vae`, la palabra `vae` permanecerá de color negro, con lo cual usted sabrá inmediatamente que la ha escrito incorrectamente. Para obtener información sobre las palabras clave, consulte "Palabras clave" en la página 103.

**Para definir las preferencias para el coloreado de la sintaxis a medida que escribe, realice una de las acciones siguientes:**

- Seleccione Edición > Preferencias (Windows) o Flash > Preferencias (Macintosh), haga clic en ActionScript de la lista Categoría y especifique la configuración de Colores de sintaxis.
- En el panel Acciones, seleccione Preferencias en el menú emergente (parte superior derecha del panel Acciones) y especifique la configuración de Colores de sintaxis en las preferencias de ActionScript.
- Con el puntero del ratón en el panel Script, presione Control-U (Windows) o Comando-U (Macintosh).

Puede cambiar la configuración de color de las palabras clave, los comentarios, los identificadores y las cadenas. Para obtener información sobre identificadores y cadenas, consulte "Terminología" en la página 851 y "Tipo de datos String (cadena)" en la página 336. Para obtener información sobre conversión en comentario, consulte "Comentarios" en la página 95.

## Utilización de números de línea y ajuste de texto

Puede especificar si se deben ver los números de línea y si se debe ajustar el texto de las líneas largas del código. Normalmente, se deben activar los números de línea y el ajuste de texto para facilitar la edición de código. Los números de línea facilitan el desplazamiento y el análisis del código mientras lo edita o modifica. El ajuste de texto ayuda a evitar tener que desplazarse horizontalmente por largas líneas de código (especialmente cuando trabaja en el entorno de edición o con resoluciones de pantalla bajas).

**Para activar o desactivar los números de línea, siga uno de estos procedimientos:**

- En el panel Acciones, seleccione Números de línea en el menú emergente.
- En la ventana Script, seleccione Herramientas > Números de línea.
- Presione Ctrl+Mayús+L (Windows) o Comando+Mayúsculas+L (Macintosh).

**Para activar o desactivar el ajuste de texto, siga uno de estos procedimientos:**

- En el panel Acciones, seleccione Ajustar texto en el menú emergente.
- En la ventana Script, seleccione Herramientas > Ajustar texto.
- Presione las teclas Ctrl+Mayús+W (Windows) o Comando+Mayúsculas+W (Macintosh).

## Utilización de las teclas de método abreviado de Esc

Puede añadir muchos elementos a un script mediante las teclas de método abreviado Esc (presionando la tecla Esc y, a continuación, dos teclas más).

| NOTA | Estos métodos abreviados son diferentes de los métodos abreviados de teclado que activan determinados comandos de menú. |
|---|---|

Por ejemplo, si trabaja en el panel Script y presiona Esc+d+o, se añade el siguiente código al script:

```
do {
} while ();
```

El punto de inserción se coloca justo después de la palabra `while`, donde puede comenzar a escribir su condición. Del mismo modo, si presiona Esc+c+h, se añade el siguiente código al script y el punto de inserción se coloca entre los paréntesis [`()`], donde puede empezar a escribir la condición:

```
catch () {
}
```

Para saber (o recordar) qué comandos tienen teclas de método abreviado de Esc, puede mostrarlos junto a elementos de la caja de herramientas del panel Acciones.


**Para mostrar u ocultar las teclas de método abreviado de Esc:**

- En el menú emergente del panel Acciones, seleccione o anule la selección de Teclas de método abreviado de Esc.

  Las teclas de método abreviado de Esc aparecen junto a los elementos de la caja de herramientas de ActionScript.

# Visualización de caracteres ocultos

Al escribir y aplicar formato al código ActionScript, introducirá espacios, tabuladores y saltos de línea en el script. Obviamente, estos son útiles y necesarios para organizar el código visualmente. Sin embargo, el compilador de Flash genera errores si encuentra espacios de doble byte que no forman parte de un valor de cadena. Al mostrar los caracteres ocultos en el panel Script, podrá ver y quitar los espacios de doble byte.

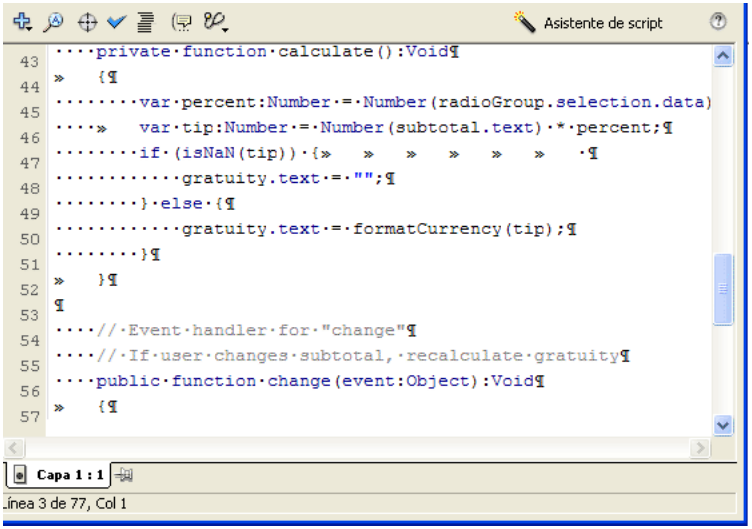
Los siguientes símbolos se utilizan para mostrar cada uno de los caracteres ocultos:

| | |
|---|---|
| espacio de un solo byte | · |
| espacio de doble byte | ˚ |
| tabulador | » |
| salto de línea | ¶ |

**Para mostrar caracteres ocultos, siga uno de estos procedimientos:**

- Seleccione Caracteres ocultos en el menú emergente.
- Presione Ctrl+Mayús+8 (Windows) o Comando+Mayúsculas+8 (Macintosh).

Cuando se muestran los caracteres ocultos, el panel Script presenta el siguiente aspecto:




# Utilización de la herramienta Buscar

La herramienta Buscar le permite localizar y, opcionalmente, reemplazar cadenas de texto de los scripts. Puede reemplazar el texto la primera o todas las veces que aparezca en el script. También puede hacer que coincidan las mayúsculas y minúsculas del texto.

**Para buscar texto en un script:**

1. En la barra de herramientas del panel Acciones o la ventana Script, seleccione la herramienta Buscar o presione Ctrl+F (Windows) o Comando+F (Macintosh).
2. Introduzca la cadena que desea buscar en el script.
3. Haga clic en Buscar siguiente.

   Si el texto o los caracteres aparecen en el script, se resaltarán las palabras o caracteres en el panel Script.

**Para buscar y reemplazar texto en un script:**

1. En la barra de herramientas del panel Acciones o la ventana Script, haga clic en la herramienta Buscar o presione Ctrl+F (Windows) o Comando+F (Macintosh).
2. Introduzca la cadena que desea localizar y reemplazar en el script.
3. En el cuadro de texto Reemplazar, introduzca la nueva cadena.
4. Haga clic en Buscar siguiente.

   Si la cadena se encuentra en el script, se resalta.
5. Haga clic en Reemplazar para reemplazar la cadena o en Reemplazar todo para reemplazar la cadena en todos los lugares en los que aparezca.

Una vez que haya introducido una cadena de búsqueda en la herramienta Buscar, podrá repetir la búsqueda si selecciona Buscar otra vez en el menú emergente.

## Comprobación de la sintaxis y la puntuación

Para determinar si el código que ha escrito responde como se esperaba, es necesario publicar o probar el archivo. Sin embargo, puede realizar una comprobación rápida del código ActionScript sin salir del archivo FLA. Los errores de sintaxis se muestran en el panel Salida. También puede comprobar si los paréntesis, las llaves o los corchetes de un bloque de código están emparejados.

Cuando se revisa la sintaxis, se comprueba el script actual. Si el script actual llama a clases de ActionScript 2.0, dichas clases se compilan y su sintaxis también se revisa. Los demás scripts que puedan encontrarse en el archivo FLA no se revisan.

**Para revisar la sintaxis, utilice uno de los procedimientos siguientes:**

- Haga clic en Revisar sintaxis en la barra de herramientas del panel Acciones o la ventana Script.
- En el panel Acciones, seleccione Revisar sintaxis en el menú emergente.
- Seleccione el panel Script (para que se resalte) y presione Ctrl+T (Windows) o Comando+T (Macintosh).

| NOTA | Si hace clic en Revisar sintaxis en un archivo de clase ActionScript 2.0 externo en la ventana Script, la ruta de clases global afecta a este proceso. Algunas veces generará errores (aunque la ruta de clases global esté correctamente configurada) porque el compilador no tiene constancia de que se esté compilando esta clase. Para más información sobre la compilación de clases, consulte "Compilación y exportación de clases" en la página 253. |
|---|---|

**Para revisar si los signos de puntuación están emparejados, siga uno de estos procedimientos:**

- Haga clic entre las llaves ({}), los corchetes ([]) o paréntesis [()] del script.
- Para Windows, presione Ctrl+' (comilla simple) o para Macintosh, presione Comando+' (comilla simple) para resaltar el texto que se encuentra entre las llaves, los corchetes o los paréntesis.

  El resaltado ayuda a comprobar si la puntuación de apertura dispone de su correspondiente puntuación de cierre.

## Importación y exportación de scripts

Puede importar un script al panel Acciones o a la ventana Script y exportar sus scripts a archivos ActionScript externos. Ambas operaciones pueden ser útiles para compartir código entre diversas aplicaciones de Flash y equipos de desarrollo.

**Para importar un archivo AS externo:**

- Para importar un script externo a un script en el que está trabajando en el panel Script, sitúe el punto de inserción en el lugar en el que desea que aparezca la primera línea del script externo y luego realice una de las siguientes operaciones:
  - En el panel Acciones, seleccione Importar script del menú emergente o presione Ctrl+Mayús+I (Windows) o Comando+Mayús+I (Macintosh).
  - En la ventana Script, seleccione Importar script del menú Archivo o presione Ctrl+Mayús+I (Windows) o Comando+Mayús+I (Macintosh).

Puede exportar un script desde el panel Acciones. Cuando esté utilizando la ventana Script, la exportación es innecesaria, ya que puede guardar el archivo AS.

**Para exportar un script desde el panel Acciones:**

1. Seleccione el script que va a exportar y luego haga clic en Exportar script del menú emergente o presione Ctrl+Mayús+X (Windows) o Comando+Mayús+X (Macintosh).

   Aparecerá el cuadro de diálogo Guardar como.
2. Guarde el archivo de ActionScript (AS).

Flash admite una serie de formatos de codificación de caracteres (incluido Unicode), por lo que puede especificar el formato que debe utilizarse al importar y exportar scripts. Para más información, consulte "Importación y exportación de scripts" en la página 60 y "Preferencias de importación y exportación" en la página 61.

## Compatibilidad de ActionScript con Unicode

Flash 8 es compatible con la codificación de texto Unicode para ActionScript. Ello significa que se puede incluir texto en diferentes idiomas en un archivo de ActionScript. Por ejemplo, puede incluir texto en inglés, japonés y francés en un mismo archivo.

| ATENCIÓN | Si utiliza una aplicación en lengua no inglesa en un sistema en lengua inglesa, el comando Probar película (consulte "Depuración de los scripts" en la página 754) dará un error si una parte cualquiera de la ruta del archivo SWF incluye caracteres que no pueden representarse mediante el esquema de codificación MBCS (Juegos de caracteres multibyte, Multibyte Character Sets). Por ejemplo, las rutas en japonés, que funcionan en un sistema en lengua japonesa, no funcionan en un sistema en inglés. Esta limitación se aplica a todas las áreas de la aplicación que utilizan el reproductor externo. |
|---|---|

## Preferencias de importación y exportación

Puede definir en las preferencias de ActionScript el tipo de codificación que se debe utilizar al importar o exportar archivos de ActionScript. Puede seleccionar la codificación UTF-8 o la codificación predeterminada. UTF-8 es un formato Unicode de 8 bits, mientras que la codificación predeterminada es el formato de codificación que admite el idioma del sistema que esté utilizando, que también se conoce como *página de códigos tradicional*.

Normalmente, si se importan o exportan archivos de ActionScript en formato UTF-8, se utiliza la preferencia UTF-8. Si se importan o exportan archivos con la página de códigos tradicional que utiliza el sistema, se utiliza la preferencia Codificación predeterminada.

Si el texto de los scripts no tiene el aspecto esperado al abrir o al importar un archivo, cambie la preferencia de codificación de importación. Si recibe un mensaje de advertencia al exportar archivos de ActionScript, puede cambiar la preferencia de codificación de exportación o desactivar este mensaje en las preferencias de ActionScript.

### Para seleccionar las opciones de codificación de texto para importar o exportar archivos de ActionScript:

1. En el cuadro de diálogo Preferencias (Edición > Preferencias (Windows) o Flash > Preferencias (Macintosh)), haga clic en ActionScript de la lista Categoría.
2. En Opciones de edición, realice una o las dos acciones siguientes:
   - En Abrir/importar, seleccione Codificación UTF-8 para abrir o importar con la codificación Unicode o bien Codificación predeterminada para abrir o importar con la codificación del idioma que utilice el sistema.
   - En Guardar/exportar, seleccione Codificación UTF-8 para guardar o exportar con la codificación Unicode, o bien Codificación predeterminada para guardar o exportar con la codificación del idioma que utilice el sistema.

**Para activar o desactivar el mensaje de advertencia de la codificación de exportación:**

1. En el cuadro de diálogo Preferencias, seleccione Edición > Preferencias (Windows) o Flash > Preferencias (Macintosh) y haga clic en Advertencias de la lista Categoría.
2. Seleccione o anule la selección de Avisar acerca de conflictos de codificación al exportar archivos ActionScript.

# Funciones del panel Acciones

Las funciones siguientes sólo están disponibles en el panel Acciones, no lo están en la ventana Script. Aunque el panel Acciones cuenta con todas las funciones de la ventana Script, esta última se utiliza con una funcionalidad distinta. El panel Acciones debe admitir parte de la funcionalidad relacionada con los archivos FLA, que conocerá en las secciones siguientes. Para ver las funciones disponibles tanto en la ventana Script como en el panel Acciones, consulte las secciones de "Codificación en el panel Acciones y la ventana Script" en la página 40.

Para ver las funciones disponibles sólo en el panel Acciones, consulte estas secciones:

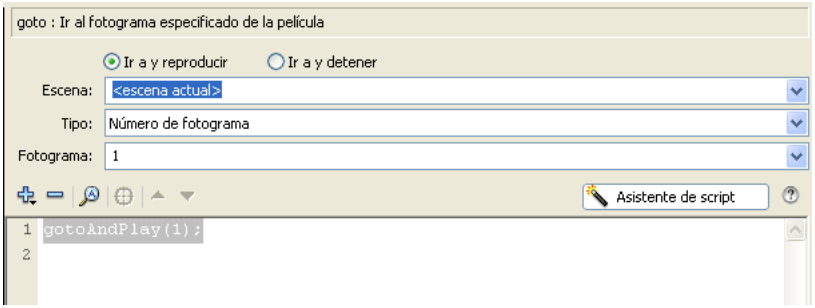
- "Asistente de script" en la página 62
- "Fijación de scripts en el panel Acciones" en la página 63
- "Inserción de rutas de destino" en la página 65

## Asistente de script

El Asistente de script le solicita la introducción de los elementos de un script, lo que le ayuda a añadir interactividad sencilla al archivo SWF o aplicación de Flash con mayor facilidad. El modo de asistente de script es idóneo para usuarios que no tengan soltura escribiendo scripts o que simplemente prefieran la comodidad que ofrece esta herramienta.

Al utilizarlo conjuntamente con el panel Acciones, el Asistente de script le pide que seleccione opciones y que introduzca parámetros. Por ejemplo, en lugar de escribir un nuevo script, puede seleccionar un elemento del lenguaje de la caja de herramientas de Acciones (o el comando Añadir (+) de la barra de herramientas), arrastrarlo al panel Script y, seguidamente, utilizar el Asistente de script para que le ayude a completar el script.

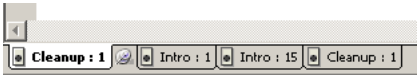
En el siguiente ejemplo, se ha añadido la función `gotoAndPlay` al panel Script. El Asistente de script muestra todos los datos necesarios para utilizar esta función de ActionScript; en este caso, el nombre de la escena, el tipo y el número de fotograma.




## Fijación de scripts en el panel Acciones

Si no concentra todo el código en un archivo FLA en una única ubicación (descrito en "Organización de código ActionScript" en la página 34) o si utiliza comportamientos (consulte "Comportamientos" en la página 65), puede *fijar* varios scripts del panel Acciones para que sea más fácil moverse por ellos. Fijar un script significa poder mantener la ubicación del código abierta en el panel Acciones y hacer clic fácilmente entre los scripts abiertos.

En la figura siguiente, el script asociado con la ubicación actual de la línea de tiempo se encuentra en el fotograma 1 de la capa llamada Simplificar. La ficha del extremo izquierdo siempre indica la ubicación a lo largo de la línea de tiempo. Ese script también se fija (se muestra como la ficha situada más a la derecha). Se fijan otros dos scripts: uno en el fotograma 1 y otro en el fotograma 15 de la capa llamada Intro. Puede desplazarse entre los scripts fijados haciendo clic en las fichas o utilizando métodos abreviados de teclado, como Control+Mayús+. (punto). Desplazarse entre scripts fijados no cambia la posición actual en la línea de tiempo. Como puede ver en la siguiente figura, se abren varios scripts en el panel Acciones y puede hacer clic en cada ficha para desplazarse entre los scripts.




**SUGERENCIA**

Si el contenido que muestra el panel Script no cambia de acuerdo con la ubicación que seleccione en la línea de tiempo, probablemente el panel Script está mostrando un script fijado. Haga clic en la ficha de la izquierda en la parte inferior izquierda del panel Script para mostrar el ActionScript asociado a su ubicación a lo largo de la línea de tiempo.

### Para fijar un script:

1. Coloque el puntero del ratón en la línea de tiempo de manera que el script aparezca en una ficha en la parte inferior izquierda del panel Script del panel Acciones.
2. Siga uno de estos procedimientos:
   - Haga clic en el icono de la chincheta situado a la derecha de la ficha.
   - Haga clic con el botón derecho del ratón (Windows) o haga clic con la tecla Control presionada (Macintosh) en la ficha y seleccione Fijar script.
   - Seleccione Fijar script en el menú emergente (en la parte superior derecha del panel Acciones).
   - Con el puntero del ratón en el panel Script, presione Ctrl+= (signo igual) en Windows o Comando+= en Macintosh.

### Para liberar uno o varios scripts, siga uno de estos procedimientos:

- Si aparece un script fijado en una ficha en la parte inferior izquierda del panel Script del panel Acciones, haga clic en el icono de la chincheta situado a la derecha de la ficha.
- Haga clic con el botón derecho del ratón (Windows) o haga clic con la tecla Control presionada (Macintosh) en una ficha y seleccione Cerrar script o Cerrar todos los script.
- Seleccione Cerrar script o Cerrar todos los script en el menú emergente (en la parte superior derecha del panel Acciones).
- Con el puntero del ratón en el panel Script, presione Ctrl+- (signo menos) en Windows o Comando+- en Macintosh.

### Para utilizar los métodos abreviados de teclado con scripts fijados:

- Puede utilizar los siguientes métodos abreviados de teclado para trabajar con scripts fijados:

| Acción | Tecla de método abreviado de Windows | Tecla de método abreviado de Macintosh |
|---|---|---|
| Fijar script | Ctrl+= (signo igual) | Comando+= |
| Liberar script | Ctrl+- (signo menos) | Comando+- |
| Desplazar selección a la ficha de la derecha | Control+Mayús+. (punto) | Comando+Mayús+. |
| Desplazar selección a la ficha de la izquierda | Control+Mayús+, (coma) | Comando+Mayús+, |
| Liberar todos los scripts | Control+Mayús+- (menos) | Comando+Mayús+- |

## Inserción de rutas de destino

Muchas de las acciones que se crean en el script afectan a clips de película, botones y otras instancias de símbolos. Para aplicar acciones a instancias de una línea de tiempo, establezca la *ruta de destino* (la dirección de la instancia a la que desea hacer referencia). Puede establecer una ruta de destino absoluta o relativa.

La herramienta Insertar una ruta de destino, disponible en el panel Acciones, le solicita la introducción de la ruta de destino para la acción seleccionada en el script.

**Para insertar una ruta de destino:**

1. Seleccione y sitúe el puntero en una acción del script.
2. Haga clic en Insertar una ruta de destino en la barra de herramientas del panel Acciones.

   Aparecerá el cuadro de diálogo Insertar ruta de destino.
3. Siga uno de estos procedimientos:
   - Introduzca manualmente la ruta de la instancia de destino.
   - Seleccione el destino de la lista de destinos disponibles.
4. Seleccione la opción de ruta Relativo o Absoluto.
5. Haga clic en Aceptar.

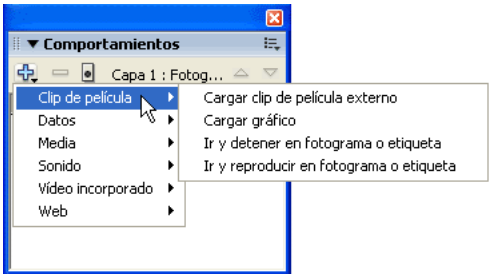
   La ruta se añade a la acción.

# Comportamientos

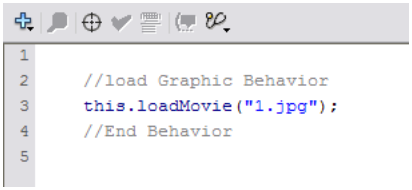
Los comportamientos son funciones predefinidas de ActionScript que puede asociar a los objetos del documento de Flash y que evitan que tenga que crear el código ActionScript manualmente. Los comportamientos ofrecen funcionalidad de ActionScript escrita previamente, como la navegación entre fotogramas, la carga de archivos SWF y JPEG externos, el control del orden de apilamiento de los clips de película y la función de arrastre de clips de película.

Puede utilizar los comportamientos para que la creación de una aplicación de Flash resulte más sencilla, como forma de evitar tener que escribir código ActionScript o, por el contrario, aprender cómo funciona el código ActionScript en determinadas situaciones.

Los comportamientos sólo están disponibles cuando se trabaja con un documento de Flash, no con un archivo de script externo. Normalmente, se selecciona un objeto desencadenante en el documento, un clip de película o un botón, se selecciona Añadir en el panel Comportamientos y, seguidamente, se elige el comportamiento deseado, como se muestra en el siguiente ejemplo:




El comportamiento se añade al objeto y se muestra en el panel Acciones.




# Configuración de publicación de ActionScript

Puede editar ActionScript de dos maneras. Se puede editar ActionScript incorporado en un documento de Flash mediante el panel Acciones, o bien editar ActionScript existente en un archivo de script independiente, externo al documento de Flash, mediante la ventana Script. Dado que el panel Acciones y la ventana Script son básicamente dos vistas diferentes que utiliza el mismo editor de ActionScript, la configuración y las preferencias para ActionScript dentro de Flash afectan a ambas vistas.

La configuración de publicación del documento de Flash se edita para cambiar la versión de ActionScript que debe utilizarse al publicar el documento. También puede establecer la ruta de clases para el documento actual, omitiendo la ruta de clases global de ActionScript.

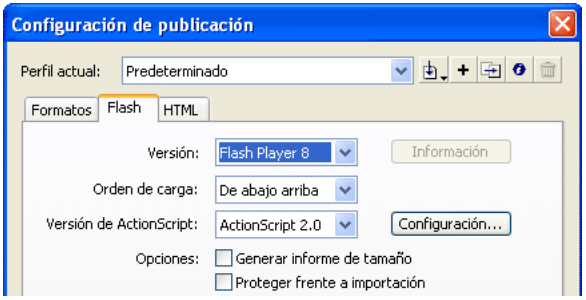
Para más información sobre la modificación de la configuración de publicación de ActionScript, consulte “Modificación de la configuración de publicación de ActionScript” en la página 67. Para más información sobre la configuración de una ruta de clases de documento o global, consulte “Modificación de la ruta de clases” en la página 68.

# Modificación de la configuración de publicación de ActionScript

De manera predeterminada, la versión de ActionScript al publicar un documento de Flash es la 2.0 y la ruta de clases se hereda de la configuración de ruta de clases global. Si necesita cambiar la versión de ActionScript o especificar una ruta de clases de documento, puede hacerlo editando la configuración de publicación.

### Para cambiar la versión de ActionScript:

1. Seleccione Archivo > Configuración de publicación y elija la ficha Flash.




2. Seleccione la versión de ActionScript en el menú emergente.

   ActionScript 2.0 está seleccionado de forma predeterminada. Si escribe los scripts en ActionScript 1.0 en lugar de 2.0, cambie esta configuración antes de publicar el documento de Flash.

El compilador de ActionScript 2.0 compila todo el código ActionScript 1.0, con la siguiente excepción: la sintaxis con barras (/) utilizada para indicar las rutas de clips de película (por ejemplo, `parentClip/testMC:varName= "hello world"`) genera errores de compilación si se selecciona ActionScript 2.0 como versión de ActionScript. Para resolver este problema, reescriba el código empleando notación de puntos (.) en lugar de barras o seleccione el compilador de ActionScript 1.0.

Se utiliza el botón Configuración (junto al menú emergente de la versión de ActionScript) para modificar la ruta de clases de documento. Para más información, consulte "Modificación de la ruta de clases" en la página 68.

# Modificación de la ruta de clases

Al utilizar ActionScript 2.0, también podrá establecer una ruta de clases de documento. Esto le será de utilidad al crear sus propias clases si desea sustituir la ruta de clases global de ActionScript establecida en las preferencias de ActionScript.

El cambio de la ruta de clases en la configuración de publicación sólo afecta al archivo de Flash actual.

Puede utilizar el cuadro de diálogo Preferencias para modificar la ruta de clases global. Para modificar la configuración de ruta de clases de documentos, utilice el cuadro de diálogo Configuración de publicación para el archivo FLA. En ambos casos, puede añadir rutas de directorios absolutas (por ejemplo, C:/my_classes) y rutas de directorios relativas (por ejemplo, ..\my_classes o ".").

### Para modificar la ruta de clases global:

1. Seleccione Edición > Preferencias (Windows) o Flash > Preferencias (Macintosh) y abra el cuadro de diálogo Preferencias.
2. Haga clic en ActionScript de la lista Categoría y seleccione Configuración de ActionScript 2.0.
3. Siga uno de estos procedimientos:
   - Para añadir un directorio a la ruta de clases, haga clic en Buscar ruta, vaya al directorio que desee añadir y haga clic en Aceptar.

     También puede hacer clic en Añadir nueva ruta (+) para añadir una nueva línea a la lista de rutas de clases. Haga doble clic en la nueva línea, escriba una ruta relativa o absoluta y haga clic en Aceptar.
   - Para editar un directorio de ruta de clases existente, seleccione la ruta en la lista de rutas de clases, haga clic en Buscar ruta, busque el directorio que desee añadir y haga clic en Aceptar.

     También puede hacer doble clic en la ruta de la lista de rutas de clases, escribir la ruta que desee y hacer clic en Aceptar.
   - Para eliminar un directorio de la ruta de clases, seleccione la ruta en la lista de rutas de clases y haga clic en Quitar ruta seleccionada.

| NOTA | No elimine la ruta de clases global absoluta (consulte Rutas de clases globales y de documentos). Flash utiliza esta ruta de clases para obtener acceso a clases incorporadas. Si elimina por error esta ruta de clases, restáurela añadiendo $(LocalData)/Classes como nueva ruta de clases. |
|---|---|

**Para modificar la ruta de clases de documentos:**

1. Seleccione Archivo > Configuración de publicación para abrir el cuadro de diálogo Configuración de publicación.
2. Haga clic en la ficha Flash.
3. Haga clic en el botón Configuración situado junto al menú emergente Versión de ActionScript.
4. Siga uno de estos procedimientos:
   - Para añadir un directorio a la ruta de clases, haga clic en Buscar ruta, vaya al directorio que desee añadir y haga clic en Aceptar.

     También puede hacer clic en Añadir nueva ruta (+) para añadir una nueva línea a la lista de rutas de clases. Haga doble clic en la nueva línea, escriba una ruta relativa o absoluta y haga clic en Aceptar.
   - Para editar un directorio de ruta de clases existente, seleccione la ruta en la lista de rutas de clases, haga clic en Buscar ruta, busque el directorio que desee añadir y haga clic en Aceptar.

     También puede hacer doble clic en la ruta de la lista de rutas de clases, escribir la ruta que desee y hacer clic en Aceptar.
   - Para eliminar un directorio de la ruta de clases, seleccione la ruta en la lista de rutas de clases y haga clic en Quitar ruta seleccionada.

Para más información sobre la configuración y modificación de las rutas de clases, consulte "Configuración y modificación de la ruta de clases" en la página 211.

## Archivos de configuración que se instalan con Flash 8

Al instalar Flash Basic 8 o Flash Professional 8, se colocan en el sistema una serie de carpetas y archivos de configuración relacionados con ActionScript. Puede utilizar estos archivos para aplicar determinadas configuraciones al entorno de edición. Como siempre, se recomienda que realice modificaciones con precaución y guarde una copia de seguridad de los archivos que vaya a cambiar.

**Carpeta de clases de ActionScript** Contiene todas las clases de ActionScript (archivos AS) que se incluyen en Flash Professional 8 o Flash Basic 8. Esta carpeta suele estar en rutas como las siguientes:

- Windows: Disco duro\Documents and Settings*usuario*\Configuración local\Datos de programa\Macromedia\Flash 8*idioma*\Configuration\Classes.

- Macintosh: Disco duro/Users/*usuario*/Library/Application Support/Macromedia/Flash 8/*idioma*/Configuration/Classes.

  La carpeta Classes se organiza en carpetas que incluyen directorios que contienen las clases para Flash Player 7 (FP7) y Flash Player 8 (FP8). También contiene un directorio para el paquete mx (mx), que se utiliza en los dos reproductores y en los archivos ASO (aso). Para más información sobre los archivos ASO, consulte "Utilización de archivos ASO" en la página 254. Para más información sobre la organización de este directorio, consulte el archivo Readme de la carpeta Classes.

**Carpeta de clases de inclusión** Contiene todos los archivos de inclusión globales de ActionScript y está ubicada en:

- Windows: Disco duro\Documents and Settings*usuario*\Configuración local\Datos de programa\Macromedia\Flash 8*idioma*\Configuration\Include.
- Macintosh: Disco duro/Users/*usuario*/Library/Application Support/Macromedia/Flash 8/*idioma*/Configuration/Include.

**Archivo de configuración ActionsPanel.xml** Incluye el archivo de configuración de sugerencias o consejos sobre códigos de ActionScript y está ubicado en:

- Windows: Disco duro\Documents and Settings*usuario*\Configuración local\Datos de programa\Macromedia\Flash 8*idioma*\Configuration\ActionsPanel\ActionScript_1_2.
- Macintosh: Disco duro/Users/*usuario*/Library/Application Support/Macromedia/Flash 8/*idioma*/Configuration/ActionsPanel/ActionScript_1_2.

**Archivo de configuración AsColorSyntax.xml** El archivo de configuración de color para la sintaxis del código ActionScript; ubicado en:

- Windows: Disco duro\Documents and Settings*usuario*\Configuración local\Datos de programa\Macromedia\Flash 8*idioma*\Configuration\ActionsPanel\.
- Macintosh: Disco duro/Users/*usuario*/Library/Application Support/Macromedia/Flash 8/*idioma*/Configuration/ActionsPanel.

CAPÍTULO 3

# ActionScript

3

Las funciones de programación orientada a objetos (OOP) de ActionScript 2.0 se basan en la propuesta de borrador de ECMAScript 4 que actualmente está desarrollando ECMA TC39-TG1 (consulte www.mozilla.org/js/language/es4/index.html). Dado que la propuesta ECMA-4 no es todavía un estándar y continúa cambiando, ActionScript 2.0 se basa de manera estricta en esta especificación.

ActionScript 2.0 admite todos los elementos estándar del lenguaje ActionScript; permite escribir scripts que se ajustan mejor a los estándares utilizados en otros lenguajes orientados a objetos, como Java. ActionScript 2.0 resultará de especial interés para los desarrolladores con conocimientos intermedios o avanzados de Flash que crean aplicaciones que requieren la implementación de clases y subclases. ActionScript 2.0 también permite declarar el tipo de objeto de una variable al crearla (consulte "Asignación de tipos de datos y "strict data typing"" en la página 337) y proporciona una gestión de errores de compilador con mejoras considerables (consulte Apéndice A, "Mensajes de error", en la página 817).

Puntos clave de ActionScript 2.0:

- Los scripts que utilizan ActionScript 2.0 para definir clases o interfaces deben almacenarse como archivos de script externos, con una sola clase definida en cada script; es decir, las clases y las interfaces no pueden definirse en el panel Acciones.
- Puede importar archivos de clase individuales implícitamente (almacenándolos en una ubicación especificada por rutas de búsqueda globales o específicas del documento y luego utilizándolos en un script) o explícitamente (utilizando el comando `import`); puede importar paquetes (conjuntos de archivos de clase en un directorio) utilizando caracteres comodín.

- Las aplicaciones desarrolladas con ActionScript 2.0 son compatibles con Flash Player 6 y versiones posteriores.

| ATENCIÓN | La configuración de publicación predeterminada para los nuevos archivos creados en Flash 8 es ActionScript 2.0. Si tiene previsto modificar un archivo FLA existente con ActionScript 1.0 de modo que utilice la sintaxis de ActionScript 2.0, asegúrese de que el archivo FLA tiene especificado ActionScript 2.0 en la configuración de publicación. Si no lo tiene especificado, el archivo no se compilará de forma correcta, aunque Flash no generará necesariamente errores de compilador. |
|---|---|

Para más información sobre la utilización de ActionScript 2.0 para escribir programas orientados a objetos en Flash, consulte el Capítulo 6, "Clases", en la página 195.

Aunque Macromedia recomienda el uso de ActionScript 2.0, puede continuar utilizando la sintaxis de ActionScript 1.0, especialmente si realiza trabajos más tradicionales de Flash, como animaciones sencillas que no exijan interacción del usuario.

# Qué es ActionScript

Las características principales de ActionScript 2.0 son las siguientes:

**Modelo común de programación orientada a objetos (OOP, Object Oriented Programming)** La principal función de ActionScript 2.0 es un modelo común para crear programas orientados a objetos. ActionScript 2.0 implementa varios nuevos conceptos y palabras clave de programación orientada a objetos, como por ejemplo *clase*, *interfaz* y *paquetes*, con los que estará familiarizado si ha programado alguna vez en código Java.

El modelo OOP que proporciona ActionScript 2.0 es una "formalización sintáctica" del método de cadenas prototipo utilizado en versiones anteriores de Macromedia Flash para crear objetos y establecer la herencia. Con ActionScript 2.0, puede crear clases personalizadas y ampliar las clases incorporadas en Flash.

**Strict data typing** ActionScript 2.0 también permite especificar de forma explícita tipos de datos para variables, parámetros de función y tipos de devolución de funciones. Por ejemplo, en el siguiente código se declara una variable denominada `userName` de tipo String (una clase o un tipo de datos de ActionScript incorporado).

```
var userName:String = "";
```

**Advertencias y errores del compilador** Las dos funciones anteriores permiten a la herramienta de edición y al compilador proporcionar advertencias y mensajes de error que le ayudan a encontrar los fallos en las aplicaciones con más rapidez que anteriormente en Flash.

Al utilizar ActionScript 2.0, asegúrese de que la configuración de publicación del archivo FLA especifica ActionScript 2.0. Ésta es la configuración predeterminada para los archivos creados con Flash MX 2004 y Flash 8. No obstante, si abre un archivo FLA más antiguo que utiliza ActionScript 1.0 y lo reescribe utilizando ActionScript 2.0, debe cambiar la configuración de publicación del archivo FLA a ActionScript 2.0. Si no lo hace, el archivo FLA no se compilará correctamente y no se generarán errores.

# Selección entre ActionScript 1.0 y ActionScript 2.0

Cuando se inicia un nuevo documento o aplicación en Flash, debe decidirse cómo organizar los archivos asociados. Pueden utilizarse clases en algunos proyectos, por ejemplo, para crear aplicaciones o archivos FLA complejos, pero no todos los documentos utilizan clases. Por ejemplo, en muchos ejemplos breves de la documentación no se utilizan clases. Utilizar clases para almacenar funcionalidad no es la mejor solución ni la más sencilla en las aplicaciones pequeñas o en los archivos FLA sencillos. Suele resultar más eficaz colocar el código ActionScript dentro del documento. En este caso, intente colocar todo el código en el menor número de fotogramas posible de la línea de tiempo y evite colocar código en las instancias (por ejemplo, botones o clips de película) de un archivo FLA.

Cuando se crea un proyecto pequeño, suele ser más trabajoso utilizar clases o archivos de código externos para organizar el código ActionScript que añadir el código ActionScript al archivo FLA. En ocasiones, es más sencillo mantener todo el código ActionScript en el archivo FLA en lugar de colocarlo en una clase importada. Esto no significa que deba utilizar necesariamente ActionScript 1.0. Podría decidir colocar el código dentro del archivo FLA mediante ActionScript 2.0 con la función “strict data typing” y los nuevos métodos y propiedades. ActionScript 2.0 también ofrece una sintaxis que cumple los estándares de otros lenguajes de programación. De esta forma, es más sencillo y útil aprender el lenguaje. Por ejemplo, se sentirá familiarizado con ActionScript si encuentra otro lenguaje basado en los mismos estándares de estructura y sintaxis. O bien podrá aplicar este conocimiento a otros lenguajes que aprenda en el futuro. ActionScript 2.0 permite utilizar un enfoque orientado a objetos para desarrollar aplicaciones a través de un conjunto adicional de elementos de lenguaje, lo que puede ser ventajoso para el desarrollo de la aplicación.

En algunos casos, no es posible elegir la versión de ActionScript que se utilizará. Si crea un archivo SWF destinado a una versión anterior de Flash Player como, por ejemplo, una aplicación para dispositivo móvil, debe utilizar ActionScript 1.0, que es compatible con Flash Player en distintos dispositivos.

Recuerde que debería seguir las prácticas recomendadas, independientemente de la versión de ActionScript. En ambas versiones se aplican muchas de estas prácticas, como mantener la coherencia en el uso de mayúsculas y minúsculas, utilizar la capacidad de completar código, mejorar la legibilidad, evitar las palabras clave en los nombres de instancia y mantener una convención de asignación de nombres coherente.

Si tiene previsto actualizar la aplicación en futuras versiones de Flash, o aumentarla y hacerla más compleja, debería utilizar ActionScript 2.0 y las clases para que sea más sencillo actualizarla y modificarla.

# ActionScript y Flash Player

Si compila un archivo SWF que contiene ActionScript 2.0 con una configuración de publicación establecida en Flash Player 6 y ActionScript 1.0, el código funcionará mientras no se utilicen clases de ActionScript 2.0. El código no distingue entre mayúsculas y minúsculas, sólo en Flash Player. Sin embargo, si se compila el archivo SWF con la opción Configuración de publicación establecida en Flash Player 7 y ActionScript 1.0, Flash aplica la distinción entre mayúsculas y minúsculas.

Las anotaciones de tipos de datos (tipos de datos estrictos) se aplican durante el tiempo de compilación para Flash Player 7 y 8 cuando la configuración de publicación está establecida en ActionScript 2.0.

ActionScript 2.0 se compila al código de bits de ActionScript 1.0 cuando publica las aplicaciones, de forma que puede utilizar la versión Flash Player 6, 7 u 8 mientras trabaja con ActionScript 2.0.

CAPÍTULO 4

# Principios básicos de la sintaxis y el lenguaje

4

El aprendizaje de la sintaxis y las sentencias de ActionScript es como aprender a crear frases mediante la combinación de palabras, para luego formar párrafos enteros. ActionScript puede resultar así de sencillo. Por ejemplo, en español, el punto indica el final de una frase; en ActionScript, el final de una sentencia se indica mediante un punto y coma. En el lenguaje ActionScript, puede escribir una acción `stop()` para detener la cabeza lectora de una instancia de clip de película o la reproducción indefinida de un archivo SWF. O bien puede escribir miles de líneas de código que permitan hacer funcionar a una aplicación de banca interactiva. Como puede comprobar, ActionScript puede hacer cosas muy sencillas o muy complejas.

En el Capítulo 10, "Datos y tipos de datos", hemos aprendido de qué forma usa el lenguaje ActionScript los datos y cómo puede aplicarle formato en el código. En este capítulo se describe cómo formar sentencias en ActionScript utilizando su *sintaxis*. Contiene numerosos fragmentos pequeños de código y algunos ejemplos que permiten demostrar los conceptos básicos de este lenguaje. En posteriores capítulos se incluyen ejemplos de código más largos y progresivamente complejos en los que se combinan y aplican los conceptos descritos en este capítulo.

Las reglas generales descritas en esta sección son aplicables a todo el lenguaje ActionScript. La mayoría de los términos de ActionScript también tienen sus propios requisitos; para saber cuáles son las reglas de un término determinado, consulte la entrada correspondiente en *Referencia del lenguaje ActionScript 2.0*.

La aplicación de los conceptos básicos de ActionScript de manera que se obtengan programas elegantes puede resultar una tarea ardua para los usuarios sin experiencia en ActionScript. Para más información sobre cómo aplicar las reglas descritas en esta sección, consulte el Capítulo 19, "Recomendaciones y convenciones de codificación para ActionScript 2.0", en la página 775.

**NOTA** En este capítulo, se añade código ActionScript directamente a un fotograma de la línea de tiempo. En capítulos posteriores, se utilizan clases para separar el código ActionScript del archivo FLA.

Para más información sobre los aspectos básicos de la utilización de la sintaxis y el lenguaje ActionScript, consulte los temas siguientes:



# Sintaxis, sentencias y expresiones

El lenguaje ActionScript está formado por clases incorporadas. Deberá utilizar la *sintaxis* correcta de ActionScript para formar sentencias de manera que el código se compile y funcione correctamente en Flash. En este caso, la sintaxis se refiere a la gramática y la ortografía de un lenguaje que le permite programar. El compilador no comprende la sintaxis incorrecta, por lo que observará errores o advertencias en el panel Salida cuando intente comprobar el documento en el entorno de prueba. Por consiguiente, la sintaxis es un conjunto de reglas y directrices que le ayudan a formar código ActionScript correcto.

Una *sentencia* es una instrucción que se da al archivo FLA para que haga algo, como, por ejemplo, ejecutar una acción concreta. Por ejemplo, puede utilizar una sentencia condicional para determinar si algo es verdadero o si existe. Posteriormente, podría ejecutar las acciones que especifique, como, por ejemplo, funciones o expresiones, en función de si la condición es verdadera o no. La sentencia `if` es una sentencia condicional que evalúa una condición para determinar la siguiente acción que debe tener lugar en el código.

```
// sentencia if
if (condition) {
  // sentencias;
}
```

Para más información sobre sentencias, consulte “Sentencias” en la página 106.

Las *expresiones*, a diferencia de las sentencias, son cualquier combinación válida de símbolos de ActionScript que representan un valor. Las expresiones tienen *valores*, mientras que los valores y las propiedades tienen *tipos*. Una expresión puede estar formada por operadores y operandos, valores, funciones y procedimientos. La expresión sigue las reglas de prioridad y asociación de ActionScript. Normalmente, Flash Player interpreta la expresión y luego devuelve un valor que puede utilizar en la aplicación.

Por ejemplo, el siguiente código es una expresión:

```
x + 2
```

En la expresión anterior, `x` y `2` son operandos y + es un operador. Para más información sobre operadores y operandos, consulte "Operadores" en la página 142. Para más información sobre objetos y propiedades, consulte "Tipo de datos Object (objeto)" en la página 335.

El modo en que aplique formato al código ActionScript también determinará las posibilidades de mantenimiento de dicho código. Por ejemplo, es difícil leer la lógica de un archivo FLA que no tenga sangrías ni comentarios o que presenten un formato y unas convenciones de asignación de nombres incoherentes. Al sangrar los bloques de código ActionScript (como bucles y sentencias `if`), el código resulta más fácil de leer y depurar en el caso de que se detecten problemas. Para más información sobre la aplicación de formato de ActionScript, consulte "Aplicación de formato a la sintaxis de ActionScript" en la página 808. También encontrará una aplicación correcta de formato al código ActionScript en estas secciones.

Para más información sobre los principios básicos de la sintaxis y el lenguaje, consulte los siguientes temas:

- "Diferencias entre ActionScript y JavaScript"
- "Distinción entre mayúsculas y minúsculas"

## Diferencias entre ActionScript y JavaScript

ActionScript es parecido al lenguaje de programación JavaScript. No es necesario tener conocimientos de JavaScript para utilizar y aprender ActionScript; sin embargo, si los tiene, ActionScript le resultará familiar.

En este manual no se pretende enseñar programación en general. Existen muchos recursos disponibles que proporcionan información sobre los conceptos generales de programación y sobre el lenguaje JavaScript.

- La especificación del lenguaje ECMAScript (ECMA-262) edición 3 se deriva de JavaScript y sirve de estándar internacional del lenguaje JavaScript. ActionScript se basa en esta especificación. Para más información, consulte www.ecma-international.org/publications/standards/Ecma-262.htm.
- El sitio sobre tecnología Java dispone de tutoriales sobre programación orientada a objetos (http://java.sun.com/docs/books/tutorial/java/index.html) específicamente diseñados para el lenguaje Java, aunque resultan útiles para comprender conceptos que puede aplicar a ActionScript.

En la siguiente lista se detallan algunas de las diferencias entre ActionScript y JavaScript:

- ActionScript no admite objetos específicos de navegador como Documento, Ventana y Anclaje.
- ActionScript no admite completamente todos los objetos incorporados de JavaScript.
- ActionScript no admite algunas construcciones sintácticas de JavaScript, como las etiquetas de sentencia.
- En ActionScript, la función `eval()` sólo puede realizar referencias de variables.
- ActionScript 2.0 admite diversas funciones que no se incluyen en la especificación ECMA-262, como las clases y la comprobación de tipos al compilar. Muchas de estas funciones se modelan a partir de la especificación del lenguaje ECMAScript (ECMA-262) edición 3 (consulte www.ecma-international.org/publications/standards/Ecma-262.htm).
- ActionScript no admite expresiones regulares mediante el objeto RegExp. No obstante, Macromedia Central sí es compatible con el objeto RegExp. Para más información sobre Macromedia Central, consulte www.macromedia.com/software/central.

## Distinción entre mayúsculas y minúsculas

Al escribir código ActionScript para Flash Player 7 y versiones posteriores, el código distingue entre mayúsculas y minúsculas. Esto significa que las variables con ligeras diferencias de mayúsculas y minúsculas se consideran diferentes entre sí. Esto se muestra en el siguiente código ActionScript:

```
// se utiliza una combinación de mayúsculas y minúsculas
var firstName:String = "Jimmy";
// se utilizan sólo minúsculas
trace(firstname); // undefined
```

O bien puede escribir lo siguiente:

```
// En archivos de Flash Player 8
// y ActionScript 1,0 o ActionScript 2.0
//
// Define las propiedades de dos objetos diferentes
cat.hilite = true;
CAT.hilite = true;

// Crea tres variables diferentes
var myVar:Number = 10;
var myvar:Number = 10;
var mYvAr:Number = 10;
```

**NOTA** No es recomendable diferenciar entre variables o cualquier identificador mediante mayúsculas y minúsculas. Para más información sobre la asignación de nombres a las variables, consulte el Capítulo 19, "Recomendaciones y convenciones de codificación para ActionScript 2.0", en la página 775.

Cuando publique para versiones de Flash Player (Flash Player 6 y anteriores), Flash traza la cadena `Jimmy` en el panel Salida. Dado que Flash Player 7 y versiones posteriores distinguen entre mayúsculas y minúsculas, `firstName` y `firstname` son dos variables independientes (cuando se utiliza ActionScript 1.0 o ActionScript 2.0). Es importante entender este concepto. Si ha creado archivos FLA para Flash Player 6 o versiones anteriores con uso diferente de mayúsculas y minúsculas en las variables, la funcionalidad y los archivos podrían romperse al convertir el archivo o la aplicación para una versión más reciente de Flash Player.

Por consiguiente, se recomienda seguir convenciones coherentes de uso de mayúsculas y minúsculas, como las que se utilizan en este manual. Al hacerlo de esta forma, también resultará más fácil distinguir variables, clases y nombres de función. No utilice mayúsculas y minúsculas para distinguir entre dos identificadores. Cambie el nombre de la instancia, variable o clase y no exclusivamente las mayúsculas y minúsculas. Para más información sobre convenciones de codificación, consulte el Capítulo 19, "Recomendaciones y convenciones de codificación para ActionScript 2.0", en la página 775.

La distinción entre mayúsculas y minúsculas puede ser determinante si se trabaja con un servicio Web que utilice sus propias reglas de asignación de nombres de variables y puede afectar al uso de mayúsculas y minúsculas en las variables cuando se devuelven al archivo SWF desde el servidor. Por ejemplo, si utiliza un servicio Web de ColdFusion, los nombres de propiedades de una estructura u objeto pueden tener sólo mayúsculas como, por ejemplo, `FIRSTNAME`. A menos que utilice las mismas mayúsculas y minúsculas en Flash, cabe la posibilidad de que obtenga resultados inesperados.

**NOTA**

La distinción entre mayúsculas y minúsculas también afecta a las variables externas que se cargan en un archivo SWF, como, por ejemplo, las que se cargan con `LoadVars.load()`.

La distinción entre mayúsculas y minúsculas se aplica a los scripts externos, como los archivos de clase de ActionScript 2.0, los scripts que se importan empleando el comando `#include` y los scripts de un archivo FLA. Si encuentra errores de tiempo de ejecución y está exportando a más de una versión de Flash Player, deberá revisar los archivos de script externos y los scripts de archivos FLA para confirmar que ha utilizado las mayúsculas de forma coherente.

La distinción entre mayúsculas y minúsculas se aplica por cada archivo SWF concreto. Si una aplicación Flash Player 8 (que distingue entre mayúsculas y minúsculas) estricta llama a un archivo SWF de Flash Player 6 no estricto, el código ActionScript ejecutado en el archivo SWF de Player 6 será no estricto. Por ejemplo, si utiliza `loadMovie()` para cargar un archivo SWF de Flash Player 6 en un archivo SWF de Flash Player 8, el archivo SWF de la versión 6 continuará sin distinguir entre mayúsculas y minúsculas, mientras que el SWF de la versión 8 se considerará como archivo que distingue entre mayúsculas y minúsculas.

Si activa la función Color de sintaxis, los elementos del lenguaje que haya escrito con el formato correcto de mayúsculas y minúsculas aparecen en azul de forma predeterminada. Para más información, consulte "Palabras reservadas" en la página 104.

# Sintaxis con puntos y rutas de destino

En ActionScript, el operador de punto (.) (*sintaxis con puntos*) permite acceder a propiedades y métodos que pertenecen a un objeto o instancia del escenario. También puede utilizar el operador de punto para identificar la ruta de destino de una instancia (como, por ejemplo, un clip de película), una variable, una función o un objeto.

Una expresión que utiliza la sintaxis con puntos empieza por el nombre del objeto o clip de película seguido de un punto y termina con el elemento que desee especificar. En las siguientes secciones se describe cómo escribir expresiones que utilizan la sintaxis con punto.

Para controlar un clip de película, archivo SWF cargado o un botón, es preciso especificar una *ruta de destino*. Las rutas de destino son direcciones jerárquicas de nombres de instancias de clips de película, variables y objetos de un archivo SWF. Para especificar una ruta de destino de un clip de película o botón, debe asignar un nombre de instancia al clip de película o botón. El nombre de una instancia de clip de película se asigna seleccionando la instancia y escribiendo el nombre de la instancia en el inspector de propiedades del clip de película. También puede especificar el nombre de instancia con código si crea la instancia mediante ActionScript. Se puede utilizar la ruta de destino para asignar una acción a un clip de película u obtener o definir el valor de una variable o propiedad.

Para más información sobre la asignación de un nombre de instancia y el uso de la sintaxis con punto para referirse a una instancia, consulte los siguientes temas:

- "Utilización de la sintaxis con punto para referirse a una instancia" en la página 81.
- "Ámbito y referencias" en la página 86
- "Utilización del botón Ruta de destino" en la página 87
- "Sintaxis con barras" en la página 87

Para más información sobre objetos y propiedades, consulte "Tipo de datos Object (objeto)" en la página 335.

## Utilización de la sintaxis con punto para referirse a una instancia

Para escribir código ActionScript que controle una instancia, como, por ejemplo, un clip de película, o que manipule activos en un archivo SWF cargado, deberá especificar su nombre y dirección en el código. Esto se conoce como *ruta de destino*. Para hacer referencia a objetos en un archivo SWF, se utiliza la sintaxis con punto (también conocida como *notación con puntos*). Por ejemplo, es necesario hacer referencia a una instancia de un clip de película o un botón antes de aplicarle una acción. La sintaxis con punto le ayuda a crear una ruta de acceso a la instancia a la que debe referirse. La ruta de acceso a la instancia a la que debe referirse a veces se denomina ruta de destino.

Un archivo FLA tiene una jerarquía concreta. Puede crear instancias en el escenario o utilizar código ActionScript. Puede incluso crear instancias que estén dentro de otras instancias. O puede que tenga instancias anidadas dentro de otras instancias. Puede manipular cualquier instancia siempre y cuando le asigne un nombre.

Para asignar nombre a las instancias, deberá utilizar un *nombre de instancia*, que puede especificar de dos formas diferentes (descritas a continuación):

- Manualmente, seleccionando una instancia y escribiendo un nombre de instancia en el inspector de propiedades (cuando una instancia se encuentra en el escenario).
- Dinámicamente, utilizando ActionScript. Puede crear una instancia mediante código ActionScript y asignarle un nombre de instancia al crearla.

Para asignarle un nombre a la instancia en el inspector de propiedades, escriba el nombre en el cuadro de texto Nombre de instancia.

También puede asignar un nombre de instancia a un objeto que cree mediante código ActionScript. Esta operación puede resultar tan sencilla como el siguiente código:

```
this.createEmptyMovieClip("pic_mc", this.getNextHighestDepth());
pic_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
```

Este código crea un nuevo clip de película, al que asigna el nombre de instancia pic_mc. Seguidamente, puede manipular la instancia `pic_mc` utilizando código, por ejemplo, cargando una imagen en ella, como se muestra en el código anterior.

Para más información sobre la utilización de ámbitos, consulte "Ámbito y referencias" en la página 86 y "Variables y ámbito" en la página 354.

## Referencia a una instancia

Si desea que algo funcione en el archivo SWF, deberá hacer referencia a la instancia en cuestión y luego indicarle que haga algo, como asignarle una acción o cambiar su propiedades. Normalmente deberá definir el lugar en el que se encuentra dicha instancia en el archivo SWF (por ejemplo, en qué línea de tiempo se encuentra o en qué instancia está anidada) mediante la creación de la ruta de destino. Recuerde que ha asignado nombres de instancia a muchas de las instancias del archivo FLA y que luego ha añadido código al archivo FLA que utiliza dichos nombres de instancias. Al hacer esto, está haciendo referencia a dicha instancia en particular e indicándole que haga algo (por ejemplo, mover la cabeza lectora o abrir una página Web). Para más información sobre objetos y propiedades, consulte "Tipo de datos Object (objeto)" en la página 335.

### Para hacer referencia a una instancia:

1. Seleccione Archivo > Nuevo y seleccione Documento de Flash.
2. Seleccione Archivo > Guardar como y asigne al archivo el nombre **target.fla**.
3. Dibuje una figura en el escenario con la herramienta Óvalo. Dibuje un óvalo de cualquier tamaño y color.
4. Con la herramienta Selección, seleccione el óvalo en el escenario.

| SUGERENCIA | Recuerde que debe seleccionar el trazo y el relleno si es preciso. |
|---|---|

5. Seleccione Modificar > Convertir en símbolo, elija la opción Clip de Película y luego haga clic en Aceptar para crear el símbolo.
6. Seleccione el clip de película en el escenario y asígnele el nombre de instancia **myClip** en el inspector de propiedades.
7. Inserte una capa nueva y asígnele el nombre **actions**.
8. Añada el siguiente código ActionScript al fotograma 1 de la capa actions:

   ```
   myClip._xscale = 50;
   ```

   Esta línea de código hace referencia a la instancia `myClip` del escenario. El código ActionScript redimensiona la instancia con la mitad de su anchura original. Dado que el código ActionScript está en la misma línea de tiempo que el símbolo del clip de película, sólo tendrá que hacer referencia a la instancia utilizando el nombre de instancia. Si la instancia estuviera en una línea de tiempo diferente o anidada dentro de otra instancia, tendría que adaptar la correspondiente ruta de destino.

## Referencia a una instancia anidada

También puede referirse a instancias que estén anidadas dentro de otras instancias. Quizá desee colocar una segunda instancia de clip de película dentro de la instancia myClip del ejercicio realizado en "Referencia a una instancia" en la página 82. También puede referirse a dicha instancia anidada utilizando ActionScript. Antes de continuar con el siguiente ejercicio, deberá finalizar el ejercicio "Referencia a una instancia" en la página 82 y, seguidamente, realizar los pasos para hacer referencia a una instancia anidada.

### Para referirse a una instancia anidada:

1. Abra el archivo target.fla creado en el procedimiento de hacer referencia a una instancia y cámbiele el nombre a **target2.fla**.
2. Haga doble clic en la instancia myClip del escenario.
3. Seleccione la herramienta Óvalo y dibuje otro óvalo dentro de la instancia myClip.
4. Seleccione la nueva figura y, a continuación, elija Modificar > Convertir en símbolo.
5. Seleccione la opción MovieClip y haga clic en Aceptar.
6. Seleccione la nueva instancia y escriba **myOtherClip** en el cuadro de texto Nombre de instancia del inspector de propiedades.
7. Haga clic en la escena 1 de la barra de edición para regresar a la línea de tiempo principal.
8. Añada el siguiente código ActionScript al fotograma 1 de la capa actions:

   ```
   myClip.myOtherClip._xscale = 50;
   ```

   Este código ActionScript asigna a la instancia myOtherClip el 50% de su anchura original. Dado que el archivo target.fla modificó la propiedad `_xscale` de las instancias de myClip y que myOtherClip es un símbolo anidado, observará que myOtherClip presenta una anchura equivalente al 25 por ciento de su anchura original.

Si utiliza clips de película anidados que tengan sus propias líneas de tiempo, podrá manipular la cabeza lectora de la línea de tiempo de una instancia anidada empleando un código similar al del siguiente fragmento:

```
myClip.nestedClip.gotoAndPlay(15);
myClip.someOtherClip.gotoAndStop("tweenIn");
```

Observe que el clip que usted manipula (como, por ejemplo, `nestedClip`) aparece justo antes de la acción. Observará esta tendencia en posteriores secciones.

No sólo puede acceder a métodos y propiedades predefinidas de instancias del escenario, como se muestra en los ejemplos anteriores. También puede establecer una variable dentro de un clip de película, como se aprecia en el siguiente código, que establece una variable en el clip de película starClip:

```
starClip.speed = 1.1;
starClip.gravity = 0.8;
```

Si las variables de velocidad o gravedad existieran con anterioridad en la instancia del clip de película starClip, los valores anteriores se habrían sobrescrito de inmediato al establecer las nuevas variables. Puede añadir nuevas propiedades al clip de película starClip, ya que la clase MovieClip se definió con la palabra clave `dynamic`. La palabra clave `dynamic` especifica que los objetos basados en la clase especificada (en este caso MovieClip) pueden añadir propiedades dinámicas y acceder a ellas en tiempo de ejecución. Para más información sobre la sentencia dinámica, consulte %{dynamic statement}% en *Referencia del lenguaje ActionScript 2.0*.

## Referencia a instancias dinámicas y contenido cargado

También puede crear un objeto empleando código ActionScript y hacer referencia a él posteriormente mediante una ruta de destino. Por ejemplo, puede utilizar el siguiente código ActionScript para crear un clip de película. Luego puede cambiar el giro de dicho clip de película empleando código ActionScript, como se muestra en el siguiente ejemplo:

### Para hacer referencia a una instancia de clip de película creada dinámicamente:

1. Cree un nuevo documento Flash y guarde el archivo como **targetClip.fla**.
2. Inserte una capa nueva y asígnele el nombre **actions**.
3. Añada el siguiente código ActionScript al fotograma 1 de la capa actions:

   ```
   this.createEmptyMovieClip("rotateClip", this.getNextHighestDepth());
   trace(rotateClip);
   rotateClip._rotation = 50;
   ```

4. Seleccione Control > Probar película para probar el documento.

   La presencia de la sentencia `trace` indica que ha creado un clip de película, pero no puede ver nada en el escenario. Aunque haya añadido código que cree una instancia de clip de película, no verá nada en el escenario a no ser que añada algo al clip de película. Por ejemplo, puede cargar una imagen en el clip de película.
5. Regrese al entorno de edición y abra el panel Acciones.

6. Escriba el siguiente código ActionScript detrás del código añadido en el paso 3:

```
rotateClip.loadMovie("http://www.helpexamples.com/flash/images/
  image1.jpg");
```

   Este código carga una imagen en el clip de película rotateClip que ha creado mediante código. Se hace referencia a la instancia rotateClip mediante código ActionScript.

7. Seleccione Control > Probar película para probar el documento.

   Ahora debería de ver una imagen en el escenario que gira 50º en el sentido de las agujas del reloj.

También puede hacer referencia o identificar partes de un archivo SWF que se cargan en un archivo SWF base.

### Para identificar un archivo SWF cargado:

- Utilice `_levelX`, donde *X* es el número de nivel especificado en la función `loadMovie()` que ha cargado el archivo SWF.

  Por ejemplo, un archivo SWF que se ha cargado en el nivel 99 tiene la ruta de destino `_level99`. En el ejemplo siguiente, se carga un archivo SWF en el nivel 99 y su visibilidad se establece en `false`:

```
//Cargue SWF en el nivel 99.
loadMovieNum("contents.swf", 99);
//Establezca la visibilidad del nivel 99 como false.
loaderClip.onEnterFrame = function(){
  _level99._visible = false;
};
```

**SUGERENCIA**

Normalmente resulta aconsejable evitar el uso de niveles si en su lugar puede cargar contenido en clips de película de diferentes profundidades. El método `MovieClip.getNextHighestDepth()` le permite crear nuevas instancias de clip de película en el escenario dinámicamente sin tener que comprobar si ya hay una instancia en una determinada profundidad.

## Definición de variables mediante una ruta

Puede definir variables para instancias anidadas dentro de otras instancias. Por ejemplo, si desea definir una variable para un formulario situado dentro de otro formulario, puede utilizar el siguiente código. La instancia submitBtn está dentro de `formClip` en la línea de tiempo principal:

```
this.formClip.submitBtn.mouseOver = true;
```

Puede expresar un método o propiedad para un objeto concreto (como un clip de película o un campo de texto) siguiendo este patrón. Por ejemplo, la propiedad de un objeto sería

```
myClip._alpha = 50;
```

## Ámbito y referencias

Al anidar instancias, el clip de película que anida un segundo clip de película se conoce como instancia *principal* de la instancia anidada. La instancia anidada se conoce como instancia secundaria. El escenario principal y la línea de tiempo principal son básicamente un clip de película en sí mismos, por lo que se puede hacer referencia a ellos como tal. Para más información sobre ámbitos, consulte "Variables y ámbito" en la página 354.

Puede hacer referencia a instancias principales y líneas de tiempo principales mediante código ActionScript. Cuando desee hacer referencia a la línea de tiempo actual, deberá utilizar la palabra clave `this`. Por ejemplo, para hacer referencia a un clip de película denominado myClip situado en la línea de tiempo principal actual, deberá utilizar

```
this.myClip.
```

Opcionalmente, puede colocar la palabra clave `this` y utilizar simplemente

```
myClip
```

Puede optar por añadir la palabra clave `this` para lograr una mayor legibilidad y coherencia. Para más información sobre prácticas de codificación recomendadas, consulte el Capítulo 19, "Recomendaciones y convenciones de codificación para ActionScript 2.0", en la página 775.

Si traza el clip de película para cualquiera de los anteriores fragmentos, observará `_level0.myClip` en el panel Salida. No obstante, si tiene código ActionScript situado dentro del clip de película myClip pero desea hacer referencia a la línea de tiempo principal, haga referencia al clip de película principal (que es el escenario principal). Haga doble clic en un clip de película y coloque el siguiente código ActionScript en la línea de tiempo del clip de película:

```
trace("me: " + this);
trace("my parent: " + this._parent);
```

Pruebe el archivo SWF y verá el siguiente mensaje en el panel Salida:

```
me: _level0.myClip
my parent: _level0
```

Esto indica que ha hecho referencia a la línea de tiempo principal. Puede utilizar `parent` para crear una ruta relativa a un objeto. Por ejemplo, si el clip de película `dogClip` está anidado dentro del clip de película de animación animalClip, la siguiente sentencia de la instancia dogClip indica a animalClip que interrumpa la animación:

```
this._parent.stop();
```

Si está familiarizado con Flash y ActionScript, es posible que haya observado que hay usuarios que emplean el ámbito `_root`. El ámbito `_root` generalmente hace referencia a la línea de tiempo principal del documento Flash actual. Debe evitar utilizar el ámbito `_root` a no ser que resulte absolutamente necesario. Puede utilizar rutas de destino relativas en lugar de `_root`.

Si utiliza `_root` en el código, puede que encuentre errores al cargar el archivo SWF en otro documento de Flash. Al cargarse el archivo SWF en otro archivo SWF diferente, puede que `_root` en el archivo cargado señale al ámbito raíz del archivo SWF en el que se carga en lugar de hacer referencia a su propia raíz como era su intención. Esto puede producir resultados impredecibles o romper completamente la funcionalidad.

## Utilización del botón Ruta de destino

En ocasiones, lleva tiempo averiguar cuál la ruta de destino o qué ruta de destino necesita para un código determinado. Si hace referencia a una instancia del escenario, puede utilizar el botón Ruta de destino para determinar la ruta de dicha instancia.

### Para utilizar el botón de ruta de destino:

1. Abra el panel Acciones (Ventana > Acciones) y haga clic en el botón Insertar una ruta de destino. Los clips de película del documento actual aparecen en un cuadro de diálogo.
2. Seleccione una de las instancias de la lista del cuadro de diálogo.
3. Haga clic en Aceptar.
4. La ruta de destino de la instancia seleccionada aparece en el panel Script.

## Sintaxis con barras

La sintaxis con barras se utilizó en Flash 3 y 4 para indicar la ruta de destino de un clip de película o de una variable. Esta sintaxis es compatible con ActionScript 1.0 en Flash Player 7 y versiones anteriores, pero no es compatible con ActionScript 2.0 y Flash Player 7 o Flash Player 8.

No es recomendable utilizar la sintaxis con barras, a no ser que no tenga otra opción, como al crear contenido específicamente pensado para Flash Player 4 o Flash Lite 1.1 (y versiones anteriores), donde es preciso usar la sintaxis con barras. Para más información sobre Flash Lite, consulte la página del producto Flash Lite.

# Signos de lenguaje

Existen diversos *signos* de lenguaje en Flash. El tipo más habitual de signos son el punto y coma (;), los dos puntos (:), los paréntesis [()] y las llaves ({}). Cada uno de estos signos tiene un significado específico en el lenguaje de Flash y contribuye a definir tipos de datos, terminar sentencias o estructurar el código ActionScript. En las siguientes secciones se describe cómo utilizar los signos en el código.

Para más información sobre los signos de lenguaje, consulte los siguientes temas:

- "Punto y coma y dos puntos" en la página 88
- "Llaves" en la página 89
- "Paréntesis" en la página 93
- "Literales" en la página 94
- "Comentarios" en la página 95

Para más información sobre el operador de punto (.) y los operadores de acceso a matrices ([]), consulte "Utilización de operadores de punto y de acceso a una matriz" en la página 152. Para más información sobre los espacios en blanco y la aplicación de formato al código, consulte "Aplicación de formato a la sintaxis de ActionScript" en la página 808.

## Punto y coma y dos puntos

Las sentencias de ActionScript terminan con un carácter de punto y coma (;), como se muestra en las dos líneas de código siguientes:

```
var myNum:Number = 50;
myClip._alpha = myNum;
```

Aunque omita el carácter de punto y coma, el compilador de ActionScript dará por hecho que cada línea de código representa a una sentencia independiente. No obstante, el uso del punto y coma es una práctica recomendable en la creación de scripts porque permite lograr una mejor legibilidad del código. Al hacer clic en el botón Formato automático del panel Acciones o la ventana Script, se añaden puntos y comas finales al terminar la sentencia de manera predeterminada.

| NOTA | El uso del punto y coma para terminar una sentencia le permite colocar más de una sentencia en una misma línea, pero, al hacerlo, normalmente el código resulta más difícil de leer. |
|---|---|

Otro lugar en el que se utiliza el punto y coma es en los bucles `for`. El punto y coma permite separar parámetros, como se muestra en el siguiente ejemplo. El ejemplo reproduce indefinidamente de 0 a 9 y luego muestra cada número en el panel Salida:

```
var i:Number;
for (i = 0; i < 10; i++) {
  trace(i); // 0,1,...,9
}
```

Los dos puntos (:) se utilizan en el código para asignar tipos de datos a las variables. Para asignar un tipo de datos específico a un elemento, especifique el tipo con la palabra clave `var` y la sintaxis de signo de dos puntos posterior, como se muestra en el siguiente ejemplo:

```
// "strict typing" de variable u objeto
var myNum:Number = 7;
var myDate:Date = new Date();
// "strict typing" de los parámetros
function welcome(firstName:String, myAge:Number) {
}
// "strict typing" de los parámetros y del valor devuelto
function square(num:Number):Number {
  var squared:Number = num * num;
  return squared;
}
```

Puede declarar el tipo de datos de objetos basándose en clases incorporadas (Button, Date, MovieClip, etc.) y en las clases e interfaces que haya creado. En el siguiente fragmento, se crea un nuevo objeto del tipo personalizado Student:

```
var firstStudent:Student = new Student();
```

También puede especificar que los objetos sean del tipo de datos Function o Void. Para más información sobre la asignación de tipos de datos, consulte el Capítulo 10, “Datos y tipos de datos”, en la página 327.

## Llaves

Las llaves ({}) permiten agrupar eventos, definiciones de clases y funciones de ActionScript en bloques. La llave inicial se coloca en la misma línea que la declaración.

| NOTA | También puede colocar la llave inicial en la línea que sigue a la declaración. Las convenciones de codificación recomiendan colocar la llave inicial en la misma línea para lograr mayor coherencia. Para más información sobre llaves y convenciones del código, consulte el Capítulo 19, “Recomendaciones y convenciones de codificación para ActionScript 2.0”, en la página 775. |
|---|---|

Coloque llaves alrededor de las sentencias cuando formen parte de una estructura de control (como, por ejemplo, `if..else` o `for`), aunque contengan una sola sentencia. Esta práctica recomendada le ayuda a evitar errores en el código ActionScript cuando se olvida de añadir llaves al código. En el siguiente ejemplo se muestra código escrito con un formato insuficiente:

```
var numUsers:Number;
if (numUsers == 0)
  trace("no users found.");
```

Aunque este código es válido, se considera que su formato es insuficiente porque las sentencias no aparecen entre llaves.

| SUGERENCIA | Si hace clic en el botón Revisar sintaxis, se añadirán llaves a esta sentencia. |
|---|---|

En este caso, si añade una segunda sentencia después de la sentencia trace, la segunda sentencia se ejecutará independientemente de que la variable `numUsers` sea igual a 0, lo que puede provocar resultados inesperados. Por este motivo, añada llaves al código como se muestra en el siguiente ejemplo:

```
var numUsers:Number;
if (numUsers == 0) {
  trace("no users found");
}
```

En el siguiente ejemplo, se crea un objeto de detector de eventos y una instancia de MovieClipLoader.

```
var imgUrl:String = "http://www.helpexamples.com/flash/images/image1.jpg";
this.createEmptyMovieClip("img_mc", 100);
var mclListener:Object = new Object();
mclListener.onLoadStart = function() {
  trace("starting");
};
mclListener.onLoadInit = function(target_mc:MovieClip):Void {
  trace("success");
};
mclListener.onLoadError = function(target_mc:MovieClip):Void {
  trace("failure");
};
var myClipl:MovieClipLoader = new MovieClipLoader();
myClipl.addListener(mclListener);
myClipl.loadClip(imgUrl, img_mc);
```

En el siguiente ejemplo se muestra un archivo de clase sencillo que podría utilizarse para crear un objeto Student. Encontrará más información sobre archivos de clases en el Capítulo 6, “Clases”, en la página 195.

### Para utilizar llaves en un archivo de ActionScript:

1. Seleccione Archivo > Nuevo y, a continuación, Archivo ActionScript.
2. Seleccione Archivo > Guardar como y guarde el nuevo documento con el nombre **Student.as**.
3. Añada el siguiente código ActionScript al archivo AS.

```
// Student.as
class Student {
  private var _id:String;
  private var _firstName:String;
  private var _middleName:String;
  private var _lastName:String;

  public function Student(id:String, firstName:String,
  middleName:String, lastName:String) {
    this._id = id;
    this._firstName = firstName;
    this._middleName = middleName;
    this._lastName = lastName;
  }
  public function get firstName():String {
    return this._firstName;
  }
  public function set firstName(value:String):Void {
    this._firstName = value;
  }
  // ...
}
```

4. Guarde el archivo de clase.
5. Seleccione Archivo > Nuevo y haga clic en Documento de Flash para crear un nuevo archivo FLA.
6. Guarde el nuevo archivo FLA como **student_test.fla**.
7. Escriba el siguiente código ActionScript en el fotograma 1 de la línea de tiempo principal:

```
// student_test.fla
import Student;
var firstStudent:Student = new Student("cst94121", "John", "H.", "Doe");
trace(firstStudent.firstName); // John
firstStudent.firstName = "Craig";
trace(firstStudent.firstName); // Craig
```

8. Seleccione Archivo > Guardar para guardar los cambios en student_test.fla.
9. Seleccione Control > Probar película para probar los archivos FLA y AS.

En el siguiente ejemplo se muestra cómo se utilizan las llaves al trabajar con funciones.

### Para utilizar llaves con funciones:

1. Seleccione Archivo > Nuevo y seleccione Documento de Flash para crear un nuevo archivo FLA.
2. Seleccione Archivo > Guardar como y asigne al nuevo archivo el nombre **checkform.fla**.
3. Arrastre una instancia del componente Label del panel Componentes al escenario.
4. Abra el inspector de propiedades (Ventana > Propiedades > Propiedades) y, con la instancia del componente Label seleccionada, escriba el nombre de instancia **status_lbl** en el cuadro de texto Nombre de instancia.
5. Escriba **200** en el cuadro de texto An (anchura) para cambiar el tamaño del componente de forma que tenga una anchura de 200 píxeles.
6. Arrastre una instancia del componente TextInput al escenario y asígnele el nombre de instancia **firstName_ti**.
7. Arrastre una instancia del componente Button al escenario y asígnele el nombre de instancia **submit_button**.
8. Seleccione el fotograma 1 de la línea de tiempo y añada el siguiente código ActionScript al panel Acciones:

```
function checkForm():Boolean {
  status_lbl.text = "";
  if (firstName_ti.text.length == 0) {
    status_lbl.text = "Please enter a first name.";
    return false;
  }
  return true;
}
function clickListener(evt_obj:Object):Void {
  var success:Boolean = checkForm();
};
submit_button.addEventListener("click", clickListener);
```

9. Seleccione Archivo > Guardar para guardar el documento de Flash.
10. Seleccione Control > Probar película para comprobar el código en el entorno de edición.

    En el archivo SWF, aparece un mensaje de error si hace clic en la instancia de Button del escenario y no hay texto en el componente TextInput `firstName_ti`. Este error aparece en el componente Label e informa a los usuarios de que deben introducir un nombre.

El ejemplo siguiente en el que se utilizan llaves muestra cómo crear y definir propiedades dentro de un objeto. En este ejemplo, las propiedades se definen en el objeto especificando los nombres de las variables entre las llaves ({}):

```
var myObject:Object = {id:"cst94121", firstName:"John", middleName:"H.",
  lastName:"Doe"};
var i:String;
for (i in myObject) {
  trace(i + ": " + myObject[i]);
}
/*
  id: cst94121
  firstName: John
  middleName: H.
  lastName: Doe
*/
```

También puede utilizar llaves vacías como métodos abreviados de sintaxis para la función `new Object()`. Por ejemplo, el código siguiente crea una instancia Object vacía:

```
var myObject:Object = {};
```

**SUGERENCIA**

Recuerde que debe asegurarse de que cada llave inicial cuente con su correspondiente llave final.

## Paréntesis

Al definir una función mediante ActionScript, los parámetros se colocan entre paréntesis [`()`], como se muestra en las siguientes líneas de código:

```
function myFunction(myName:String, myAge:Number, happy:Boolean):Void {
  // El código se escribe aquí.
}
```

Al llamar a una función, los parámetros que se desee pasar a ésta también se incluyen entre paréntesis, como se muestra en el siguiente ejemplo:

```
myFunction("Carl", 78, true);
```

Puede utilizar paréntesis para modificar el orden de precedencia de ActionScript o para hacer más legibles las sentencias de ActionScript. Esto significa que puede cambiar el orden en que se calculan los valores escribiendo entre corchetes determinados valores, como se muestra en el siguiente ejemplo:

```
var computedValue:Number = (circleClip._x + 20) * 0.8;
```

Debido al orden de precedencia, si no ha utilizado paréntesis o utiliza dos sentencias independientes, la multiplicación se calculará en primer lugar, lo que significa que la primera operación será 20 * 0.8. El resultado, 16, se sumaría a continuación al valor actual de circleClip._x y, finalmente, se asignaría a la variable `computedValue`.

Si no utiliza paréntesis, deberá añadir una sentencia para evaluar la expresión, como se muestra en el siguiente ejemplo:

```
var tempValue:Number = circleClip._x + 20;
var computedValue:Number = tempValue * 0.8;
```

Al igual que en el caso de los corchetes y las llaves, debe asegurarse de que cada paréntesis inicial cuenta con su correspondiente paréntesis final.

## Literales

Un *literal* es un valor que aparece directamente en el código. Los literales son valores constantes (que no cambian) contenidos en los documentos de Flash. Ejemplos de literales son `true`, `false`, `0`, `1`, `52` o incluso la cadena "`foo`".

Todos los ejemplos siguientes son literales:

```
17
"hello"
-3
9.4
null
undefined
true
false
```

Los literales también pueden agruparse para formar literales compuestos. Los literales de matrices se escriben entre corchetes (`[]`) y utilizan la coma (`,`) para separar los elementos de la matriz. Un literal de matriz puede utilizarse para inicializar una matriz. En los siguientes ejemplos se muestran dos matrices que se inicializan mediante literales de matriz. Puede utilizar la sentencia `new` y pasar el literal compuesto como parámetro al constructor de la clase Array, pero no puede asignar valores literales directamente al crear instancias de clases incorporadas en ActionScript.

```
// se utiliza una nueva sentencia
var myStrings:Array = new Array("alpha", "beta", "gamma");
var myNums:Array = new Array(1, 2, 3, 5, 8);

// asignar literal directamente
var myStrings:Array = ["alpha", "beta", "gamma"];
var myNums:Array = [1, 2, 3, 5, 8];
```

Los literales también pueden utilizarse para inicializar un objeto genérico. Un objeto genérico es una instancia de la clase Object. Los literales de objetos se escriben entre llaves ({}) y utilizan la coma (,) para separar las propiedades del objeto. Cada propiedad se declara mediante el signo de dos puntos (:), que separa el nombre de la propiedad del valor de la propiedad.

Puede crear un objeto genérico utilizando la sentencia `new` y pasar el literal de objeto como parámetro al constructor de la clase Object, o bien asignar el literal de objeto directamente a la instancia que está declarando. En el siguiente ejemplo se crea un nuevo objeto genérico y se inicializa el objeto con tres propiedades, `propA`, `propB` y `propC` establecidas con los valores `1`, `2` y `3` respectivamente.

```
// se utiliza una nueva sentencia
var myObject:Object = new Object({propA:1, propB:2, propC:3});

// asignar literal directamente
var myObject:Object = {propA:1, propB:2, propC:3};
```

No debe confundir un literal de cadena con un objeto String. En el siguiente ejemplo, la primera línea de código crea el literal de cadena `firstStr`, mientras que la segunda línea de código crea el objeto String `secondStr`:

```
var firstStr:String = "foo"
var secondStr:String = new String("foo")
```

Utilice literales de cadena a no ser que necesite utilizar un objeto String específicamente para lograr un mejor rendimiento. Para más información sobre cadenas, consulte "Cadenas y la clase String" en la página 480.

## Comentarios

Los comentarios son una forma de realizar anotaciones en el código con descripciones en lenguaje sencillo que el compilador no evalúa. Puede utilizar comentarios dentro del código para describir qué hace el código o qué datos se devuelven al documento. La utilización de comentarios puede ayudarle a recordar decisiones importantes de codificación y puede ser de gran ayuda para otras personas que lean el código. Los comentarios deben explicar claramente el propósito del código y no simplemente traducirlo. Si alguna parte del código no queda suficientemente clara, debería añadir comentarios.

Es muy aconsejable añadir notas a los scripts utilizando comentarios. Los comentarios documentan las decisiones que se toman con respecto al código y responden a las preguntas cómo y por qué. Hacen que el código ActionScript resulte más comprensible. Por ejemplo, podría describir una solución en los comentarios. De esta forma, usted u otro desarrollador podría localizar fácilmente partes del código para resolver errores o llevar a cabo actualizaciones. O si el problema se resuelve o se produce una mejora en una futura versión de Flash o Flash Player, podría mejorar el código ActionScript eliminando la solución implementada.

Evite utilizar comentarios saturados. Un ejemplo de comentario saturado sería el uso de una línea de signos igual (=) o asteriscos (*) para crear un bloque de separación alrededor de los comentarios. En lugar de eso, utilice un espacio en blanco para separar los comentarios del código ActionScript. Si aplica formato al código ActionScript mediante el botón Formato automático del panel Acciones o la ventana Script, se elimina el espacio en blanco. No olvide volver a añadir espacio en blanco al código o utilizar líneas de comentario sencillas (//) para mantener el espaciado; estas líneas resultan más fáciles de eliminar después de aplicar formato al código que determinar el lugar en el que previamente se encontraba el espacio en blanco.

Antes de desplegar el proyecto, elimine los comentarios superfluos del código, como "definir las variables x e y" u otros comentarios que resulten obvios de manera inmediata para otros desarrolladores. Si observa que hay demasiados comentarios sobrantes en el código ActionScript, piense si necesita volver a escribir parte del código. Si necesita incluir muchos comentarios sobre el funcionamiento del código, ello será indicativo de que el código ActionScript no es ni elegante ni intuitivo.

Si activa la aplicación de color de sintaxis, los comentarios aparecen en gris de forma predeterminada. Los comentarios pueden tener cualquier longitud sin que ello afecte al tamaño del archivo exportado y no es necesario que sigan las reglas de las palabras clave y de la sintaxis de ActionScript.

**NOTA** El uso de comentarios es especialmente importante si se va a presentar el código ActionScript a un público. Añada comentarios al código si está creando aplicaciones de ejemplo para el aprendizaje de Flash o si está escribiendo artículos o tutoriales sobre ActionScript.

## Comentarios de una sola línea

Los comentarios de una sola línea permiten añadir al código un comentario que ocupe una sola línea. Puede convertir en comentario una sola línea de código o añadir una breve descripción de lo que se consigue mediante un fragmento de código determinado. Para indicar que una línea o parte de una línea es un comentario, iníciela con dos barras inclinadas (//), como se muestra en el siguiente código:

```
// Lo que sigue establece una variable local de edad.
var myAge:Number = 26;
```

Los comentarios de una sola línea suelen utilizarse para explicar un pequeño fragmento de código. Puede utilizar comentarios de una sola línea para los comentarios breves que quepan en una línea. El siguiente ejemplo contiene un comentario de una sola línea:

```
while (condition) {
  // gestionar condición con sentencias
}
```

## Comentarios de varias líneas

Utilice comentarios de varias líneas, también llamados comentarios en bloque, para crear comentarios formados por más de una línea. Los desarrolladores utilizan habitualmente los comentarios de varias líneas para describir archivos, estructuras de datos y métodos. Normalmente se colocan al principio de un archivo y antes de un método o en él.

Para crear un bloque de comentario, coloque `/*` al principio de las líneas de comentario y `*/` al final del bloque de comentario. Esta técnica le permite crear comentarios largos sin añadir // al comienzo de cada línea. La utilización de // para varias líneas consecutivas puede originar problemas al modificar los comentarios.

El formato de un comentario de varias líneas es el siguiente.

```
/*
  El siguiente código ActionScript inicializa variables que se utilizan en
  sistemas de menú principal y submenús. Las variables se utilizan para
  realizar un seguimiento de las opciones en las que se hace clic.
*/
```

**SUGERENCIA**

Si coloca los caracteres de comentario (/* y */) en líneas separadas al principio y al final del comentario, puede anularlos fácilmente colocando dos barras diagonales (//) delante de ellos (por ejemplo, ///* y //*/). De esta manera, puede añadir comentarios al código y anularlos de forma rápida y sencilla.

Al colocar grandes trozos de script en un bloque de comentario, lo que se conoce como *convertir en comentario* una parte del script, podrá comprobar determinadas partes de un script. Por ejemplo, cuando se ejecute el siguiente script, no se ejecutará el código incluido en el bloque de comentario:

```
// El siguiente código se ejecuta.
var x:Number = 15;
var y:Number = 20;

// El siguiente código se convierte en un comentario y no se ejecuta.
/*
// crear un objeto Date nuevo
var myDate:Date = new Date();
var currentMonth:Number = myDate.getMonth();
// convertir número de mes en nombre de mes
var monthName:String = calcMonth(currentMonth);
var year:Number = myDate.getFullYear();
var currentDate:Number = myDate.getDate();
*/

// El código siguiente se ejecuta.
var namePrefix:String = "My name is";
var age:Number = 20;
```

| SUGERENCIA | Es recomendable colocar una línea en blanco delante de un comentario en bloque. |
|---|---|

## Comentarios finales

Los comentarios finales permiten añadir un comentario en la misma línea que el código. Estos comentarios aparecen en la misma línea que el código ActionScript. Los desarrolladores utilizan habitualmente los comentarios finales para indicar cuál es el contenido de una variable o para describir o llamar la atención sobre el valor devuelto por una línea de código ActionScript. Aplique formato a los comentarios finales de la siguiente forma:

```
var myAge:Number = 26; // variable de mi edad
trace(myAge); // 26
```

Inserte los comentarios a la derecha y con espacios para que los lectores puedan distinguirlos del código. Si es posible, intente alinear los comentarios, tal y como se muestra en el código siguiente.

```
var myAge:Number = 28;            // mi edad
var myCountry:String = "Canada";  // mi país
var myCoffee:String = "Hortons";  // mi café preferido
```

Si utiliza el formato automático (haga clic en el botón Formato automático del panel Acciones), los comentarios finales se desplazarán a la siguiente línea. Añada estos comentarios después de aplicar formato al código pues, de lo contrario, deberá modificar su ubicación tras utilizar el botón Formato automático.

## Comentarios dentro de clases

Los comentarios dentro de las clases e interfaces permiten documentarlas con el fin de que los desarrolladores comprendan el contenido de una clase. Puede empezar todos los archivos de clase con un comentario que indique el nombre de clase, el número de versión, la fecha y el copyright. Por ejemplo, podría crear para la clase una documentación similar al siguiente comentario:

```
/**
  Pelican class
  version 1.2
  10/10/2005
  copyright Macromedia, Inc.
*/
```

Utilice los comentarios en bloque para describir archivos, estructuras de datos y métodos. Normalmente se colocan al principio de un archivo y antes de un método o en él.

En un archivo de clase o interfaz típico hay dos tipos de comentarios: comentarios de documentación y comentarios de implementación. Los comentarios de documentación se utilizan para describir las especificaciones del código y no describen la implementación. Los comentarios de documentación se utilizan para describir interfaces, clases, métodos y constructores. Los comentarios de implementación se utilizan para marcar el código o para comentar la implementación de determinadas secciones del código.

Incluya un solo comentario de documentación por cada clase, interfaz o miembro y colóquelo inmediatamente antes de la declaración. Si debe proporcionar más información de la que cabe en los comentarios de documentación, utilice los comentarios de implementación (con el formato de comentarios en bloque o comentarios de una sola línea). Los comentarios de implementación deben ir inmediatamente después de la declaración.

Los dos tipos de comentarios utilizan delimitadores ligeramente distintos. Los comentarios de documentación se delimitan con `/**` y `*/`, y los comentarios de implementación se delimitan con `/*` y `*/`.

| SUGERENCIA | No incluya comentarios que no estén directamente relacionados con la clase que se está leyendo. Por ejemplo, no incluya comentarios que describan el paquete correspondiente. |
|---|---|

También puede incluir comentarios de una sola línea, comentarios en bloque y comentarios finales en los archivos de clases. Hallará más información acerca de estos tipos de comentarios en las secciones siguientes:

- “Comentarios de una sola línea” en la página 96
- “Comentarios de varias líneas” en la página 97
- “Comentarios finales” en la página 98

# Constantes y palabras clave

Las constantes y las palabras clave constituyen la base de la sintaxis de ActionScript. Las constantes son propiedades con un valor fijo que no puede alterarse, por lo que son valores que no cambian en ninguna parte de una aplicación.

Flash incluye diversas constantes predefinidas que contribuyen a simplificar el desarrollo de aplicaciones. Un ejemplo de constantes lo podemos encontrar en la clase Key, que incluye numerosas propiedades, como `Key.ENTER` o `Key.PGDN`. Si basa su trabajo en constantes, no tendrá que recordar nunca que los valores de código de las teclas Intro y AvPág son 13 y 34. La utilización de valores constantes no hace que el desarrollo o la depuración resulten más sencillos, pero sí permite a otros desarrolladores leer el código con mayor facilidad.

Las palabras clave de ActionScript se utilizan para realizar tipos de acciones específicos. También son palabras reservadas por este motivo, de modo que no se pueden utilizar como identificadores (por ejemplo, nombres de variables, de funciones o de etiquetas). Ejemplos de palabras reservadas son: `if`, `else`, `this`, `function` y `return`.

Para más información sobre constantes y palabras clave, consulte los siguientes temas:

- "Utilización de constantes" en la página 100
- "Palabras clave" en la página 103
- "Palabras reservadas" en la página 104

Para más información sobre objetos y propiedades, consulte "Tipo de datos Object (objeto)" en la página 335. Para obtener una lista de constantes del lenguaje (por ejemplo, `false` y `NaN`), consulte la categoría Constantes >de Elementos del lenguaje ActionScript en *Referencia del lenguaje ActionScript 2.0*.

## Utilización de constantes

Las constantes son propiedades con un valor fijo que no puede alterarse; dicho de otro modo, son valores que no cambian en ninguna parte de una aplicación. El lenguaje ActionScript contiene numerosas constantes predefinidas. Por ejemplo, las constantes `BACKSPACE`, `ENTER`, `SPACE` y `TAB` son propiedades de la clase Key y se refieren a las teclas del teclado. La constante `Key.TAB` siempre tiene el mismo significado: indica la tecla Tabulador de un teclado. Las constantes resultan útiles para comparar valores y utilizar en la aplicación valores que no cambian.

Para comprobar si el usuario está presionando la tecla Intro, utilice la siguiente sentencia:

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.getCode() == Key.ENTER) {
    trace("Are you ready to play?");
  }
};
Key.addListener(keyListener);
```

Para que funcione el anterior código ActionScript, puede que sea necesario desactivar los métodos abreviados de teclado en el entorno de edición. Seleccione Control > Probar película del menú principal y, seguidamente, mientras previsualiza el archivo SWF en el reproductor, seleccione Control > Deshabilitar métodos abreviados de teclado de la ventana de vista previa del archivo SWF.

En Flash, no existe ninguna forma de crear sus propios valores constantes, salvo si se crea clases personalizadas con variables de miembros privadas. No puede crear una variable de "sólo lectura" en Flash.

Las variables deben escribirse en minúsculas o mezclando mayúsculas y minúsculas; sin embargo, las constantes (variables que no cambian) deberían escribirse en mayúsculas. Separe las palabras con caracteres de subrayado, tal y como se muestra en el siguiente código ActionScript:

```
var BASE_URL:String = "http://www.macromedia.com"; //constante
var MAX_WIDTH:Number = 10;                         //constante
```

Escriba las constantes estáticas en mayúsculas y separe las palabras con un carácter de subrayado. No programe directamente las constantes numéricas, a menos que la constante sea 1, 0 o -1, que es la constante que utilizaría en un bucle `for` como valor de contador.

Puede utilizar las constantes en situaciones en las que es necesario referirse a una propiedad cuyo valor es invariable. De esta forma será más sencillo encontrar errores tipográficos en el código que podrían pasar inadvertidos si utilizara literales. También podrá modificar el valor en un solo lugar. Para más información sobre el uso de literales, consulte "Literales" en la página 94.

Por ejemplo, la definición de clase del siguiente ejemplo crea tres constantes que siguen la convención de denominación empleada por ActionScript 2.0.

### Para utilizar constantes en una aplicación:

1. Seleccione Archivo > Nuevo y elija Archivo ActionScript para crear un archivo AS.
2. Asigne al nuevo archivo el nombre **ConstExample.as**.
3. Escriba el siguiente código en la ventana Script:

```
class ConstExample {
  public static var EXAMPLE_STATIC:String = "Global access";
  public var EXAMPLE_PUBLIC:String = "Public access";
  private var EXAMPLE_PRIVATE:String = "Class access";
}
```

   La propiedad `EXAMPLE_STATIC` es estática, lo que significa que la propiedad afecta a la clase en su conjunto y no sólo a una instancia concreta de la clase. Para acceder a una propiedad estática de una clase, debe utilizar el nombre de la clase en lugar del nombre de una instancia. No es posible acceder a una propiedad estática a través de una instancia de clase.
4. Cree un nuevo documento de Flash y guárdelo como **const.fla**.
5. Abra el panel Acciones y escriba el siguiente código en el fotograma 1 de la línea de tiempo:

```
trace(ConstExample.EXAMPLE_STATIC); // salida: acceso global
```

   Al declarar la propiedad `EXAMPLE_STATIC` como estática, este código permite acceder al valor de la propiedad.
6. Seleccione Control > Probar película para probar el documento.

   Observará `Global access` en el panel Salida.
7. En el panel Acciones, escriba este código tras el código añadido en el paso 5.

```
trace(ConstExample.EXAMPLE_PUBLIC); // error
trace(ConstExample.EXAMPLE_PRIVATE); // error
```

8. Seleccione Control > Probar película para probar el documento.

   Las propiedades `EXAMPLE_PUBLIC` y `EXAMPLE_PRIVATE` no son propiedades estáticas. Cuando intente acceder a sus valores a través de la clase, verá el siguiente error:

```
The property being referenced does not have the static attribute.
```

   Para acceder a una propiedad que no sea estática, deberá acceder al valor a través de una instancia de la clase. Dado que la propiedad `EXAMPLE_PUBLIC` es una propiedad pública, está disponible para el código situado fuera de la definición de clase.
9. En el panel Acciones, elimine las sentencias `trace` que añadió en los pasos 5 y 7.
10. Introduzca el código siguiente en el panel Acciones:

```
var myExample:ConstExample = new ConstExample();
trace(myExample.EXAMPLE_PUBLIC); // salida: Public access
```

    Este código crea la instancia myExample y accede a la propiedad `EXAMPLE_PUBLIC`.

11. Seleccione Control > Probar película para probar el documento.

    Observará `Public access` en el panel Salida.

12. En el panel Acciones, elimine la sentencia trace que añadió en el paso 10.

13. Introduzca el código siguiente en el panel Acciones.

```
trace(myExample.EXAMPLE_PRIVATE); // error
```

    La propiedad `EXAMPLE_PRIVATE` es una propiedad privada, por lo que sólo está disponible dentro de la definición de clase.

14. Seleccione Control > Probar película para probar el documento.

    Verá `El miembro de clase es privado y no permite el acceso` en el panel Salida.

    Para más información sobre clases incorporadas y la creación de clases personalizadas, consulte el Capítulo 6, “Clases”, en la página 195.

## Palabras clave

Las palabras clave son palabras que hacen algo concreto en ActionScript. Por ejemplo, la palabra clave `var` se utiliza para declarar una variable. La palabra clave `var` se muestra en la siguiente línea de código:

```
var myAge:Number = 26;
```

Una palabra clave es una palabra reservada que tiene un significado específico: por ejemplo, la palabra clave `class` se utiliza para definir una nueva clase de ActionScript, mientras que la palabra clave `var` se utiliza para declarar variables locales. Otros ejemplos de palabras reservadas son: `if`, `else`, `this`, `function` y `return`.

Las palabras clave no pueden utilizarse como identificadores (como, por ejemplo, nombres de variables, de funciones o de etiquetas), por lo que no deberá utilizarlas en ningún otro lugar de los archivos FLA para ningún otro fin (por ejemplo, como nombres de instancias). Ya ha utilizado en numerosas ocasiones la palabra clave `var`, especialmente si ha leído el Capítulo 10, “Datos y tipos de datos”, en la página 327. ActionScript reserva palabras para usarlas específicamente en su lenguaje. Por consiguiente, no podrá utilizar las palabras clave como identificadores (por ejemplo, nombres de variables, de funciones o de etiquetas). Encontrará una lista de estas palabras clave en “Palabras reservadas” en la página 104.

# Palabras reservadas

Las *palabras reservadas* son aquellas que no puede utilizar como identificadores en el código porque su uso está reservado a ActionScript. Entre las palabras reservadas se encuentran las *palabras clave*, que son sentencias en ActionScript, y palabras que están reservadas para su uso en el futuro. Esto significa que no debe utilizarlas como nombres de variables, instancias, clases personalizadas, etc.; si lo hace, podrían producirse problemas técnicos en el código.

En la siguiente tabla se enumeran las palabras clave reservadas en Flash que provocan errores en los scripts:

| | | | |
|---|---|---|---|
| add | and | break | case |
| catch | class | continue | default |
| delete | do | dynamic | else |
| eq | extends | finally | for |
| function | ge | get | gt |
| if | ifFrameLoaded | implements | import |
| in | instanceof | interface | intrinsic |
| le | lt | ne | new |
| not | on | onClipEvent | or |
| private | public | return | set |
| static | switch | tellTarget | this |
| throw | try | typeof | var |
| Void | while | with | |

En la siguiente tabla se enumeran las palabras clave que están reservadas para su uso en el futuro por parte de ActionScript o el borrador de especificación del lenguaje ECMAScript (ECMA-262) edición 4. También debe evitar utilizar estas palabras clave en el código:

| | | | |
|---|---|---|---|
| abstract | enum | export | short |
| byte | long | synchronized | char |
| debugger | protected | double | volatile |
| float | throws | transient | goto |

Todos los nombres de clases incorporadas, nombres de clases de componentes y nombres de interfaces son clases reservadas y no deben utilizarse como identificadores en el código:

| | | | |
|---|---|---|---|
| Accessibility | Accordion | Alert | Array |
| Binding | Boolean | Button | Camera |
| CellRenderer | CheckBox | Collection | Color |
| ComboBox | ComponentMixins | ContextMenu | ContextMenuItem |
| CustomActions | CustomFormatter | CustomValidator | DataGrid |
| DataHolder | DataProvider | DataSet | DataType |
| Date | DateChooser | DateField | Delta |
| DeltaItem | DeltaPacket | DepthManager | EndPoint |
| Error | FocusManager | Form | Function |
| Iterator | Key | Label | List |
| Loader | LoadVars | LocalConnection | Log |
| Math | Media | Menu | MenuBar |
| Microphone | Mouse | MovieClip | MovieClipLoader |
| NetConnection | NetStream | Number | NumericStepper |
| Object | PendingCall | PopUpManager | PrintJob |
| ProgressBar | RadioButton | RDBMSResolver | Screen |
| ScrollPane | Selección | SharedObject | Slide |
| SOAPCall | Sound | Stage | String |
| StyleManager | System | TextArea | TextField |
| TextFormat | TextInput | TextSnapshot | TransferObject |
| Tree | TreeDataProvider | TypedValue | UIComponent |
| UIEventDispatcher | UIObject | Video | WebService |
| WebServiceConnector | Window | XML | XMLConnector |
| XUpdateResolver | | | |

Existen varias palabras que, aunque no son palabras reservadas, no deben utilizarse como identificadores en el código ActionScript (por ejemplo, como nombres de variables o instancias). Hay palabras que son utilizadas por las clases incorporadas que conforman el lenguaje ActionScript. Por consiguiente, no debe utilizar los nombres de propiedades, métodos, clases, interfaces, nombres de clases de componentes y valores como nombres en el código (por ejemplo, al asignar nombre a las variables, clases o instancias).

Para conocer cuáles son estos nombres, consulte *Referencia del lenguaje ActionScript 2.0* y busque en el panel Ayuda las secciones de uso de *Aprendizaje de ActionScript 2.0 en Flash*.

# Sentencias

Una *sentencia* es una instrucción que se da al archivo FLA para que haga algo, como, por ejemplo, ejecutar una acción concreta. Por ejemplo, puede utilizar una sentencia condicional para determinar si algo es verdadero o si existe. Posteriormente, podría ejecutar las acciones que especifique, como, por ejemplo, funciones o expresiones, en función de si la condición es verdadera o no.

Por ejemplo, la sentencia `if` es una sentencia condicional que evalúa una condición para determinar la siguiente acción que debe tener lugar en el código.

```
// sentencia if
if (condition) {
  // sentencias;
}
```

Otro ejemplo es la sentencia `return`, que devuelve un resultado como valor de la función en la que se ejecuta.

Hay muchas formas de escribir o aplicar formato a un código ActionScript. Puede que su forma de construir la sintaxis del código ActionScript difiera de la de otros usuarios, por ejemplo, en lo que se refiere al espaciado de las sentencias o al lugar en el que coloca llaves (`{}`) en el código. Aunque existen diversas formas de construir las sentencias sin romper el código, puede seguir algunas directrices para escribir código ActionScript bien formado.

**Coloque una sola sentencia por línea para aumentar la legibilidad del código ActionScript.** En el siguiente ejemplo se muestra el uso de sentencias recomendado y no recomendado:

```
theNum++;       // recomendado
theOtherNum++;  // recomendado
aNum++; anOtherNum++;  // no recomendado
```

**Asigne las variables como sentencias independientes.** Observe el siguiente ejemplo de código ActionScript:

```
var myNum:Number = (a = b + c) + d;
```

Este código ActionScript incorpora una asignación dentro del código que resulta difícil de leer. La legibilidad mejora si se asignan variables como sentencias independientes, como se muestra en el ejemplo siguiente:

```
var a:Number = b + c;
var myNum:Number = a + d;
```

En las siguientes secciones se muestra cómo construir sentencias específicas en ActionScript. Para información sobre la escritura y aplicación de formato a eventos, consulte el Capítulo 9, “Gestión de eventos”, en la página 305.

Para más información sobre cada sentencia, consulte los siguientes temas:

- "Sentencias compuestas" en la página 107
- "Condiciones" en la página 107
- "Repetición de acciones mediante bucles" en la página 118

## Sentencias compuestas

Una sentencia compuesta contiene numerosas sentencias que colocan entre llaves ({}). Las sentencias situadas dentro de una sentencia compuesta pueden ser de cualquier tipo de sentencias de ActionScript. A continuación se muestra una sentencia compuesta típica.

Las sentencias entre llaves aparecen con sangría con respecto a la sentencia compuesta, como se muestra en el siguiente código ActionScript:

```
var a:Number = 10;
var b:Number = 10;
if (a == b) {
  // Este código aparece sangrado.
  trace("a == b");
  trace(a);
  trace(b);
}
```

Esta sentencia compuesta contiene varias sentencias, pero actúa como una sola sentencia en el código ActionScript. La llave de apertura se coloca al final de la sentencia compuesta. La llave de cierre empieza una línea y se alinea con el principio de la sentencia compuesta.

Para más información sobre el uso de llaves, consulte "Llaves" en la página 89.

## Condiciones

Las condiciones permiten determinar si algo es verdadero o existe y, seguidamente, repetir opcionalmente una acción (mediante bucles) o ejecutar las acciones que se especifiquen, como, por ejemplo, funciones o expresiones, en función de si la condición es verdadera o no lo es. Por ejemplo, puede determinar si una determinada variable está definida o tiene un valor concreto y ejecutar un bloque de código en función del resultado. También puede cambiar los gráficos del documento de Flash dependiendo de la hora que marque el reloj del sistema del usuario o del tiempo existente en el lugar actual de éste.

Para realizar una acción en función de si se da o no una condición, o para repetir una acción (crear sentencias de bucle), puede utilizar las sentencias `if`, `else`, `else if`, `for`, `while`, `do while`, `for..in` o `switch`.

Para más información sobre condiciones que puede utilizar y cómo escribirlas, consulte los siguientes temas:



## Escritura de condiciones

Las sentencias que comprueban si una condición es verdadera (true) o falsa (false) comienzan con el término `if`. Si la condición da como resultado `true`, ActionScript ejecuta la siguiente sentencia. Si el resultado de la condición es `false`, ActionScript salta a la siguiente sentencia fuera del bloque de código.

| SUGERENCIA | Para optimizar el rendimiento de su código, compruebe primero las condiciones más probables. |
|---|---|

Las sentencias siguientes comprueban tres condiciones. El término `else if` especifica comprobaciones alternativas que se deberán llevar a cabo si las condiciones anteriores son falsas.

```
if ((passwordTxt.text.length == 0) || (emailTxt.text.length == 0)) {
  gotoAndStop("invalidLogin");
}  else if (passwordTxt.text == userID){
  gotoAndPlay("startProgram");
}
```

En este fragmento de código, si la longitud de los campos de texto passwordTxt o emailTxt es 0 (por ejemplo, si el usuario no ha introducido ningún valor), el documento de Flash se redirige a la etiqueta de fotograma `invalidLogin`. Si ambos campos de texto, passwordTxt y emailTxt, contienen valores y el contenido del campo de texto passwordTxt coincide con la variable `userID`, el archivo SWF se redirige a la etiqueta de fotograma `startProgram`.

Si desea comprobar una entre varias condiciones, puede utilizar la sentencia `switch` en lugar de varias sentencias `else if`. Para más información sobre sentencias `switch`, consulte "Utilización de una sentencia switch" en la página 112.

Consulte las siguientes secciones para aprender a escribir diferentes tipos de condiciones en las aplicaciones ActionScript.

## Utilización de la sentencia if

Utilice la sentencia `if` cuando desee ejecutar una serie de sentencias en función de si una determinada condición es verdadera (true) o no lo es.

```
// sentencia if
if (condition) {
  // sentencias;
}
```

Existen diversas situaciones en las que deberá utilizar sentencias `if` cuando trabaje en un proyecto de Flash. Por ejemplo, si está creando un sitio con Flash que exige que los usuarios inicien una sesión antes de acceder a determinadas secciones del sitio Web, puede utilizar una sentencia `if` para validar si los usuarios han introducido texto en los campos de nombre de usuario y contraseña.

Si necesita validar nombres de usuario y contraseñas utilizando una base de datos externa, probablemente desee verificar que la combinación de nombre de usuario/contraseña enviada por un usuario coincide con un registro de la base de datos. También puede comprobar si el usuario tiene permiso para acceder a la parte del sitio especificada.

Si crea scripts de animaciones en Flash, puede que desee utilizar la sentencia `if` para comprobar si una instancia del escenario continúa estando dentro de los límites de éste. Por ejemplo, si una pelota se mueve hacia abajo a lo largo del eje vertical, puede que sea necesario detectar cuándo la pelota entra en contacto con el borde inferior del escenario para cambiar su dirección de forma que parezca que rebota hacia arriba.

### Para utilizar una sentencia if:

1. Seleccione Archivo > Nuevo y seleccione Documento de Flash.
2. Seleccione el fotograma 1 de la línea de tiempo y escriba el siguiente código ActionScript en el panel Acciones:

```
// crea una cadena que contiene AM y PM
var amPm:String = "AM";
// no se pasan parámetros a Date, por lo que devuelve la fecha y la hora
  actuales
var current_date:Date = new Date();
// si la hora actual es posterior o igual a 12, la cadena amPm se
  establece con el valor "PM".
if (current_date.getHours() >= 12) {
  amPm = "PM";
}
trace(amPm);
```

3. Seleccione Control > Probar película para probar el código ActionScript.

   En este código, se crea una cadena que contiene `AM` o `PM` en función de la hora del día. Si la hora actual es posterior o igual a 12, la cadena `amPM` se establece con el valor `PM`. Finalmente, se traza la cadena `amPm` y, si la hora es posterior o igual a 12, se muestra `PM`. En caso contrario, verá `AM`.

## Utilización de la sentencia if..else

La sentencia condicional `if..else` le permite comprobar una condición y, seguidamente, ejecutar un bloque de código si dicha condición existe, o ejecutar un bloque de código alternativo si dicha condición no existe.

Por ejemplo, el siguiente código comprueba si el valor de `x` es superior a 20 y genera una sentencia `trace()` en caso afirmativo o genera una sentencia `trace()` diferente en caso negativo:

```
if (x > 20) {
  trace("x is > 20");
} else {
  trace("x is <= 20");
}
```

Si no desea ejecutar un bloque de código alternativo, puede utilizar la sentencia `if` sin la sentencia `else`.

En Flash, la sentencia `if..else` es similar a la sentencia `if`. Por ejemplo, si utiliza la sentencia `if` para validar si un usuario ha proporcionado un nombre de usuario y una contraseña que coincida con un valor almacenado en una base de datos, puede que desee redirigir al usuario en función de si el nombre de usuario y la contraseña son correctos. Si el inicio de sesión es válido, puede redirigir al usuario a una página de bienvenida mediante el bloque `if`. No obstante, si el inicio de sesión no es válido, puede redirigir al usuario al formulario de inicio de sesión o mostrar un mensaje de error mediante el bloque `else`.

### Para utilizar una sentencia if..else en un documento:

1. Seleccione Archivo > Nuevo y, a continuación, seleccione Documento de Flash para crear un nuevo archivo FLA.
2. Seleccione el fotograma 1 de la línea de tiempo y escriba el siguiente código ActionScript en el panel Acciones:

```
// crea una cadena que contiene AM/PM dependiendo de la hora del día.
var amPm:String;
// no se pasan parámetros a Date, por lo que devuelve la fecha y la hora
  actuales.
var current_date:Date = new Date();
// si la hora actual es posterior o igual a 12, la cadena amPm se
  establece con el valor "PM".
if (current_date.getHours() >= 12) {
  amPm = "PM";
} else {
  amPm = "AM";
}
trace(amPm);
```

3. Seleccione Control > Probar película para probar el código ActionScript.

   En este código, se crea una cadena que contiene `AM` o `PM` en función de la hora del día. Si la hora actual es posterior o igual a 12, la cadena `amPM` se establece con el valor `PM`. Finalmente, se traza la cadena `amPm` y, si la hora es posterior o igual a 12, se muestra `PM`. En caso contrario, observará `AM` en el panel Salida.

## Utilización de la sentencia if..else if

Puede comprobar varias condiciones utilizando la sentencia condicional `if..else if`. Para ello, deberá utilizar la siguiente sintaxis en una sentencia `if..else if`:

```
// sentencia else-if
if (condition) {
  // sentencias;
} else if (condition) {
  // sentencias;
} else {
  // sentencias;
}
```

Deberá utilizar un bloque `if..else if` en los proyectos de Flash cuando desee comprobar una serie de condiciones. Por ejemplo, si desea mostrar una imagen diferente en la pantalla dependiendo de la hora del día a la que se produzca la visita del usuario, puede crear una serie de sentencias `if` que determinen si es por la mañana, la hora de la sobremesa, por la tarde o por la noche. Seguidamente puede mostrar un gráfico que resulte adecuado.

El siguiente código no sólo comprueba si el valor de `x` es superior a 20, sino que también comprueba si el valor de `x` es negativo:

```
if (x > 20) {
  trace("x is > 20");
} else if (x < 0) {
  trace("x is negative");
}
```

### Para utilizar una sentencia if..else if en un documento:

1. Seleccione Archivo > Nuevo y seleccione Documento de Flash.
2. Seleccione el fotograma 1 de la línea de tiempo y escriba el siguiente código ActionScript en el panel Acciones:

```
var now_date:Date = new Date();
var currentHour:Number = now_date.getHours();
// si la hora actual es anterior a 11AM...
if (currentHour < 11) {
  trace("Good morning");
  // else..if si la hora actual es anterior a 3PM...
```

```
} else if (currentHour < 15) {
  trace("Good afternoon");
  // else..if si la hora actual es anterior a 8PM...
} else if (currentHour < 20) {
  trace("Good evening");
  // else en caso de que la hora actual sea de 8PM a 11:59PM
} else {
  trace("Good night");
}
```

3. Seleccione Control > Probar película para probar el código ActionScript.

   En este código, se crea una cadena denominada `currentHour` que contiene el número de la hora actual (por ejemplo, si son las 6:19 pm, `currentHour` contendrá el número `18`). El método `getHours()` de la clase Date permite obtener la hora actual. Seguidamente, puede utilizar la sentencia `if..else if` para trazar la información en el panel Salida dependiendo del número devuelto. Para más información, consulte los comentarios del fragmento de código anterior.

## Utilización de una sentencia switch

La sentencia `switch` crea una estructura ramificada para sentencias de ActionScript. Al igual que la sentencia `if`, la sentencia `switch` prueba una condición y ejecuta sentencias si la condición devuelve un valor `true`.

Cuando se utiliza en una sentencia `switch`, la sentencia `break` ordena a Flash que omita el resto de sentencias existentes en ese bloque case y que salte a la primera sentencia que vaya a continuación de la sentencia `switch`. Si un bloque case no contiene una sentencia break, se producirá una condición conocida como de "paso al siguiente caso". En esta situación, la siguiente sentencia case también se ejecuta hasta que se encuentra una sentencia break o termina la sentencia `switch`. Este comportamiento se muestra en el siguiente ejemplo, en el que la primera sentencia case no contiene ninguna sentencia break y, por consiguiente, se ejecutan ambos bloques de código para los dos primeros casos (A y B).

Todas las sentencias `switch` deben incluir un caso `default` (predeterminado). El caso `default` siempre debe ser el último caso de una sentencia `switch` y también debe incluir una sentencia `break` para evitar un error de paso al siguiente caso si se añade otro caso. Por ejemplo, si la condición del siguiente ejemplo da como resultado A, se ejecutarán las sentencias de los casos A y B porque el caso A carece de sentencia `break`. Cuando se pasa al siguiente caso, no se incluye una sentencia break sino un comentario en lugar de la sentencia `break`, como puede apreciarse en el siguiente ejemplo, después de case A. Utilice el siguiente formato al escribir sentencias `switch`:

```
switch (condition) {
case A :
  // sentencias
  // pasa al siguiente caso
case B :
  // sentencias
  break;
case Z :
  // sentencias
  break;
default :
  // sentencias
  break;
}
```

### Para utilizar una sentencia switch en un documento:

1. Seleccione Archivo > Nuevo y seleccione Documento de Flash.
2. Seleccione el fotograma 1 de la línea de tiempo y escriba el siguiente código ActionScript en el panel Acciones:

```
var listenerObj:Object = new Object();
listenerObj.onKeyDown = function() {
  // Utiliza el método String.fromCharCode() para devolver una cadena.
  switch (String.fromCharCode(Key.getAscii())) {
  case "A" :
    trace("you pressed A");
    break;
  case "a" :
    trace("you pressed a");
    break;
  case "E" :
  case "e" :
    /* E no tiene una sentencia break, por lo que este bloque se ejecuta
  si se pulsa e o E. */
    trace("you pressed E or e");
    break;
  case "I" :
  case "i" :
    trace("you pressed I or i");
    break;
  default :
    /* Si la tecla pulsada no está reflejada en ninguno de los casos
  anteriores, aquí se ejecuta el caso predeterminado (default). */
    trace("you pressed some other key");
  }
};
Key.addListener(listenerObj);
```

3. Seleccione Control > Probar película para probar el código ActionScript.

   Escriba letras mediante el teclado, incluidas las teclas a, e o i. Al pulsar estas tres teclas, observará sentencias `trace` en el código ActionScript anterior. La línea de código crea un nuevo objeto que se utiliza como detector de la clase Key. Este objeto se utiliza para notificar el evento `onKeyDown()` cuando el usuario pulsa una tecla. El método `Key.getAscii()` devuelve el código ASCII de la última tecla que pulsa o suelta el usuario, por lo que debe utilizar el método `String.fromCharCode()` para devolver una cadena que contenga los caracteres representados por los valores ASCII de los parámetros. Dado que "E" no tiene sentencia `break`, el bloque se ejecuta si el usuario pulsa las teclas *e* o *E*. Si el usuario pulsa una tecla que no está reflejada en ninguno de los tres primeros casos, se ejecuta el caso predeterminado (default).

## Utilización de sentencias try..catch y try..catch..finally

La utilización de bloques `try..catch..finally` le permite añadir gestión de errores a las aplicaciones Flash. Las palabras clave `try..catch..finally` le permiten colocar un bloque de código en el lugar en el que puede producirse un error y responder ante dicho error. Si cualquier parte del código contenido en el bloque de código `try` devuelve un error (empleando la sentencia `throw`), el control pasa al bloque `catch`, en el caso de que exista. El control pasa al bloque de código `finally`, en el caso de que exista. El bloque `finally` opcional se ejecuta siempre, independientemente de que se haya emitido o no un error.

Si el código del bloque `try` no emite un error (es decir, si el bloque `try` se completa con normalidad), el código continúa ejecutándose por el bloque `finally`.

| NOTA | El bloque finally se ejecuta aunque el bloque try finalice con una sentencia return. |
|---|---|

Las sentencias `try..catch` y `try..catch..finally` se escriben con el siguiente formato:

```
//try-catch
try {
  // sentencias
} catch (myError) {
  // sentencias
}

// try-catch-finally
try {
  // sentencias
} catch (myError) {
  // sentencias
} finally {
  // sentencias
}
```

En el momento en que el código emita un error, podrá escribir controladores personalizados para gestionar el error de forma elegante adoptar las medidas más adecuadas. Puede que tenga que intentar cargar datos externos de un servicio Web o un archivo de texto o mostrar un mensaje de error para el usuario final. Puede incluso utilizar el bloque `catch` para intentar conectar con un servicio Web que avise al administrador de que se ha producido un error concreto y permita a éste asegurarse de que la aplicación funciona correctamente.

### Para utilizar el bloque try..catch..finally con el fin de validar datos antes de dividir algunos números:

1. Seleccione Archivo > Nuevo y seleccione Documento de Flash.
2. Seleccione el fotograma 1 de la línea de tiempo y escriba el siguiente código ActionScript en el panel Acciones:

```
var n1:Number = 7;
var n2:Number = 0;
try {
  if (n2 == 0) {
    throw new Error("Unable to divide by zero");
  }
  trace(n1/n2);
} catch (err:Error) {
  trace("ERROR! " + err.toString());
} finally {
  delete n1;
  delete n2;
}
```

3. Seleccione Control > Probar película para probar el documento.
4. El panel Salida muestra `Unable to divide by zero`.
5. Regrese al entorno de edición y cambie la siguiente línea de código:

```
var n2:Number = 0;
```

a

```
var n2:Number = 2;
```

6. Seleccione Control > Entrar para probar el documento de nuevo.

   Si el valor de `n2` es igual a cero, se emite un error que es recogido por el bloque `catch`, que a su vez muestra un mensaje en el panel Salida. Si el valor de `y` no es igual a cero, el panel Salida muestra el resultado de `n1` dividido por `n2`. El bloque `finally` se ejecuta con independencia de si se produce un error y elimina los valores de las variables `n1` y `n2` del documento de Flash.

No sólo existe la posibilidad de emitir nuevas instancias de la clase Error cuando se produce un error, sino que también puede ampliar la clase Error para crear errores personalizados, como se muestra en el siguiente ejemplo.

### Para crear un error personalizado:

1. Seleccione Archivo > Nuevo y cree un archivo de ActionScript nuevo.
2. Seleccione Archivo > Guardar como y asigne al archivo el nombre **DivideByZeroException.as**.
3. Escriba el siguiente código ActionScript en el panel Script:

```
// En DivideByZeroException.as:
class DivideByZeroException extends Error {
   var message:String = "Divide By Zero error";
}
```

4. Guarde el archivo de ActionScript.
5. Cree un nuevo documento de Flash denominado **exception_test.fla** en el mismo directorio que el archivo de ActionScript y luego guarde el archivo.
6. Escriba el siguiente código ActionScript en el panel Acciones en el fotograma 1 de la línea de tiempo principal:

```
var n1:Number = 7;
var n2:Number = 0;
try {
  if (n2 == 0) {
    throw new DivideByZeroException();
  } else if (n2 < 0) {
    throw new Error("n2 cannot be less than zero");
  } else {
    trace(n1/n2);
  }
} catch (err:DivideByZeroException) {
  trace(err.toString());
} catch (err:Error) {
  trace("An unknown error occurred; " + err.toString());
}
```

7. Guarde el documento de Flash y seleccione Control > Probar película para comprobar el archivo en el entorno de prueba.

   Dado que el valor de `n2` es igual a 0, Flash emite su clase de error personalizada DivideByZeroException y muestra `Divide By Zero error` en el panel Salida. Si cambia el valor de `n2` en la línea dos de `0` a `-1` y vuelve a probar el documento de Flash, observará `An unknown error occurred; n2 cannot be less than zero` en el panel Salida. La configuración del valor de `n2` con cualquier número mayor que `0` hace que el resultado de la división aparezca en el panel Salida. Para más información sobre la creación de clases personalizadas, consulte el Capítulo 6, “Clases”, en la página 195.

## El operador condicional y sintaxis alternativa

Si le gustan los atajos, puede utilizar el operador (`?:`) condicional, también conocido como *expresión condicional*. El operador condicional permite convertir sentencias `if..else` sencillas en una sola línea de código. El operador ayuda a reducir la cantidad de código que debe escribir para lograr el mismo objetivo, aunque también tiende a hacer más difícil la lectura del código ActionScript.

La siguiente condición, que se escribe sin abreviar, comprueba si la variable `numTwo` es mayor que cero y devuelve el resultado de `numOne/numTwo` o la cadena `carrot`:

```
var numOne:Number = 8;
var numTwo:Number = 5;
if (numTwo > 0) {
  trace(numOne / numTwo); // 1.6
} else {
  trace("carrot");
}
```

Mediante una expresión condicional, podría escribir el mismo código empleando este formato:

```
var numOne:Number = 8;
var numTwo:Number = 0;
trace((numTwo > 0) ? numOne/numTwo : "carrot");
```

Como puede observar, la sintaxis abreviada reduce la legibilidad, por lo que no es la opción preferida. Si tiene que utilizar operadores condicionales, coloque entre paréntesis la condición inicial (antes del signo de interrogación [`?`]). De esta forma mejorará la legibilidad del código ActionScript. El código siguiente es un ejemplo de código ActionScript con legibilidad mejorada:

```
var numOne:Number;
(numOne >= 5) ? numOne : -numOne;
```

Puede escribir una sentencia condicional que devuelva un valor booleano, como se muestra en el ejemplo siguiente:

```
if (cartArr.length > 0) {
  return true;
} else {
  return false;
}
```

Sin embargo, comparado con el código anterior, se prefiere el código ActionScript del siguiente ejemplo:

```
return (cartArr.length > 0);
```

El segundo fragmento es más corto y tiene menos expresiones para evaluar. Es más fácil leerlo y entenderlo.

Si se escriben condiciones complejas, es aconsejable agrupar las condiciones con el uso de paréntesis [`()`]. Si no se utilizan paréntesis, el desarrollador u otras personas que trabajen con el código ActionScript pueden encontrar errores de precedencia del operador. Para más información sobre la precedencia de los operadores, consulte "Precedencia y asociatividad de operadores" en la página 145.

Por ejemplo, en el código siguiente la condición no se escribe entre paréntesis:

```
if (fruit == "apple" && veggie == "leek") {}
```

En el código siguiente se utiliza el formato correcto, pues las condiciones se escriben entre paréntesis:

```
if ((fruit == "apple") && (veggie == "leek")) {}
```

## Repetición de acciones mediante bucles

ActionScript puede repetir una acción un número especificado de veces o mientras se dé una condición. Los bucles le permiten repetir una serie de sentencias cuando una condición concreta es `true`. Existen cuatro tipos de bucles en ActionScript: los bucles `for`, los bucles `for..in`, los bucles `while` y los bucles `do..while`. Cada tipo de bucle se comporta de forma ligeramente distinta, por lo que cada uno de ellos resulta útil en una situación diferente.

La mayoría de las reproducciones indefinidas utilizan algún tipo de contador para controlar cuántas veces se ejecutan. Cada ejecución se denomina *repetición*. Puede declarar una variable y escribir una sentencia que incremente o disminuya el valor de la variable cada vez que se ejecute la reproducción. En la acción `for`, el contador y la sentencia que incrementa el contador son parte de la acción.

| Loop | Descripción |
|---|---|
| Bucles `for` | Repite una acción mediante un contador incorporado. |
| Bucles `for..in` | Repite los subniveles de un clip de película o un objeto. |
| Bucles `while` | Repite una acción mientras se cumpla una condición. |
| Bucles `do..while` | Iguales que los bucles while, con la diferencia de que la expresión se evalúa en la parte inferior del bloque de código, con el que el bucle siempre se ejecuta al menos una vez. |

Los bucles más habituales son los bucles `for`, que reproduce un bloque de código un número preestablecido de veces. Por ejemplo, si tiene una matriz de elemento y desea ejecutar una serie de sentencias en cada elemento de la matriz, deberá utilizar un bucle `for` y que dicho bucle se repita de 0 hasta el número de elementos de la matriz. Otro tipo es el bucle `for..in`, que puede resultar de gran ayuda si se desea ejecutar por cada par nombre/valor contenido en un objeto y que luego realice algún tipo de acción. Al depurar proyectos de Flash, esto puede ser muy útil si desea ver los valores que se cargan de fuentes externas, como, por ejemplo, servicios Web o archivos de texto/XML externos. Los dos últimos tipos de bucles (`while` y `do..while`) son útiles si desea reproducir en bucle una serie de sentencias sin conocer necesariamente el número de veces que deben reproducirse. En este caso, puede utilizar un bucle `while`, que se reproduce en tanto en cuento una determinada condición sea verdadera (true).

ActionScript puede repetir una acción un número especificado de veces o mientras se dé una condición. Utilice las acciones `while`, `do..while`, `for` y `for..in` para crear reproducciones indefinidas. En esta sección se incluye información general sobre estos bucles. Consulte los siguientes procedimientos para obtener más información sobre cada uno de estos bucles.

### Para repetir una acción mientras se cumpla una condición:

- Utilice la sentencia `while`.

  Un bucle `while` calcula una expresión y ejecuta el código en el cuerpo del bucle si la expresión es `true`. Tras ejecutar cada una de las sentencias del bucle, se calcula de nuevo el resultado de la expresión. En el siguiente ejemplo, la reproducción se ejecuta cuatro veces:

  ```
  var i:Number = 4;
  while (i>0) {
    myClip.duplicateMovieClip("newMC" + i, i, {_x:i*20, _y:i*20});
    i--;
  }
  ```

  Puede utilizar la sentencia `do...while` para crear el mismo tipo de reproducción indefinida que con un bucle `while`. En un bucle `do...while`, el resultado de la expresión se calcula al final del bloque de código, de modo que el bucle siempre se ejecuta al menos una vez.

  Esto se ilustra en el siguiente ejemplo:

  ```
  var i:Number = 4;
  do {
    myClip.duplicateMovieClip("newMC" + i, i, {_x:i*20, _y:i*20});
    i--;
  } while (i>0);
  ```

  Para más información sobre la sentencia `while`, consulte “Utilización de bucles while” en la página 126.

### Para repetir una acción mediante un contador incorporado:

- Utilice la sentencia `for`.

  La mayoría de las reproducciones indefinidas utilizan algún tipo de contador para controlar cuántas veces se ejecutan. Cada ejecución se denomina *repetición*. Puede declarar una variable y escribir una sentencia que incremente o disminuya el valor de la variable cada vez que se ejecute la reproducción. En la acción `for`, el contador y la sentencia que incrementa el contador son parte de la acción.

  En el ejemplo siguiente, la primera expresión (`var i:Number = 4`) es la expresión inicial cuyo resultado se calcula antes de la primera repetición. La segunda expresión ( `i > 0`) es la condición que se comprueba antes de que se ejecute la reproducción indefinida. La tercera expresión (`i--`) se denomina *expresión posterior* y su resultado siempre se calcula después de que se ejecute la reproducción.

  ```
  for (var i:Number = 4; i > 0; i--) {
    myClip.duplicateMovieClip("newMC" + i, i, {_x:i*20, _y:i*20});
  }
  ```

  Para más información sobre la sentencia `for`, consulte "Utilización de bucles for" en la página 123.

### Para reproducir indefinidamente los subniveles de un clip de película o un objeto:

- Utilice la sentencia `for..in`.

  Los subniveles incluyen otros clips de película, funciones, objetos y variables. El ejemplo siguiente utiliza la sentencia `trace` para imprimir el resultado en el panel Salida:

  ```
  var myObject:Object = {name:'Joe', age:25, city:'San Francisco'};
  var propertyName:String;
  for (propertyName in myObject) {
    trace("myObject has the property: " + propertyName + ", with the
    value: " + myObject[propertyName]);
  }
  ```

  Este ejemplo genera el siguiente resultado en el panel Salida:

  ```
  myObject has the property: name, with the value: Joe
  myObject has the property: age, with the value: 25
  myObject has the property: city, with the value: San Francisco
  ```

Si lo desea, puede establecer que su script se repita sobre un tipo de subnivel en particular, por ejemplo, solamente sobre clips de película secundarios. Puede hacerlo utilizando `for..in` con el operador `typeof`. En el siguiente ejemplo, una instancia de clip de película secundario (denominada `instance2`) se encuentra dentro de una instancia de clip de película en el escenario. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
for (var myName in this) {
  if (typeof (this[myName]) == "movieclip") {
    trace("I have a movie clip child named " + myName);
  }
}
```

Para más información sobre la sentencia `for..in`, consulte "Utilización de bucles for..in" en la página 124.

**ADVERTENCIA**

Las repeticiones en Flash se ejecutan con gran rapidez en Flash Player; los bucles, sin embargo, dependen en gran medida del procesador. Cuantas más repeticiones tenga un bucle y más sentencias se ejecuten en cada bloque, más recursos se consumirán del procesador. Los bucles mal escritos pueden generar problemas de rendimiento y de estabilidad.

Para más información sobre cada sentencia, consulte las secciones correspondientes incluidas posteriormente en este capítulo como, por ejemplo, "Utilización de bucles while" en la página 126, así como sus respectivas entradas en *Referencia del lenguaje ActionScript 2.0*.

## Creación y finalización de bucles

En el siguiente ejemplo se muestra una matriz sencilla de nombres de meses. Un bucle `for` repite de `0` hasta el número de elementos de la matriz y muestra cada elemento en el panel Salida.

```
var monthArr:Array = new Array("Jan", "Feb", "Mar", "Apr", "May", "Jun",
  "Jul", "Aug", "Sep", "Oct", "Nov", "Dec");
var i:Number;
for (i = 0; i < monthArr.length; i++) {
  trace(monthArr[i]);
}
```

Al utilizar matrices, ya sean éstas simples o complejas, deberá tener en cuenta una condición denominada *bucle infinito*. Un bucle infinito, como su propio nombre indica, es un bucle que carece de condición de finalización. Esto provoca problemas reales: que se bloquee la aplicación Flash, que el documento de Flash deje de responder en un navegador Web o un comportamiento extremadamente incoherente del documento de Flash. El siguiente código es un ejemplo de bucle infinito:

```
// CÓDIGO ERRÓNEO: crea un bucle infinito
// NO SE ASUMEN RESPONSABILIDADES POR SU USO.
var i:Number;
for (i = 0; i < 10; i--) {
  trace(i);
}
```

El valor de `i` se inicializa en 0 y la condición de finalización se cumple cuando `i` es mayor o igual que 10; tras cada repetición, se reduce el valor de `i`. Probablemente observe el error de inmediato: si el valor de `i` se reduce después de cada repetición del bucle, la condición de finalización nunca se cumple. Los resultados varían dependiendo del equipo en que se ejecute, mientras que la velocidad con que se produce el error del código está en función de la velocidad de la CPU y de otros factores. Por ejemplo, el bucle se ejecuta unas 142.620 veces antes de mostrar un mensaje de error en un equipo determinado.

El siguiente mensaje de error se muestra en un cuadro de diálogo:

```
A script in this movie is causing Flash Player to run slowly. If it
  continues to run, your computer may become unresponsive. Do you want to
  abort the script?
```

Al utilizar bucles (y, especialmente, bucles `while` y `do..while`), asegúrese siempre de que el bucle puede terminarse correctamente y que no se convierte en un bucle infinito.

Para más información sobre el control de bucles, consulte “Utilización de una sentencia switch” en la página 112.

## Utilización de bucles for

El bucle `for` le permite repetir una variable para un intervalo de valores específico. Los bucles `for` resultan útiles cuando se conoce exactamente el número de veces que es necesario repetir una serie de sentencias ActionScript. Esto puede ser de utilidad si se desea duplicar un clip de película en el escenario un número determinado de veces o reproducir en bucle una matriz y realizar una tarea en cada elemento de dicha matriz. Un bucle `for` repite una acción mediante un contador incorporado. En una sentencia `for`, el contador y la sentencia que incrementa el contador son parte de la sentencia `for`. La sentencia `for` se escribe utilizando el siguiente formato básico:

```
for (init; condition; update) {
  // sentencias;
}
```

Debe proporcionar tres expresiones a una sentencia `for`: una variable que se establece con un valor inicial, una sentencia condicional que determina cuándo termina la reproducción en bucle y una expresión que cambia el valor de la variable con cada bucle. Por ejemplo, el siguiente código realiza cinco bucles. El valor de la variable `i` comienza en `0` y termina en `4`, mientras que la salida son los números `0` a `4`, cada uno de ellos en su propia línea.

```
var i:Number;
for (i = 0; i < 5; i++) {
  trace(i);
}
```

En el ejemplo siguiente, la primera expresión (`i = 0`) es la expresión inicial cuyo resultado se calcula antes de la primera repetición. La segunda expresión ( `i < 5`) es la condición que se comprueba antes de que se ejecute cada bucle. La tercera expresión (`i++`) se denomina *expresión posterior* y su resultado siempre se calcula después de que se ejecute el bucle.

### Para crear un bucle for:

1. Seleccione Archivo > Nuevo y seleccione Documento de Flash.
2. Cree un clip de película en el escenario.
3. Haga clic con el botón derecho del ratón en el símbolo de clip de película en el panel Biblioteca y seleccione Vinculación del menú contextual.
4. Seleccione la casilla de verificación Exportar para ActionScript y escriba **libraryLinkageClassName** en el campo de introducción de texto Clase. Haga clic en Aceptar.

5. Seleccione el fotograma 1 de la línea de tiempo y escriba el siguiente código ActionScript en el panel Acciones:

```
var i:Number;
for (i = 0; i < 5; i++) {
  this.attachMovie("libraryLinkageClassName", "clip" + i + "_mc", i,
  {_x:(i * 100)});
}
```

6. Seleccione Control > Probar película para probar el código en Flash Player.

   Observe cómo se duplican cinco clips de película a lo largo de la parte superior del escenario. Este código ActionScript duplica el símbolo de clip de película en la biblioteca y cambia la posición de los clips en el escenario para situarlos en las coordenadas *x* correspondientes a 0, 100, 200, 300 y 400 píxeles. El bucle se ejecuta cinco veces, asignándose a la variable `i` un valor entre 0 y 4. En la última repetición del bucle, el valor de `i` se incrementa hasta 4 y la segunda expresión (`i < 5`) deja de ser verdadera (true), lo que provoca que el bucle finalice.

Recuerde incluir un espacio después de cada expresión de una sentencia `for`. Para más información, consulte `%{for statement}%` en la *Referencia del lenguaje ActionScript 2.0*.

## Utilización de bucles for..in

Utilice la sentencia `for..in` para ejecutar en bucle (o *repetir*) los subniveles de un clip de película, las propiedades de un objeto o los elementos de una matriz. Los subniveles, a los que se ha hecho referencia previamente, incluyen otros clips de película, funciones, objetos y variables. Entre los usos habituales del bucle `for..in` figura la reproducción en bucle de instancias de una línea de tiempo o de pares clave/valor contenidos en un objeto. La reproducción en bucle de objetos puede resultar efectiva para depurar aplicaciones porque le permite ver qué datos se devuelven del servicio Web o de documentos externos tales como archivos de texto o XML.

Por ejemplo, puede utilizar un bucle `for...in` para repetir las propiedades de un objeto genérico (las propiedades de un objeto no se guardan en ningún orden concreto, por lo que aparecen de forma impredecible):

```
var myObj:Object = {x:20, y:30};
for (var i:String in myObj) {
  trace(i + ": " + myObj[i]);
}
```

Este código muestra la siguiente información en el panel Salida:

```
x: 20
y: 30
```

También puede repetir los elementos de una matriz:

```
var myArray:Array = ["one", "two", "three"];
for (var i:String in myArray) {
  trace(myArray[i]);
}
```

Este código muestra la siguiente información en el panel Salida:

```
three
two
one
```

Para más información sobre objetos y propiedades, consulte "Tipo de datos Object (objeto)" en la página 335.

| NOTA | No se pueden repetir las propiedades de un objeto si se trata de una instancia de una clase personalizada, a no ser que la clase sea una clase dinámica. Incluso con instancias de clases dinámicas, sólo puede repetir las propiedades que se añadan dinámicamente. |
|---|---|

| NOTA | Las llaves (`{}`) que se utilizan para incluir el bloque de sentencias que se ejecutan con la sentencia `for..in` no son necesarias si sólo se ejecuta una sentencia. |
|---|---|

En el siguiente ejemplo se utiliza `for..in` para repetir las propiedades de un objeto:

### Para crear un bucle for:

1. Seleccione Archivo > Nuevo y seleccione Documento de Flash.
2. Seleccione el fotograma 1 de la línea de tiempo y escriba el siguiente código ActionScript en el panel Acciones:

   ```
   var myObj:Object = {name:"Tara", age:27, city:"San Francisco"};
   var i:String;
   for (i in myObj) {
     trace("myObj." + i + " = " + myObj[i]);
   }
   ```

3. Seleccione Control > Probar película para probar el código en Flash Player.

   Al probar el archivo SWF, debería ver el siguiente texto en el panel Salida:

   ```
   myObj.name = Tara
   myObj.age = 27
   myObj.city = San Francisco
   ```

Si escribe un bucle `for..in` en un archivo de clase (un archivo de ActionScript externo), los miembros de instancias no estarán disponibles para el bucle, pero sí los miembros estáticos. Sin embargo, si escribe un bucle `for..in` en un archivo FLA para una instancia de la clase, los miembros de instancias estarán disponibles, pero no los miembros estáticos. Para más información sobre la escritura de archivos de clases, consulte el Capítulo 6, "Clases", en la página 195. Para más información, consulte `%{for..in statement}%` en la *Referencia del lenguaje ActionScript 2.0*.

## Utilización de bucles while

Utilice la sentencia `while` para repetir una acción mientras se dé una condición; es similar a una sentencia `if` que se repite mientras la condición sea `true`.

Un bucle `while` calcula el resultado de una expresión y ejecuta el código en el cuerpo del bucle si la expresión es `true`. Si la condición da como resultado `true`, se ejecuta una sentencia o una serie de sentencias antes de regresar para comprobar la condición de nuevo. Cuando la condición dé como resultado `false`, se omitirá la sentencia o serie de sentencias y finaliza el bucle. La utilización de bucles `while` puede resultar muy útil si no está seguro del número de veces que debe repetirse el bucle de un bloque de código.

Por ejemplo, el siguiente código traza números en el panel Salida:

```
var i:Number = 0;
while (i < 5) {
  trace(i);
  i++;
}
```

Observará cómo se trazan los siguientes números en el panel Salida:

```
0
1
2
3
4
```

Una desventaja que presenta el uso de los bucles `while` frente a los bucles `for` es que resulta más probable escribir un bucle infinito con bucles `while`. El código de ejemplo de bucle `for` no se compila si se omite la expresión que aumenta la variable counter, mientras que el ejemplo de bucle `while` sí se compila si se omite dicho paso. Sin la expresión que incrementa `i`, el bucle se convierte en un bucle infinito.

Para crear y utilizar un bucle `while` en un archivo FLA, siga este ejemplo.

### Para crear un bucle while:

1. Seleccione Archivo > Nuevo y seleccione Documento de Flash.
2. Abra el panel Componentes y arrastre un componente DataSet al escenario.
3. Abra el inspector de propiedades (Ventana > Propiedades > Propiedades) y escriba el nombre de instancia **users_ds**.

4. Seleccione el fotograma 1 de la línea de tiempo y escriba el siguiente código ActionScript en el panel Acciones:

```
var users_ds:mx.data.components.DataSet;
//
users_ds.addItem({name:"Irving", age:34});
users_ds.addItem({name:"Christopher", age:48});
users_ds.addItem({name:"Walter", age:23});
//
users_ds.first();
while (users_ds.hasNext()) {
  trace("name:" + users_ds.currentItem["name"] + ", age:" +
  users_ds.currentItem["age"]);
  users_ds.next();
}
```

5. Seleccione Control > Probar película para probar el documento.

   El panel Salida muestra la siguiente información:

```
name:Irving, age:34
name:Christopher, age:48
name:Walter, age:23
```

Para más información, consulte `%{while statement}%` en la *Referencia del lenguaje ActionScript 2.0*.

## Bucles do..while

Puede utilizar la sentencia `do...while` para crear el mismo tipo de reproducción que `while`. Sin embargo, la expresión se calcula al final del bloque de código en un bucle `do..while` (se comprueba después de que se ejecute el bloque de código), por lo que el bucle siempre se ejecuta al menos una vez. Las sentencias sólo se ejecutan si la condición da como resultado `true`.

En el siguiente código se muestra un ejemplo sencillo de bucle `do..while` que genera un resultado aunque la condición no se cumpla.

```
var i:Number = 5;
do {
  trace(i);
  i++;
} while (i < 5);
// Salida: 5
```

Cuando utilice bucles, deberá evitar escribir bucles infinitos. Si la condición de un bucle `do..while` da permanentemente como resultado `true`, habrá creado un bucle infinito que muestra una advertencia o provoca que Flash Player se bloquee. Utilice como alternativa un bucle `for` si conoce el número de veces que debe ejecutarse el bucle. Para más información y ejemplos de `%{do..while statement}%`, consulte *Referencia del lenguaje ActionScript 2.0*.

## Utilización de bucles anidados en el código ActionScript

En el siguiente ejemplo se muestra cómo crear una matriz de objetos y mostrar los valores de cada uno de ellos en la estructura anidada. En este ejemplo se muestra cómo utilizar el bucle `for` para ejecutarlo en cada uno de los elementos de la matriz y cómo utilizar el bucle `for..in` para repetir cada par clave/valor de los objetos anidados.

### Anidación de un bucle dentro de otro:

1. Cree un nuevo documento de Flash.
2. Seleccione Archivo > Guardar como y asigne al documento el nombre **loops.fla**.
3. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
var myArr:Array = new Array();
myArr[0] = {name:"One", value:1};
myArr[1] = {name:"Two", value:2};
//
var i:Number;
var item:String;
for (i = 0; i < myArr.length; i++) {
  trace(i);
  for (item in myArr[i]) {
    trace(item + ": " + myArr[i][item]);
  }
  trace("");
}
```

4. Seleccione Control > Probar película para probar el código.

   El panel Salida muestra esta información.

```
0
name: One
value: 1

1
name: Two
value: 2
```

   Usted conoce el número de elementos existentes en la matriz, por lo que puede ejecutar el bucle por cada uno de los elementos empleando simplemente un bucle `for`. Dado que cada objeto de la matriz puede tener diferentes pares nombre/valor, puede utilizar un bucle `for..in` para repetir cada valor y mostrar los resultados en el panel Salida.

# Matrices

Una *matriz* es un objeto cuyas propiedades se identifican mediante números que representan sus posiciones en la estructura. Básicamente, una matriz es una lista de elementos. Es importante recordar que no todos los elementos de una matriz tienen que pertenecer al mismo tipo de datos. Puede combinar números, fechas, cadenas, objetos e incluso añadir una matriz anidada en cada índice de matriz.

En el siguiente ejemplo se incluye una matriz sencilla de nombres de meses.

```
var myArr:Array = new Array();
myArr[0] = "January";
myArr[1] = "February";
myArr[2] = "March";
myArr[3] = "April";
```

La matriz anterior de nombres de meses también puede escribirse de nuevo de la siguiente forma:

```
var myArr:Array = new Array("January", "February", "March", "April");
```

O bien puede utilizar la sintaxis abreviada de la siguiente forma:

```
var myArr:Array = ["January", "February", "March", "April"];
```

Una matriz es una especie de estructura para albergar datos. Una matriz es como un edificio de oficinas, en el que cada planta contiene un tipo de datos diferente (por ejemplo, *accounting* -contabilidad- en la 3ª planta y *engineering* -diseño- en la 5ª planta). Esto permite almacenar diferentes tipos de datos en una sola matriz, incluidas otras matrices. Cada planta de este edificio puede contener múltiples tipos de contenido (*executives* -ejecutivos- y *accounting* -contabilidad- podrían compartir la 3ª planta).

Una matriz contiene *elementos* que equivalen a cada una de las plantas del edificio. Cada elemento tiene una posición numérica (el *índice*), lo que le permite referirse a la posición de cada elemento en la matriz. Esto es igual que asignar un número a cada planta de un edificio. Cada elemento puede contener un dato (que podría ser un número, una cadena, un valor booleano o incluso una matriz o un objeto) o estar vacío.

También puede controlar y modificar la propia matriz. Por ejemplo, puede trasladar el departamento de diseño a la planta baja del edificio. Las matrices le permiten desplazar valores y cambiar el tamaño de la matriz (por continuar con el símil, digamos que se reforma el edificio y se añaden o eliminan plantas). Esto permite añadir o eliminar elementos y mover los valores a elementos diferentes.

Por consiguiente, el edificio (la matriz) contiene plantas (los elementos) que están numeradas (el índice), al tiempo que cada planta contiene uno o más departamentos (los valores).

Para más información sobre la modificación de matrices, consulte "Modificación de matrices" en la página 132. Para obtener información sobre la utilización de matrices e índices, consulte "Utilización de matrices" en la página 130. Para obtener información sobre la adición y eliminación de elementos, consulte "Adición y eliminación de elementos" en la página 134. Para obtener información sobre el operador de acceso a una matriz, consulte "Utilización de operadores de punto y de acceso a una matriz" en la página 152.

El archivo de origen de muestra, array.fla, se puede encontrar en la carpeta Samples del disco duro. Este ejemplo ilustra la manipulación de matrices mediante ActionScript. El código del ejemplo crea una matriz y ordena, añade y elimina elementos de dos componentes List. Busque el archivo de muestra en los siguientes directorios:

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Arrays.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Arrays.

## Utilización de matrices

Puede utilizar las matrices de varias formas en su trabajo. Puede emplearlas para almacenar listas de objetos, como, por elemento, una serie de artículos devueltos. Si carga datos de servidores Web remotos, puede que incluso reciba datos en forma de matriz de objetos anidados. Las matrices suelen contener datos con un formato similar. Por ejemplo, si crea una aplicación de audio en Flash, puede que tenga una lista de reproducción de un usuario almacenada en forma de matriz con información de canciones almacenadas en objetos. Cada objeto contiene el nombre de la canción, el nombre del intérprete, la duración de la canción, la ubicación de un archivo sonoro (por ejemplo, un archivo MP3) o cualquier otra información que sea preciso asociar a un archivo concreto.

La ubicación de un elemento de una matriz se conoce como *índice*. Todas las matrices están basadas en cero, lo que significa que el primer elemento de la matriz es [0], el segundo elemento, [1], y así sucesivamente.

Existen diversos tipos de matrices que irá descubriendo en las siguientes secciones. Las matrices suelen utilizar un índice numérico para consultar un elemento concreto de una *matriz indexada*. El segundo tipo de matriz se conoce como *matriz asociativa* y emplea un índice de texto en lugar de un índice numérico para consultar información. Para más información sobre matrices comunes, consulte "Matrices" en la página 129. Para más información sobre matrices asociativas, consulte "Creación de matrices asociativas" en la página 138. Para más información sobre matrices multidimensionales, consulte "Creación de matrices multidimensionales" en la página 135. Para obtener información sobre el operador de acceso a una matriz, consulte "Utilización de operadores de punto y de acceso a una matriz" en la página 152.

La clase incorporada Array le permite obtener acceso a matrices y manipularlas. Para crear un objeto Array, deberá utilizar el constructor `new Array()` o el operador de acceso a una matriz (`[]`). Para obtener acceso a elementos de una matriz, utilice el operador de acceso a matriz (`[]`). En el siguiente ejemplo se utiliza una matriz indexada.

### Para utilizar matrices en el código:

1. Cree un nuevo documento de Flash y guárdelo como **basicArrays.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
// define una nueva matriz
var myArr:Array = new Array();
// define valores en dos índices
myArr[1] = "value1";
myArr[0] = "value0";
// repite los elementos de la matriz
var i:String;
for (i in myArr) {
  // traza los pares clave/valor
  trace("key: " + i + ", value: " + myArr[i]);
}
```

   En la primera línea del código ActionScript, deberá definir una nueva matriz para que contenga los valores. Seguidamente, deberá definir datos (`value0` y `value1`) en dos índices de la matriz. Un bucle `for..in` permite repetir cada uno de los elementos de la matriz y mostrar los pares clave/valor en el panel Salida empleando una sentencia trace.
3. Seleccione Control > Probar película para probar el código.

   El panel Salida muestra este texto:

```
key: 0, value: value0
key: 1, value: value1
```

   Para más información sobre bucles `for..in`, consulte "Utilización de bucles for..in" en la página 124.

Para obtener información sobre cómo crear diferentes tipos de matrices, consulte las siguientes secciones:

- "Creación de matrices indexadas" en la página 135
- "Creación de matrices multidimensionales" en la página 135
- "Creación de matrices asociativas" en la página 138

El archivo de origen de muestra, array.fla, se puede encontrar en la carpeta Samples del disco duro. Este ejemplo ilustra la manipulación de matrices mediante ActionScript. El código del ejemplo crea una matriz y ordena, añade y elimina elementos de dos componentes List. Busque el archivo de muestra en los siguientes directorios:

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Arrays.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Arrays.

## Modificación de matrices

También puede controlar y modificar la matriz mediante código ActionScript. Puede mover valores a otro lugar de la matriz o cambiar el tamaño de la matriz. Por ejemplo, si desea intercambiar los datos contenidos en dos índices de una matriz, puede utilizar el siguiente código:

```
var buildingArr:Array = new Array();
buildingArr[2] = "Accounting";
buildingArr[4] = "Engineering";
trace(buildingArr); // undefined,undefined,Accounting,undefined,Engineering

var temp_item:String = buildingArr[2];
buildingArr[2] = buildingArr[4];
buildingArr[4] = temp_item;
trace(buildingArr); // undefined,undefined,Engineering,undefined,Accounting
```

Puede que se pregunte por qué es necesario crear una variable temporal en el ejemplo anterior. Si ha copiado el contenido del índice de matriz 4 en el índice de matriz 2 y viceversa, se perderá el contenido original del índice de matriz 2. Al copiar el valor de uno de los índices de matriz a una variable temporal, podrá guardar el valor y copiarlo de nuevo más tarde en el código sin riesgo alguno. Por ejemplo, si utiliza el siguiente código en lugar del anterior, observará que se ha perdido el valor del índice de matriz 2 (Accounting). Ahora tiene dos equipos de diseño (engineering) pero ninguno de contabilidad (accounting).

```
// erróneo (no hay variable temporal)
buildingArr[2] = buildingArr[4];
buildingArr[4] = buildingArr[2];
trace(buildingArr); //
  undefined,undefined,Engineering,undefined,Engineering
```

El archivo de origen de muestra, array.fla, se puede encontrar en la carpeta Samples del disco duro. Este ejemplo ilustra la manipulación de matrices mediante ActionScript. El código del ejemplo crea una matriz y ordena, añade y elimina elementos de dos componentes List. Busque el archivo de muestra en los siguientes directorios:

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Arrays.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Arrays.

## Referencia y localización de longitud

Al utilizar matrices, normalmente es necesario conocer el número de elementos existentes en la matriz. Esto puede resultar especialmente útil al escribir bucles `for` que se repiten en cada elemento de la matriz y ejecutan una serie de sentencias. Observe el ejemplo del siguiente fragmento:

```
var monthArr:Array = new Array("Jan", "Feb", "Mar", "Apr", "May", "Jun",
  "Jul", "Aug", "Sep", "Oct", "Nov", "Dec");
trace(monthArr); // Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec
trace(monthArr.length); // 12
var i:Number;
for (i = 0; i < monthArr.length; i++) {
  monthArr[i] = monthArr[i].toUpperCase();
}
trace(monthArr); // JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC
```

En el ejemplo anterior, se crea una matriz y se rellena con los nombres de los meses. Se muestra el contenido, así como la longitud de la matriz. Un bucle `for` se repite sobre cada elemento de la matriz y convierte el valor a mayúsculas, tras lo cual vuelve a mostrarse el contenido de la matriz.

En el siguiente código ActionScript, si crea un elemento en el índice de matriz 5, la longitud de la matriz devuelve `6` (porque la matriz está basada en cero) y no el número real de elementos de la matriz como podría esperarse:

```
var myArr:Array = new Array();
myArr[5] = "five";
trace(myArr.length); // 6
trace(myArr); // undefined,undefined,undefined,undefined,undefined,five
```

Para más información sobre bucles `for`, consulte "Utilización de bucles for" en la página 123. Para obtener información sobre el operador de acceso a una matriz, consulte "Utilización de operadores de punto y de acceso a una matriz" en la página 152.

El archivo de origen de muestra, array.fla, se puede encontrar en la carpeta Samples del disco duro. Este ejemplo ilustra la manipulación de matrices mediante ActionScript. El código del ejemplo crea una matriz y ordena, añade y elimina elementos de dos componentes List. Busque el archivo de muestra en los siguientes directorios:

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Arrays.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Arrays.

## Adición y eliminación de elementos

Una matriz contiene elementos, cada uno de ellos con su posición numérica (el índice), lo que le permite referirse a la posición de cada elemento en la matriz. Cada elemento puede contener un dato o estar vacío. Un elemento puede contener los siguientes datos: un número, una cadena, un valor booleano o incluso una matriz o un objeto.

Al crear elementos en una matriz, debe crear los índices secuencialmente siempre que sea posible. Esto facilita la depuración de aplicaciones. En "Referencia y localización de longitud" en la página 133, hemos visto que, si asigna un valor único en el índice 5 de una matriz, el valor que se devuelve como longitud de la matriz es 6. Esto provoca que se inserten cinco valores no definidos (undefined) en la matriz.

En el siguiente ejemplo se muestra cómo crear una matriz nueva, eliminar un elemento de un índice concreto y añadir y reemplazar datos de un índice de la matriz:

```
var monthArr:Array = new Array("Jan", "Feb", "Mar", "Apr", "May", "Jun",
  "Jul", "Aug", "Sep", "Oct", "Nov", "Dec");
delete monthArr[5];
trace(monthArr); // Jan,Feb,Mar,Apr,May,undefined,Jul,Aug,Sep,Oct,Nov,Dec
trace(monthArr.length); // 12
monthArr[5] = "JUN";
trace(monthArr); // Jan,Feb,Mar,Apr,May,JUN,Jul,Aug,Sep,Oct,Nov,Dec
```

Aunque haya eliminado el elemento del índice de matriz 5, la longitud de la matriz continúa siendo 12 y el elemento que había en el índice de matriz 5 se convierte en una cadena en blanco en lugar de desaparecer por completo.

El archivo de origen de muestra, array.fla, se puede encontrar en la carpeta Samples del disco duro. Este ejemplo ilustra la manipulación de matrices mediante ActionScript. El código del ejemplo crea una matriz y ordena, añade y elimina elementos de dos componentes List. Busque el archivo de muestra en los siguientes directorios:

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Arrays.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Arrays.

## Creación de matrices indexadas

Las matrices indexadas almacenan uno o varios valores. Puede consultar elementos por su posición en la matriz, lo que puede que haya hecho ya en secciones anteriores. El primer número de índice siempre es `0` y aumenta en uno por cada elemento posterior que añada a la matriz. Una matriz indexada se crea llamando al constructor de la clase Array o inicializando la matriz con un literal de matriz. Las matrices se crean mediante el constructor Array y un literal de matriz en el siguiente ejemplo.

### Para crear una matriz indexada:

1. Cree un nuevo documento de Flash y guárdelo como **indexArray.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

   ```
   var myArray:Array = new Array();
   myArray.push("one");
   myArray.push("two");
   myArray.push("three");
   trace(myArray); // one,two,three
   ```

   En la primera línea del código ActionScript, deberá definir una nueva matriz para que contenga los valores.
3. Seleccione Control > Probar película para probar el código.

   El panel Salida muestra este texto:

   ```
   one,two,three
   ```
4. Regrese a la herramienta de edición y elimine el código del panel Acciones.
5. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

   ```
   var myArray:Array = ["one", "two", "three"];
   trace(myArray); // one,two,three
   ```

   En este código, se emplea el literal de matriz para definir una nueva matriz en el código. Este código es equivalente al código ActionScript escrito en el paso 2. Al probar el código, se ve la misma pantalla de salida en el panel Salida.

## Creación de matrices multidimensionales

En ActionScript, puede implementar matrices como *matrices anidadas* que, en definitiva, son matrices de matrices. Las matrices anidadas, también conocidas como *matrices multidimensionales*, pueden concebirse como cuadrículas. Por consiguiente, puede utilizar matrices multidimensionales al programar para modelar este tipo de estructuras. Por ejemplo, un tablero de ajedrez es una cuadrícula de ocho columnas y filas; puede modelarse como una matriz con ocho elementos, cada uno de los cuales es, a su vez, una matriz que contiene ocho elementos.

Piense, por ejemplo, en una lista de tareas almacenadas en forma de matriz de cadenas indexadas:

```
var tasks:Array = ["wash dishes", "take out trash"];
```

Si desea almacenar una lista independiente de tareas por cada día de la semana, puede crear una matriz multidimensional con un elemento por cada día de la semana. Cada elemento contiene una matriz indexada que almacena la lista de tareas.

| ATENCIÓN | Al utilizar el operador de acceso a una matriz, el compilador de ActionScript no puede comprobar si el elemento al que se accede es una propiedad válida del objeto. |
|---|---|

### Para crear una matriz multidimensional básica y recuperar elementos de la matriz:

1. Cree un nuevo documento de Flash y guárdelo como **multiArray1.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

   ```
   var twoDArray:Array = new Array(new Array("one","two"), new
     Array("three", "four"));
   trace(twoDArray);
   ```

   Esta matriz, `twoDArray`, consta de dos elementos de matriz. Estos elementos son en sí mismos matrices formadas por dos elementos. En este caso, `twoDArray` es la matriz principal que contiene dos matrices anidadas.
3. Seleccione Control > Probar película para probar el código. Se ve lo siguiente en el panel Salida.

   ```
   one,two,three,four
   ```
4. Regrese a la herramienta de edición y abra el panel Acciones. Convierta la sentencia `trace` en un comentario como se muestra a continuación:

   ```
   // trace(twoDArray);
   ```
5. Añada el siguiente código ActionScript al final del código en el fotograma 1 de la línea de tiempo:

   ```
   trace(twoDArray[0][0]); // one
   trace(twoDArray[1][1]); // four
   ```

   Para recuperar elementos de una matriz multidimensional, deberá utilizar múltiples operadores de acceso a una matriz (`[]`) tras el nombre de la matriz de nivel superior. Los primeros `[]` se refieren al índice de la matriz de nivel superior. Los posteriores operadores de acceso a una matriz se refieren a los elementos de las matrices anidadas.

6. Seleccione Control > Probar película para probar el código. Se ve lo siguiente en el panel Salida.

```
one
four
```

Puede utilizar bucles `for` anidados para crear matrices multidimensionales. En el siguiente ejemplo se muestra cómo hacerlo.

### Para crear una matriz multidimensional utilizando un bucle for:

1. Cree un nuevo documento de Flash y guárdelo como **multiArray2.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
var gridSize:Number = 3;
var mainArr:Array = new Array(gridSize);
var i:Number;
var j:Number;
for (i = 0; i < gridSize; i++) {
  mainArr[i] = new Array(gridSize);
  for (j = 0; j < gridSize; j++) {
    mainArr[i][j] = "[" + i + "][" + j + "]";
  }
}
trace(mainArr);
```

   Este código ActionScript crea una matriz de 3 x 3 y establece el valor de cada nodo de la matriz con su índice. Luego deberá trazar la matriz (`mainArr`).

3. Seleccione Control > Probar película para probar el código.

   Se ve lo siguiente en el panel Salida:

```
[0][0],[0][1],[0][2],[1][0],[1][1],[1][2],[2][0],[2][1],[2][2]
```

También puede utilizar bucles `for` anidados para repetir los elementos de una matriz multidimensional, como se muestra en el siguiente ejemplo.

### Para utilizar un bucle for con el fin de repetir una matriz multidimensional:

1. Cree un nuevo documento de Flash y guárdelo como **multiArray3.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
// del ejemplo anterior
var gridSize:Number = 3;
var mainArr:Array = new Array(gridSize);
var i:Number;
var j:Number;
for (i = 0; i < gridSize; i++) {
  mainArr[i] = new Array(gridSize);
  for (j = 0; j < gridSize; j++) {
    mainArr[i][j] = "[" + i + "][" + j + "]";
  }
}
```

En este código, que ya hemos visto en el ejemplo anterior, el bucle externo repite cada uno de los elementos de `mainArray`. El bucle interno repite cada una de las matrices anidadas y da como resultado el nodo de cada matriz.

3. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo, detrás del código introducido en el paso 2:

```
// repetir elementos
var outerArrayLength:Number = mainArr.length;
for (i = 0; i < outerArrayLength; i++) {
  var innerArrayLength:Number = mainArr[i].length;
  for (j = 0; j < innerArrayLength; j++) {
    trace(mainArr[i][j]);
  }
}
```

Este código ActionScript repite los elementos de la matriz. La propiedad `length` de cada matriz se utiliza como condición para el bucle.

4. Seleccione Control > Probar película para ver los elementos que se muestran en el panel Salida. Deberá ver lo siguiente en el panel Salida:

```
[0][0]
[0][1]
[0][2]
[1][0]
[1][1]
[1][2]
[2][0]
[2][1]
[2][2]
```

Para obtener información sobre la utilización de matrices, consulte "Utilización de matrices" en la página 130. Para obtener información sobre elementos de matrices, consulte "Adición y eliminación de elementos" en la página 134. Para obtener información sobre el operador de acceso a una matriz, consulte "Utilización de operadores de punto y de acceso a una matriz" en la página 152.

## Creación de matrices asociativas

Una matriz asociativa, que es como un objeto, está formada por *claves* y *valores* sin ordenar. Las matrices asociativas utilizan claves en lugar de un índice numérico para organizar los valores almacenados. Cada clave es una cadena exclusiva y está asociada y se utiliza para acceder a un valor. El tipo de datos de dicho valor puede ser Number, Array, Object, etc. Al crear código para localizar un valor asociado a una clave, estará indexando o realizando una *consulta*. Probablemente éste será el uso más habitual que dará a las matrices asociativas.

La asociación entre una clave y un valor se conoce normalmente como su *vinculación*; la clave y el valor están *asignados* el uno al otro. Por ejemplo, una agenda de contactos podría considerarse una matriz asociativa en la que los nombres son las claves y las direcciones de correo electrónico son los valores.

| NOTA | Las matrices asociativas son conjuntos no ordenados de pares formados por una clave y un valor. El código no debe dar por sentado que las claves de una matriz asociativa estén en un orden específico. |
|---|---|

Al utilizar matrices asociativas, puede llamar al elemento de matriz que necesite empleando una cadena en lugar de un número, ya que las cadenas suelen ser más fáciles de recordar. La desventaja es que no son tan útiles en un bucle porque no utilizan números como valor de índice. *Sí* son útiles si normalmente necesita consultar por valor de clave. Por ejemplo, si tiene una matriz de nombres y edades que necesita consultar con frecuencia, podría utilizar una matriz asociativa.

En el siguiente ejemplo se muestra cómo crear un objeto y definir una serie de propiedades en una matriz asociativa.

### Para crear una matriz asociativa sencilla:

1. Cree un nuevo documento de Flash.
2. Escriba el siguiente código ActionScript en el fotograma 1 de la línea de tiempo principal:

```
// Definir el objeto que se va a usar como matriz asociativa.
var someObj:Object = new Object();
// Definir una serie de propiedades.
someObj.myShape = "Rectangle";
someObj.myW = 480;
someObj.myH = 360;
someObj.myX = 100;
someObj.myY = 200;
someObj.myAlpha = 72;
someObj.myColor = 0xDFDFDF;
// Mostrar una propiedad utilizando el operador de punto y sintaxis de
  acceso a una matriz.
trace(someObj.myAlpha); // 72
trace(someObj["myAlpha"]); // 72
```

   La primera línea del código ActionScript define un nuevo objeto (`someObj`) que se utiliza como matriz asociativa. Seguidamente, se definen una serie de propiedades de `someObj`. Finalmente, se muestra una propiedad que se selecciona empleando el operador de punto y la sintaxis de acceso a una matriz.

| NOTA | Puede acceder a las variables de una matriz asociativa empleando dos métodos diferentes: sintaxis con puntos (`someObj.myColor`) y sintaxis de acceso a una matriz (`someObj['myColor']`). |
|---|---|

3. Seleccione Control > Probar película para probar el código ActionScript.

   El panel Salida muestra el número `72` dos veces, que representan a los dos niveles alfa trazados.

Existen dos formas de crear matrices asociativas en ActionScript 2.0:

- Mediante un constructor Object
- Mediante un constructor Array

Ambos se demuestran en los próximos ejemplos.

| NOTA | En el ejemplo anterior, se ha utilizado un constructor Object para crear una matriz asociativa. |
|---|---|

Si utiliza el constructor Object para crear una matriz asociativa, puede aprovechar las ventajas que ofrece la inicialización de la matriz con un literal de objeto. Una instancia de la clase Object, conocida también como objeto genérico, es funcionalmente idéntica a una matriz asociativa. De hecho, las instancias de Object son básicamente matrices asociativas. Puede utilizar matrices asociativas para lograr una funcionalidad similar a la de un diccionario, en los que es más adecuado contar con claves de cadena que con índices numéricos. El nombre de cada propiedad del objeto genérico actúa a modo de clave que permite el acceso a un valor almacenado. Para más información sobre el uso de literales, consulte "Literales" en la página 94. Para más información sobre clases, consulte el Capítulo 6, "Clases", en la página 195.

### Para crear una matriz asociativa empleando un constructor Object:

1. Cree un nuevo documento de Flash y guárdelo como **assocArray.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
var monitorInfo:Object = {type:"Flat Panel", resolution:"1600 x 1200"};
trace(monitorInfo["type"] + ", " + monitorInfo["resolution"]);
```

   Este código crea una matriz asociativa denominada `monitorInfo` y utiliza un literal de objeto para inicializar la matriz con dos pares clave/valor.

| NOTA | Si no necesita inicializar la matriz en el momento de su declaración, puede utilizar el constructor Object para crear la matriz: |
|---|---|

```
var monitorInfo:Object = new Object();
```

3. Seleccione Control > Probar película.

   El panel Salida muestra el siguiente texto:

```
Flat Panel, 1600 x 1200
```

4. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo, detrás del código introducido previamente:

```
monitorInfo["aspectRatio"] = "16:10";
monitorInfo.colors = "16.7 million";
trace(monitorInfo["aspectRatio"] + ", " + monitorInfo.colors);
```

Tras utilizar un literal de objeto o el constructor de la clase Object para crear la matriz, puede añadir nuevos valores a la matriz empleando el operador de corchetes (`[]`) o el operador de punto (`.`), como se muestra en este código. El código que acaba de escribir añade dos valores nuevos a la matriz `monitorInfo`.

5. Seleccione Control > Probar película.

El panel Salida muestra el siguiente texto:

```
16:10, 16.7 million
```

Tenga en cuenta que una clave puede contener un carácter de espacio. Esto es posible con el operador de corchetes, pero genera un error con el operador de punto. No es recomendable utilizar espacios en los nombres de claves. Para más información sobre operadores de corchetes y de punto, consulte "Operadores" en la página 142. Para más información sobre la aplicación de un formato correcto, consulte "Aplicación de formato a la sintaxis de ActionScript" en la página 808.

La segunda forma de crear una matriz asociativa consiste en utilizar el constructor Array y utilizar después el operador de corchetes (`[]`) o el operador de punto (`.`) para añadir pares de claves y valores a la matriz. Si declara la matriz asociativa con el tipo Array, no podrá utilizar un literal de objeto para inicializar la matriz.

**NOTA** La utilización del constructor Array para crear una matriz asociativa no aporta ninguna ventaja. El constructor Array es más adecuado para crear matrices indexadas.

En el siguiente ejemplo se muestra cómo utilizar el constructor Array para crear una matriz asociativa.

### Para crear una matriz asociativa empleando un constructor Array:

1. Cree un nuevo documento de Flash y guárdelo como **assocArray2.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
var monitorInfo:Array = new Array();
monitorInfo["type"] = "Flat Panel";
monitorInfo["resolution"] = "1600 x 1200";
trace(monitorInfo["type"] + ", " + monitorInfo["resolution"]);
```

Este código crea una matriz asociativa denominada `monitorInfo` empleando el constructor Array y añade una clave denominada `type` y otra denominada `resolution`, junto con sus correspondientes valores.

3. Seleccione Control > Probar película.

   El panel Salida muestra el siguiente texto:

```
Flat Panel, 1600 x 1200
```

| NOTA | La utilización del constructor Array para crear una matriz asociativa no aporta ninguna ventaja. El constructor Array es más adecuado para crear matrices indexadas. |
|---|---|

Las matrices asociativas son básicamente instancias de la clase Object, por lo que no hay ninguna ventaja en crear matrices asociativas empleando el constructor Array. Aunque cree una matriz asociativa con el constructor `new Array()`, no puede utilizar ninguno de los métodos y propiedades de la clase Array (como `sort()` o `length`) al utilizar la matriz asociativa. Si desea utilizar pares clave/valor en lugar de un índice numérico, debe emplear la clase Object en lugar de una matriz asociativa.

# Operadores

En esta sección se describen las reglas generales para los tipos comunes de operadores, precedencia del operador y asociatividad de operadores.

Los operadores son caracteres que especifican cómo combinar, comparar o cambiar los valores de una expresión. Una expresión es cualquier sentencia para la que Flash puede calcular el resultado y que devuelve un valor. Puede crear una expresión combinando operadores y valores, o bien llamando a una función. Para más información sobre expresiones, consulte "Sintaxis, sentencias y expresiones" en la página 76.

Por ejemplo, una expresión matemática utiliza operadores numéricos para manipular los valores que utiliza. Ejemplos de caracteres operadores son +, <, * y =. Una expresión está formada por operadores y *operandos* y puede ser cualquier combinación válida de símbolos de ActionScript que representen un valor. Un operando es la parte del código sobre la que actúa el operador. Por ejemplo, en la expresión `x + 2`, `x` y 2 son operandos y + es un operador.

Las expresiones y los operadores se utilizan con frecuencia en el código. Puede combinar operadores y valores para crear una expresión, o bien llamar a una función.

| NOTA | En esta sección se describe cómo utilizar cada tipo de operador; no obstante, no se abarcan todos y cada uno de ellos. Para obtener información sobre cada operador, incluidos los operadores especiales que no pertenecen a ninguna de las siguientes categorías, consulte *Referencia del lenguaje ActionScript 2.0*. |
|---|---|

Las partes del código sobre la que el operador actúa se denominan *operandos*. Por ejemplo, puede utilizar el operador de suma (+) para sumar valores a un literal numérico. Podría hacerlo para sumar el valor de una variable denominada `myNum`.

```
myNum + 3;
```

En este ejemplo, `myNum` y `3` son operandos.

En esta sección se describen las reglas generales para los tipos comunes de operadores, precedencia del operador y asociatividad de operadores:



Para obtener información sobre operadores que no pertenecen a ninguna de estas categorías, consulte *Referencia del lenguaje ActionScript 2.0*, donde se incluye información sobre todos los operadores que puede utilizar.

En las siguientes secciones se muestran algunos usos habituales de los operadores. Para más información sobre la utilización de muchos operadores en una sola muestra de código, consulte "Utilización de operadores en un documento" en la página 167.

# Utilización de operadores para manipular valores

Los operadores se utilizan habitualmente para manipular valores en Flash. Por ejemplo, puede crear un juego en Flash en el que la puntuación cambie dependiendo de la interacción del usuario con instancias del escenario. Puede utilizar una variable para que contenga el valor y operadores que manipulen el valor de la variable.

Por ejemplo, puede que desee aumentar el valor de una variable denominada `myScore`. En el siguiente ejemplo se muestra cómo utilizar los operadores + (suma) y += (asignación de suma) para sumar e incrementar valores en el código.

### Para manipular valores mediante operadores:

1. Cree un nuevo documento de Flash.
2. Abra el panel Acciones (Ventana > Acciones) e introduzca el código siguiente en el panel Script:

```
// ejemplo uno
var myScore:Number = 0;
myScore = myScore + 1;
trace("Example one: " + myScore); // 1

// ejemplo dos
var secondScore:Number = 1;
secondScore += 3;
trace("Example two: " + secondScore); // 4
```

3. Seleccione Control > Probar película.

   El panel Salida muestra el siguiente texto:

```
Example one: 1
Example two: 4
```

   El operador de suma es bastante sencillo, ya que suma dos valores. En el primer ejemplo de código, se suma el valor actual de `myScore` y el número 1 y, seguidamente, se almacena el resultado en la variable `myScore`.

El segundo ejemplo de código utiliza el operador de asignación de suma para sumar y asignar un nuevo valor en un solo paso. Puede reescribir la línea `myScore = myScore + 1` (del ejercicio anterior) como `myScore++` o incluso `myScore += 1`. El operador de incremento (++) es una forma simplificada de expresar `myScore = myScore + 1`, ya que manipula un incremento y una asignación al mismo tiempo. En el siguiente código ActionScript puede verse un ejemplo de operador de incremento:

```
var myNum:Number = 0;
myNum++;
trace(myNum); // 1
myNum++;
trace(myNum); // 2
```

Observe que el fragmento de código anterior carece de operadores de asignación. Por contra, utiliza el operador de incremento.

Puede manipular el valor de una variable empleando operadores mientras una condición sea `true`. Por ejemplo, puede utilizar el operador de incremento (++) para incrementar la variable `i` mientras la condición sea verdadera. En el siguiente código, la condición es `true` mientras `i` sea menor que 10. Mientras esto sea verdad (true), `i` se incrementa en un número mediante `i++`.

```
var i:Number;
for (i = 1; i < 10; i++) {
  trace(i);
}
```

El panel Salida muestra los números 1 a 9, que es el incremento del valor de `i` hasta alcanzar la condición de finalización (`i` es igual a 10) cuando se detiene. El último valor mostrado es 9. Por consiguiente, el valor de `i` es 1 cuando el archivo SWF empieza a reproducirse y 9 al finalizar trace.

Para más información sobre condiciones y bucles, consulte "Sentencias" en la página 106.

## Precedencia y asociatividad de operadores

Al utilizar dos o más operadores en una sentencia, algunos operadores tienen precedencia sobre otros operadores. La precedencia y asociatividad de los operadores determina el orden en que se procesan los operadores. ActionScript tiene una jerarquía que determina qué operadores se ejecutan antes que otros. Al final de esta sección encontrará una tabla en la que se expone esta jerarquía.

Aunque para aquellos usuarios familiarizados con la programación aritmética o básica puede parecer algo natural que el compilador procese el operador de multiplicación (*) antes que el operador de suma (+), el compilador necesita instrucciones explícitas sobre qué operadores debe procesar primero. Dichas instrucciones se conocen colectivamente como *precedencia de operadores*.

Puede ver un ejemplo de precedencia de operadores al utilizar los operadores de multiplicación y suma:

```
var mySum:Number;
mySum = 2 + 4 * 3;
trace(mySum); // 14
```

Observará que el resultado de esta sentencia es 14, ya que la multiplicación tiene una precedencia de operador superior. Por consiguiente, se calcula primero 4 * 3 y el resultado se suma a 2.

Puede controlar lo que debe ocurrir colocando las expresiones entre paréntesis. ActionScript establece una precedencia de operadores predeterminada que puede modificar utilizando el operador de paréntesis (`()`). Al colocar la expresión de suma entre paréntesis, ActionScript calcula primero la suma:

```
var mySum:Number;
mySum = (2 + 4) * 3;
trace(mySum); // 18
```

Ahora el resultado de esta sentencia es `18`.

Los operadores también pueden tener la misma precedencia. En este caso, la asociatividad determina el orden en que deben actuar los operadores. La asociatividad puede ser de izquierda a derecha o de derecha a izquierda.

Observe de nuevo el operador de multiplicación. La asociatividad en este caso es de izquierda a derecha, por lo que las dos sentencias siguientes son iguales.

```
var mySum:Number;
var myOtherSum:Number;
mySum = 2 * 4 * 3;
myOtherSum = (2 * 4) * 3;
trace(mySum); // 24
trace(myOtherSum); // 24
```

Pueden darse situaciones en las que dos o más operadores con la misma precedencia aparezcan en la misma expresión. En estos casos, el compilador utiliza las reglas de *asociatividad* para determinar qué operador se procesa primero. Todos los operadores binarios, salvo los operadores de asignación, tienen *asociatividad desde la izquierda*, lo que significa que los operadores de la izquierda se procesan antes que los operadores de la derecha. Los operadores de asignación y el operador condicional (`?:`) tienen *asociatividad desde la derecha*, lo que significa que los operadores de la derecha se procesan antes que los operadores de la izquierda. Para más información sobre operadores de asignación, consulte "Utilización de operadores de asignación" en la página 161. Para más información sobre el operador condicional (?:), consulte "El operador condicional" en la página 167.

Piense, por ejemplo, en los operadores menor que (<) y mayor que (>), que tienen la misma precedencia. Si ambos operadores se utilizan en la misma expresión, el operador de la izquierda se procesará en primer lugar porque ambos operadores tienen asociatividad desde la izquierda. Esto significa que las dos sentencias siguientes generan el mismo resultado:

```
trace(3 > 2 < 1);   // false
trace((3 > 2) < 1); // false
```

El operador mayor que (>) se procesa primero, lo que da como resultado el valor `true` porque el operando `3` es mayor que el operando `2`. El valor `true` se pasa al operador menor que (<) junto con el operando `1`. El operador menor que (<) convierte el valor `true` en el valor numérico `1` y compara el valor numérico con el segundo operando `1` para devolver el valor `false` (el valor `1` no es menor que `1`).

Estudie el orden de los operandos del código ActionScript, especialmente si establece condiciones complejas y conoce la frecuencia con la que dichas condiciones son verdaderas (true). Por ejemplo, si sabe `i` va a ser mayor que 50 en su condición, deberá escribir primero `i<50`. De esta forma, se comprobará primero, con lo que no será necesario comprobar la segunda condición con tanta frecuencia.

En la siguiente tabla se enumeran todos los operadores de ActionScript y su asociatividad, ordenados de la precedencia más alta a la más baja. Para más información y directrices sobre el uso de operadores y paréntesis, consulte el Capítulo 19, "Aplicación de formato a la sintaxis de ActionScript", en la página 808.

| Operador | Descripción | Asociatividad |
|---|---|---|
| | **Precedencia más alta** | |
| *x*++ | Incremento posterior | De izquierda a derecha |
| *x*-- | Decremento posterior | De izquierda a derecha |
| . | Acceso a propiedades del objeto | De izquierda a derecha |
| [ ] | Elemento de matriz | De izquierda a derecha |
| ( ) | Paréntesis | De izquierda a derecha |
| function ( ) | Llamada a función | De izquierda a derecha |
| ++*x* | Incremento previo | De derecha a izquierda |
| --*x* | Decremento previo | De derecha a izquierda |
| - | Unario negativo, como por ejemplo x = -1 | De izquierda a derecha |
| ~ | NOT en modo bit | De derecha a izquierda |

| Operador | Descripción | Asociatividad |
|---|---|---|
| `!` | NOT lógico | De derecha a izquierda |
| `new` | Asignar objeto | De derecha a izquierda |
| `delete` | Anular la asignación de objeto | De derecha a izquierda |
| `typeof` | Tipo de objeto | De derecha a izquierda |
| `void` | Devuelve un valor no definido | De derecha a izquierda |
| `*` | Multiplicar | De izquierda a derecha |
| `/` | Dividir | De izquierda a derecha |
| `%` | Módulo | De izquierda a derecha |
| `+` | Más unario | De derecha a izquierda |
| `-` | Menos unario | De derecha a izquierda |
| `<<` | Desplazamiento a la izquierda en modo bit | De izquierda a derecha |
| `>>` | Desplazamiento a la derecha en modo bit | De izquierda a derecha |
| `>>>` | Desplazamiento a la derecha en modo bit (sin signo) | De izquierda a derecha |
| `instanceof` | Instancia de (busca la clase de la que el objeto es una instancia)<br>Requiere Flash Player 6 o posterior | De izquierda a derecha |
| `<` | Menor que | De izquierda a derecha |
| `<=` | Menor o igual que | De izquierda a derecha |
| `>` | Mayor que | De izquierda a derecha |

| Operador | Descripción | Asociatividad |
|---|---|---|
| >= | Mayor o igual que | De izquierda a derecha |
| == | Igual | De izquierda a derecha |
| != | Distinto | De izquierda a derecha |
| & | AND en modo bit | De izquierda a derecha |
| ^ | XOR en modo bit | De izquierda a derecha |
| \| | OR en modo bit | De izquierda a derecha |
| && | AND lógico | De izquierda a derecha |
| \|\| | OR lógico | De izquierda a derecha |
| ?: | Condicional | De derecha a izquierda |
| = | Asignación | De derecha a izquierda |
| *=, /=, %=, +=, -=, &=, \|=, ^=, <<=, >>=, >>>= | Asignación compuesta | De derecha a izquierda |
| , | Coma | De izquierda a derecha |

**Precedencia más baja**

## Utilización de operadores con cadenas

Los operadores de comparación solamente comparan cadenas si ambos operandos son cadenas. Una excepción a esta regla es el operador de igualdad estricta (===). Si solamente uno de los operandos es una cadena, ActionScript convierte ambos operandos en números y realiza una comparación numérica. Para más información sobre operadores numéricos, consulte "Utilización de operadores numéricos" en la página 156.

Salvo en el caso del operador de igualdad (==), los operadores de comparación (>, >=, < y <=) afectan a las cadenas de manera diferente a cuando operan en otros valores.

Los operadores de comparación comparan cadenas para determinar cuál de ellas es la primera por orden alfabético. Las cadenas con caracteres en mayúsculas preceden a las cadenas en minúsculas. Dicho de otro modo, "Egg" va antes que "chicken".

```
var c:String = "chicken";
var e:String = "Egg";
trace(c < e); // false
var riddleArr:Array = new Array(c, e);
trace(riddleArr); // chicken,Egg
trace(riddleArr.sort()); // Egg,chicken
```

En este código ActionScript, el método `sort()` de la clase Array reordena el contenido de la matriz alfabéticamente. Observará que el valor "Egg" va antes que el valor "chicken", ya que la E mayúscula va antes que la c minúscula. Si desea comparar las cadenas con independencia de si están en mayúsculas o en minúsculas, deberá convertir las cadenas a mayúsculas o minúsculas antes de compararlas. Para más información sobre operadores de comparación, consulte "Operadores de igualdad" en la página 157 y "Utilización de operadores relacionales y de igualdad" en la página 158.

Puede utilizar los métodos `toLowerCase()` o `toUpperCase()` para convertir cadenas a mayúsculas o minúsculas antes de compararlas. En el siguiente ejemplo, ambas cadenas se convierten a minúsculas y se comparan, por lo que chicken va a ahora antes que egg:

```
var c:String = "chicken";
var e:String = "Egg";
trace(c.toLowerCase() < e.toLowerCase()); // true
```

| NOTA | Los operadores de comparación sólo comparan dos cadenas. Por ejemplo, los operadores no comparan los valores si un operando es un valor numérico. Si solamente uno de los operandos es una cadena, ActionScript convierte ambos operandos en números y realiza una comparación numérica. |
|---|---|

Puede utilizar operadores para manipular cadenas. Puede utilizar el operador de suma (+) para concatenar operandos que sean cadenas. Puede que ya haya utilizado el operador de suma para concatenar cadenas al escribir sentencias `trace`. Por ejemplo, podría escribir lo siguiente:

```
var myNum:Number = 10;
trace("The variable is " +  myNum + ".");
```

Al probar este código, el panel Salida mostrará lo siguiente:

```
The variable is 10.
```

En el siguiente ejemplo, la sentencia `trace` utiliza el operador + para concatenar en lugar de sumar. Al manipular cadenas y números, a veces Flash concatena en lugar de sumar numéricamente.

Por ejemplo, podría concatenar dos cadenas de variables diferentes en un único campo de texto. En el siguiente código ActionScript, la variable `myNum` se concatena con una cadena y dicha cadena se muestra en el campo de texto `myTxt` del escenario.

```
this.createTextField("myTxt", 11, 0, 0, 100, 20);
myTxt.autoSize = "left";
var myNum:Number = 10;
myTxt.text = "One carrot. " + myNum + " large eggplants.";
myTxt.text += " Lots of vegetable broth.";
```

Este código da el siguiente resultado en un campo de texto con el nombre de instancia myTxt:

```
One carrot. 10 large eggplants. Lots of vegetable broth.
```

En el ejemplo anterior, se muestra cómo utilizar los operadores de suma (+) y asignación de suma (+=) para concatenar cadenas. Observe cómo la tercera línea del código utiliza el operador de suma para concatenar el valor de la variable `myNum` en el campo de texto, mientras que la cuarta línea del código utiliza el operador de asignación de suma para concatenar una cadena con el valor existente del campo de texto.

Si sólo uno de los operandos de cadena de texto es realmente una cadena, Flash convierte el otro operando en una cadena. Por consiguiente, el valor de `myNum` se convierte en una cadena en el ejemplo anterior.

| NOTA | ActionScript trata los espacios del comienzo o del final de una cadena como parte literal de la cadena. |
|---|---|

# Utilización de operadores de punto y de acceso a una matriz

Puede utilizar el operador de punto (`.`) y el operador de acceso a una matriz (`[]`) para acceder a propiedades incorporadas en ActionScript o personalizadas. Los operadores de punto se utilizan para hacer referencia a determinados índices de un objeto. Por ejemplo, si tiene un objeto que contiene diversos datos del usuario, puede especificar un determinado nombre de clave en el operador de acceso a la matriz para recuperar un nombre de usuario, como se muestra en el siguiente código ActionScript:

```
var someUser:Object = {name:"Hal", id:2001};
trace("User's name is: " + someUser["name"]); // El nombre del usuario es:
  Hal
trace("User's id is: " + someUser["id"]); // El ID del usuario es: 2001
```

Por ejemplo, en el siguiente código ActionScript se utiliza el operador de punto para establecer determinadas propiedades en los objetos:

```
myTextField.border = true;
year.month.day = 9;
myTextField.text = "My text";
```

El operador de punto y el operador de acceso a una matriz son muy similares. El operador de punto toma un identificador como propiedad, mientras que el operador de acceso a una matriz calcula el contenido y da como resultado un nombre y, seguidamente, accede al valor de dicha propiedad con nombre. El operador de acceso a una matriz permite establecer dinámicamente y recuperar nombres de instancia y variables.

El operador de acceso a una matriz es útil si no conoce exactamente qué claves hay en un objeto. Cuando ocurre esto, puede utilizar un bucle `for..in` para repetir un objeto o un clip de película y mostrar su contenido.

### Para utilizar operadores de punto y de acceso a una matriz:

1. En un nuevo documento de Flash, cree un clip de película en la línea de tiempo principal.
2. Seleccione el clip de película y abra el inspector de propiedades.
3. Escriba un nombre de instancia de myClip.
4. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

   ```
   myClip.spam = 5;
   trace(myClip.spam); // 5
   ```

   Si desea establecer un valor en la instancia myClip de la línea de tiempo principal, puede utilizar los operadores de punto y de acceso a una matriz, como se muestra en este código ActionScript. Si escribe una expresión dentro del operador de acceso a una matriz, se calcula dicha expresión en primer lugar y se utiliza el resultado como nombre de variable.

5. Seleccione Control > Probar película para probar el documento.

   El panel Salida muestra 5.

6. Regrese al entorno de edición y reemplace la primera línea del código ActionScript por lo siguiente:

```
myClip["spam"] = 10;
```

7. Seleccione Control > Probar película para probar el documento.

   El panel Salida muestra 10.

8. Regrese al entorno de edición y haga doble clic en la instancia myClip.

9. Añada cuatro instancias nuevas dentro de la instancia myClip.

10. Utilice el inspector de propiedades para añadir los siguientes nombres de instancia a cada una de las cuatro instancias nuevas: **nestedClip1**, **nestedClip2**, **nestedClip3**, **nestedClip4**.

11. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
var i:Number;
for (i = 1; i <= 4; i++) {
  myClip["nestedClip" + i]._visible = false;
}
```

    Este código ActionScript activa o desactiva la visibilidad de los clip de película anidados.

12. Seleccione Control > Probar película para probar el código ActionScript que acaba de añadir.

    Las cuatro instancias anidadas son ahora invisibles. El operador de acceso a una matriz le permite repetir la ejecución en cada uno de los clip de película anidados en la instancia myClip y establecer su propiedad visible dinámicamente. Esto le permite ahorrar tiempo, ya que no tiene que hacer referencia a cada una de las instancias específicamente.

También puede utilizar el operador de acceso a una matriz en el lado izquierdo de una asignación, lo que le permite establecer los nombres de instancia, variables y objetos dinámicamente:

```
myNum[i] = 10;
```

En ActionScript 2.0, puede utilizar el operador de corchete para acceder a las propiedades de un objeto que se crean dinámicamente, en el caso de que no se asigne el atributo `dynamic` a la definición de clase de dicho objeto. También puede crear matrices multidimensionales empleando este operador. Para más información sobre la creación de matrices multidimensionales empleando operadores de acceso a una matriz, consulte "Creación de matrices multidimensionales" en la página 135.

## Operadores de sufijo

Los operadores de sufijo toman un operador y aumentan o reducen el valor del operador. Aunque estos operadores son unarios, se clasifican independientemente del resto de operadores unarios debido a su mayor precedencia y a su comportamiento especial. Para información sobre operadores unarios, consulte "Operadores unarios" en la página 154.

Al utilizar un operador de sufijo como parte de una expresión mayor, el valor de la expresión se devuelve antes de que se procese el operador de sufijo. Por ejemplo, el siguiente código muestra cómo se devuelve el valor de la expresión `xNum++` antes de que se incremente el valor.

```
var xNum:Number = 0;
trace(xNum++); // 0
trace(xNum); // 1
```

Al trazar este código, el texto del panel Salida es el siguiente:

```
0
1
```

Los operadores de esta tabla tienen idéntica precedencia:

| Operador | Operación realizada |
|---|---|
| `++` | Incremento (sufijo) |
| `--` | Decremento (sufijo) |

## Operadores unarios

Los operadores unarios utilizan un operando. Los operadores de incremento (++) y decremento (`--`) de este grupo son operadores de *prefijo*, lo que significa que aparecen delante del operando en una expresión. También pueden aparecer tras el operando, en cuyo caso son operadores *de sufijo*. Para más información sobre operadores de sufijo, consulte "Operadores de sufijo" en la página 154.

Los operadores de prefijo difieren de los correspondientes operadores de sufijo en que la operación de incremento o decremento se realiza antes de que se devuelva el valor de la expresión global. Por ejemplo, el siguiente código muestra cómo se devuelve el valor de la expresión `xNum++` después de que se incremente el valor.

```
var xNum:Number = 0;
trace(++xNum); // 1
trace(xNum); // 1
```

Todos los operadores de esta tabla tienen idéntica precedencia:

| Operador | Operación realizada |
|---|---|
| `++` | Incremento (prefijo) |
| `--` | Decremento (prefijo) |
| `+` | + unario |
| `!` | - unario (negación) |
| `typeof` | Devuelve información de tipo |
| `void` | Devuelve un valor no definido |

## Operadores multiplicativos

Los operadores multiplicativos toman dos operandos y realizan cálculos de multiplicación, división o módulo. Entre los operadores numéricos también figuran los operadores aditivos. Para información sobre operadores aditivos, consulte "Operadores aditivos" en la página 155.

Todos los operadores de esta tabla tienen idéntica precedencia:

| Operador | Operación realizada |
|---|---|
| `*` | Multiplicación |
| `/` | División |
| `%` | Módulo |

Para más información sobre la utilización de operadores multiplicativos, consulte "Utilización de operadores numéricos" en la página 156.

## Operadores aditivos

Los operadores aditivos toman dos operandos y realizan cálculos de suma y resta. Entre los operadores numéricos también figuran los operadores multiplicativos. Para información sobre operadores multiplicativos, consulte "Operadores multiplicativos" en la página 155.

Los operadores de esta tabla tienen idéntica precedencia:

| Operador | Operación realizada |
|---|---|
| `+` | Suma |
| `-` | Resta |

Para más información sobre la utilización de operadores aditivos, consulte "Utilización de operadores numéricos" en la página 156.

# Utilización de operadores numéricos

Los operadores numéricos se utilizan para sumar, restar, dividir y multiplicar valores en ActionScript. Puede realizar diferentes tipos de operaciones aritméticas. Uno de los operadores más comunes es el operador de incremento, que habitualmente adopta la forma `i++`. Este operador también permite realizar otras operaciones. Para más información sobre el operador de incremento, consulte "Utilización de operadores para manipular valores" en la página 144.

Puede añadir el incremento antes (*preincremento*) o después (*posincremento*) de un operando.

### Para comprender los operadores numéricos de ActionScript:

1. Cree un nuevo documento de Flash.
2. Escriba el siguiente código ActionScript en el fotograma 1 de la línea de tiempo:

```
// ejemplo uno
var firstScore:Number = 29;
if (++firstScore >= 30) {
  // debe trazar
  trace("Success! ++firstScore is >= 30");
}
// ejemplo dos
var secondScore:Number = 29;
if (secondScore++ >= 30) {
  // no debe trazar
  trace("Success! secondScore++ is >= 30");
}
```

3. Seleccione Control > Probar película para probar el código ActionScript.

   El bloque de código del "Ejemplo uno" se traza, mientras que el bloque de código del "Ejemplo dos" no se traza. El primer ejemplo utiliza un preincremento (`++firstScore`) para incrementar y calcular `firstScore` antes de probarlo con 30. Por consiguiente, `firstScore` se incrementa a 30 y luego se prueba con 30.

   Sin embargo, el Ejemplo dos utiliza un posincremento (`secondScore++`), que se calcula después de realizar la prueba. Por consiguiente, 29 se compara con 30 y luego se incrementa a 30 tras la evaluación.

Para más información sobre la precedencia de los operadores, consulte "Precedencia y asociatividad de operadores" en la página 145.

Al cargar datos de fuentes externas (por ejemplo, de archivos XML, FlashVars, servicios Web, etc.), deberá extremar los cuidados cuando utilice operadores numéricos. A veces, Flash trata los números como cadenas porque el archivo SWF no conoce el tipo de datos de los números. En este caso, podría sumar 3 y 7 y obtener como resultado 37 porque ambos números se concatenan como cadenas en lugar de sumarse numéricamente. En este caso, tendrá que convertir manualmente los datos de cadenas a números empleando la función `Number()`.

## Operadores relacionales

Los operadores relacionales toman dos operandos, comparan sus valores y devuelven un valor booleano. Todos los operadores de esta tabla tienen idéntica precedencia:

| Operador | Operación realizada |
|---|---|
| `<` | Menor que |
| `>` | Mayor que |
| `<=` | Menor o igual que |
| `>=` | Mayor o igual que |
| `instanceof` | Comprueba una cadena prototipo |
| `in` | Comprueba las propiedades de objetos |

Para más información sobre la utilización de operadores relacionales, consulte "Utilización de operadores relacionales y de igualdad" en la página 158.

## Operadores de igualdad

Los operadores de igualdad toman dos operandos, comparan sus valores y devuelven un valor booleano. Todos los operadores de esta tabla tienen idéntica precedencia:

| Operador | Operación realizada |
|---|---|
| `==` | Igualdad |
| `!=` | Desigualdad |
| `===` | Igualdad estricta |
| `!==` | Desigualdad estricta |

Para más información sobre la utilización de operadores de igualdad, consulte "Utilización de operadores relacionales y de igualdad" en la página 158.

# Utilización de operadores relacionales y de igualdad

Los operadores relacionales y de igualdad, también conocidos como *operadores de comparación*, comparan valores de expresiones y devuelven `true` o `false` (un valor booleano). Los operadores de comparación se utilizan con frecuencia en sentencias condicionales y bucles para especificar la condición con la que debe detenerse el bucle.

El operador de igualdad (==) se emplea para averiguar si los valores o referencias de dos operandos son iguales; el resultado de esta comparación devuelve un valor booleano. Los valores de operandos de cadena, número o booleanos comparan utilizando un valor. Los operandos de objeto y matriz se comparan mediante una referencia.

En este ejemplo, se muestra cómo se utiliza el operador de igualdad para comprobar la longitud de la matriz y mostrar un mensaje en el panel Salida si no hay elementos en la matriz.

```
var myArr:Array = new Array();
if (myArr.length == 0) {
  trace("the array is empty.");
}
```

Al seleccionar Control > Probar película, la cadena `the array is empty` aparece en el panel Salida.

Puede utilizar el operador de igualdad para comparar valores, pero no puede utilizarlo para establecer valores. Pruebe el operador de asignación (=) para comprobar la igualdad.

### Para utilizar operadores relacionales y de igualdad en el código:

1. Cree un nuevo documento de Flash.
2. Escriba el siguiente código ActionScript en el fotograma 1 de la línea de tiempo:

   ```
   var myNum:Number = 2;
   if (myNum == 2) {
     // realizar una acción
     trace("It equals 2");
   }
   ```

   En este código ActionScript, el operador de igualdad (==) se utiliza para comprobar la igualdad. Se comprueba si la variable `myNum` es igual a 2.
3. Seleccione Control > Probar película.

   La cadena `It equals 2` aparece en el panel Salida.
4. Regrese al entorno de edición y cambie:

   ```
   var myNum:Number = 2;
   ```

   a:

   ```
   var myNum:Number = 4;
   ```

5. Seleccione de nuevo Control > Probar película.

   La cadena `It equals 2` no aparece en el panel Salida.

6. Regrese al entorno de edición y cambie:

   ```
   if (myNum == 2) {
   ```

   a

   ```
   if (myNum = 2) {
   ```

7. Seleccione de nuevo Control > Probar película.

   La cadena `It equals 2` aparece de nuevo en el panel Salida.

   En el paso 6, se asigna el valor `2` a `myNum`, en lugar de comparar `myNum` con `2`. En este caso, la sentencia `if` se ejecuta con independencia del valor anterior de `myNum`, lo que puede originar resultados inesperados al probar el documento de Flash.

   Para más información sobre la utilización correcta del operador de asignación, consulte "Utilización de operadores de asignación" en la página 161.

El operador de igualdad estricta (===) es parecido al operador de igualdad, con la diferencia de que no realiza conversión de tipos. Si dos operandos son de distinto tipo, el operador de igualdad devuelve `false`. El operador de desigualdad estricta (!==) devuelve el contrario del operador de igualdad estricta.

El siguiente código ActionScript muestra la diferencia clave entre el operador de igualdad (==) y el operador de igualdad estricta (===):

```
var num1:Number = 32;
var num2:String = new String("32");
trace(num1 == num2); // true
trace(num1 === num2); // false
```

En primer lugar, se definen variables numéricas: `num1` y `num2`. Si compara las variables utilizando el operador de igualdad, Flash intenta convertir los valores al mismo tipo de datos y luego compara los valores para comprobar si son iguales. Al utilizar el operador de igualdad estricta (===), Flash no intenta realizar ninguna conversión de tipos de datos antes de comparar los valores. Como resultado, Flash considera las variables como dos valores independientes.

En el siguiente ejemplo, se utiliza el operador mayor o igual que (>=) para comparar valores y ejecutar un código en función del valor que introduzca el usuario en un campo de texto.

### Para utilizar el operador mayor o igual que en el código:

1. Seleccione Archivo > Nuevo y, a continuación, seleccione Documento de Flash para crear un nuevo archivo FLA.
2. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
this.createTextField("myTxt", 20, 0, 0, 100, 20);
myTxt.type = "input";
myTxt.border = true;
myTxt.restrict = "0-9";

this.createEmptyMovieClip("submit_mc", 30);
submit_mc.beginFill(0xFF0000);
submit_mc.moveTo(0, 0);
submit_mc.lineTo(100, 0);
submit_mc.lineTo(100, 20);
submit_mc.lineTo(0, 20);
submit_mc.lineTo(0, 0);
submit_mc.endFill();
submit_mc._x = 110;

submit_mc.onRelease = function(evt_obj:Object):Void {
  var myNum:Number = Number(myTxt.text);
  if (isNaN(myNum)) {
    trace("Please enter a number");
    return;
  }
  if (myNum >= 10) {
    trace("Your number is greater than or equal to 10");
  } else {
    trace("Your number is less than 10");
  }
};
```

3. Seleccione Control > Probar película para probar el código ActionScript.

   También puede comprobar si determinadas condiciones son verdaderas (true) y ejecutar un bloque de código alternativo si la condición no es verdadera.
4. Cambie la condición en el código ActionScript por lo siguiente.

```
if (myNum == 10) {
  trace("Your number is 10");
} else {
    trace("Your number is not 10");
}
```

5. Seleccione Control > Probar película para probar de nuevo el código ActionScript.

Salvo en el caso del operador de igualdad estricta (===), los operadores de comparación comparan cadenas sólo si ambos operandos son cadenas. Si solamente uno de los operandos es una cadena, se convierten ambos operandos en números y se realiza una comparación numérica. Para más información sobre cadenas y operadores, consulte "Utilización de operadores con cadenas" en la página 150. Para obtener información sobre cómo afecta el orden o la precedencia de los operadores al código ActionScript, consulte "Precedencia y asociatividad de operadores" en la página 145.

## Operadores de asignación

Los operadores de asignación toman dos operandos y asignan un valor a un operando en función del valor del otro operando. Todos los operadores de esta tabla tienen idéntica precedencia:

| Operador | Operación realizada |
|---|---|
| = | Asignación |
| *= | Asignación de multiplicación |
| /= | Asignación de división |
| %= | Asignación de módulo |
| += | Asignación de suma |
| -= | Asignación de resta |
| <<= | Asignación de desplazamiento a la izquierda en modo bit |
| >>= | Asignación de desplazamiento a la derecha en modo bit |
| >>>= | Asignación de desplazamiento a la derecha en modo bit sin signo |
| &= | Asignación de AND en modo bit |
| ^= | Asignación de XOR en modo bit |
| \|= | Asignación de OR en modo bit |

Para más información sobre la utilización de operadores de asignación, consulte "Utilización de operadores de asignación" en la página 161.

## Utilización de operadores de asignación

El operador de asignación (=) permite asignar un valor determinado a una variable. Puede asignar una cadena a una variable de la siguiente forma:

```
var myText:String = "ScratchyCat";
```

También puede utilizar el operador de asignación para asignar valores a diversas variables en la misma expresión. En la siguiente sentencia, se asigna el valor 10 a las variables `numOne`, `numTwo` y `numThree`.

```
var numOne:Number;
var numTwo:Number;
var numThree:Number;
numOne = numTwo = numThree = 10;
```

También puede utilizar operadores de asignación compuestos para combinar operaciones. Estos operadores actúan sobre los dos operandos y después asignan un nuevo valor al primer operando. Por ejemplo, estas dos sentencias hacen lo mismo:

```
var myNum:Number = 0;
myNum += 15;
myNum = myNum + 15;
```

Al utilizar el operador de asignación, puede tener problemas si intenta sumar valores de una expresión, como puede observarse en el siguiente ejemplo:

```
trace("the sum of 5 + 2 is: " + 5 + 2); // la suma de 5 + 2 es: 52
```

Flash concatena los valores `5` y `2` en lugar de sumarlos. Para resolver este problema, puede colocar la expresión `5+2` entre paréntesis, como se muestra en el siguiente código:

```
trace("the sum of 5 + 2 is: " + (5 + 2)); // la suma de 5 + 2 es: 7
```

## Operadores lógicos

Los operadores lógicos permiten comparar valores booleanos (`true` y `false`) y, seguidamente, devuelven un valor booleano en función del resultado de dicha comparación. Por ejemplo, si tiene dos operandos que dan como resultado `true`, el operador lógico AND (`&&`) devuelve `true`. O bien, si uno o ambos operandos dan como resultado `true`, el operador lógico OR (`||`) devuelve `true`.

Los operadores lógicos toman dos operandos y devuelven un resultado booleano. Los operadores lógicos, que tienen una precedencia diferente, se enumeran en esta tabla por orden decreciente de precedencia:

| Operador | Operación realizada |
|---|---|
| `&&` | AND lógico |
| `\|\|` | OR lógico |

Para más información sobre la utilización de operadores lógicos, consulte "Utilización de operadores lógicos" en la página 163.

## Utilización de operadores lógicos

Los operadores lógicos se utilizan a menudo con los operadores de comparación para determinar la condición de una sentencia `if`. Esto se muestra en el siguiente ejemplo.

### Para utilizar operadores lógicos en el código:

1. Seleccione Archivo > Nuevo para crear un documento de Flash nuevo.
2. Abra el panel Acciones y escriba el siguiente código ActionScript en el fotograma 1 de la línea de tiempo:

```
this.createTextField("myTxt", 20, 0, 0, 100, 20);
myTxt.type = "input";
myTxt.border = true;
myTxt.restrict = "0-9";

this.createEmptyMovieClip("submit_mc", 30);
submit_mc.beginFill(0xFF0000);
submit_mc.moveTo(0, 0);
submit_mc.lineTo(100, 0);
submit_mc.lineTo(100, 20);
submit_mc.lineTo(0, 20);
submit_mc.lineTo(0, 0);
submit_mc.endFill();
submit_mc._x = 110;

submit_mc.onRelease = function():Void {
  var myNum:Number = Number(myTxt.text);
  if (isNaN(myNum)) {
    trace("Please enter a number");
    return;
  }
  if ((myNum > 10) && (myNum < 20)) {
    trace("Your number is between 10 and 20");
  } else {
    trace("Your number is NOT between 10 and 20");
  }
};
```

En este código ActionScript, se crea un campo de texto durante la ejecución. Si escribe un número en el campo de texto y hace clic en el botón del escenario, Flash utilizará el operador lógico para mostrar un mensaje en el panel Salida. El mensaje depende del valor del número que escriba en el campo de texto.

Al utilizar operandos, debe tener cuidado con el orden de éstos, especialmente si utiliza condiciones complejas. En el siguiente fragmento, se observa cómo se utiliza el operador lógico AND para comprobar que un número está comprendido entre 10 y 20. En función del resultado, se muestra el mensaje más adecuado. Si el número es menor que 10 o mayor que 20, se muestra un mensaje alternativo en el panel Salida.

```
submit_mc.onRelease = function():Void {
  var myNum:Number = Number(myTxt.text);
  if (isNaN(myNum)) {
    trace("Please enter a number");
    return;
  }
  if ((myNum > 10) && (myNum < 20)) {
    trace("Your number is between 10 and 20");
  } else {
    trace("Your number is NOT between 10 and 20");
  }
};
```

## Operadores de desplazamiento en modo bit

Los operadores de desplazamiento en modo bit toman dos operandos y desplazan los bits del primer operando según lo especificado por el segundo operando. Todos los operadores de esta tabla tienen idéntica precedencia:

| Operador | Operación realizada |
|---|---|
| << | Desplazamiento a la izquierda en modo bit |
| >> | Desplazamiento a la derecha en modo bit |
| >>> | Desplazamiento a la derecha en modo bit sin signo |

Para más información sobre la utilización de operadores en modo bit, consulte "Utilización de operadores en modo bit" en la página 165. Para obtener información específica sobre cada operador en modo bit, consulte su entrada en *Referencia del lenguaje ActionScript 2.0*.

## Operadores lógicos en modo bit

Los operadores lógicos en modo bit toman dos operandos y realizan operaciones lógicas a nivel de bits. Los operadores lógicos en modo bit, que tienen una precedencia diferente, se enumeran en esta tabla por orden decreciente de precedencia:

| Operador | Operación realizada |
|---|---|
| `&` | AND en modo bit |
| `^` | XOR en modo bit |
| `\|` | OR en modo bit |

Para más información sobre la utilización de operadores en modo bit, consulte "Utilización de operadores en modo bit" en la página 165. Para obtener información específica sobre cada operador en modo bit, consulte su entrada en *Referencia del lenguaje ActionScript 2.0*.

## Utilización de operadores en modo bit

Los operadores en modo bit manipulan internamente los números de coma flotante para cambiarlos por enteros de 32 bits. La operación exacta realizada depende del operador, pero todas las operaciones en modo bit calculan el resultado de cada dígito binario (bit) del entero de 32 bits de forma individual para calcular un nuevo valor. Para obtener una lista de operadores de desplazamiento en modo bit, consulte "Operadores de desplazamiento en modo bit" en la página 164. Para obtener una lista de operadores lógicos en modo bit, consulte "Operadores lógicos en modo bit" en la página 165.

La utilización de operadores en modo bit en Flash no es muy habitual, aunque pueden resultar útiles en determinadas circunstancias. Por ejemplo, puede que desee crear una matriz de permisos para un proyecto de Flash pero sin crear variables independientes para cada tipo de permiso. En este caso, podría utilizar operadores en modo bit.

En el siguiente ejemplo se muestra cómo utilizar el operador en modo bit OR con el método `Array.sort()` para especificar las opciones de orden.

### Para utilizar el operador en modo bit OR:

1. Seleccione Archivo > Nuevo para crear un documento de Flash nuevo.
2. Introduzca el siguiente código ActionScript en el panel Acciones:

```
var myArr:Array = new Array("Bob", "Dan", "doug", "bill", "Hank",
  "tom");
trace(myArr); // Bob,Dan,doug,bill,Hank,tom
myArr.sort(Array.CASEINSENSITIVE | Array.DESCENDING);
trace(myArr); // tom,Hank,doug,Dan,Bob,bill
```

   La primera línea define una matriz de nombres aleatorios y los traza en el panel Salida. Posteriormente, se llama al método `Array.sort()` y se especifican dos opciones de orden mediante valores constantes `Array.CASEINSENSITIVE` y `Array.DESCENDING`. El resultado del método sort provoca que los elementos de la matriz se clasifiquen en orden inverso (de la z a la a). La búsqueda distingue entre mayúsculas y minúsculas; `a` y `A` se consideran equivalentes, en lugar de lugar de realizarse una búsqueda con distinción entre mayúsculas y minúsculas en la que `Z` va antes que `a`.

3. Seleccione Control > Probar película para probar el código ActionScript. El panel Salida muestra este texto:

```
Bob,Dan,doug,bill,Hank,tom
tom,Hank,doug,Dan,Bob,bill
```

Hay cinco opciones disponibles en el método sort:

- 1 o Array.CASEINSENSITIVE (binario = 1)
- 2 o Array.DESCENDING (binario = 10)
- 4 o Array.UNIQUESORT (binario = 100)
- 8 o Array.RETURNINDEXEDARRAY (binario = 1000)
- 16 o Array.NUMERIC (binario = 10000)

Existen tres maneras distintas de definir las opciones de orden de una matriz:

```
my_array.sort(Array.CASEINSENSITIVE | Array.DESCENDING); // constantes
my_array.sort(1 | 2); // números
my_array.sort(3); // suma de los números
```

Aunque pueda no resultar obvio de forma inmediata, los valores numéricos de las opciones de orden (sort) son en realidad dígitos en modo bit (binarios o en base 2). El valor constante `Array.CASEINSENSITIVE` es igual al valor numérico 1, que a su vez es el valor binario 1. El valor constante `Array.DECENDING` tiene el valor numérico 2 o el valor binario 10.

La utilización de números binarios puede resultar compleja. Sólo hay dos valores binarios posibles, 1 o 0, razón por la cual el valor 2 se representa como 10. Si desea mostrar el número 3 en forma binaria, éste sería 11 (1+10). La representación binaria del número 4 es 100, la del 5, 101 y, así, sucesivamente.

El siguiente código ActionScript muestra cómo ordenar una matriz de valores numéricos por orden descendente empleando el operador en modo bit AND para sumar las constantes `Array.DESCENDING` y `Array.NUMERIC`.

```
var scores:Array = new Array(100,40,20,202,1,198);
trace(scores); // 100,40,20,202,1,198
trace(scores.sort()); // 1,100,198,20,202,40
var flags:Number = Array.NUMERIC|Array.DESCENDING;
trace(flags); // 18 (base 10)
trace(flags.toString(2)); // 10010 (binario -- base2)
trace(scores.sort(flags)); // 202,198,100,40,20,1
```

## El operador condicional

El operador condicional es un operador ternario, lo que significa que toma tres operandos. El operador condicional es un método abreviado para la aplicación de la sentencia condicional `if..else`:

| Operador | Operación realizada |
|---|---|
| `?:` | Condicional |

Para obtener información sobre la utilización del operador condicional y ver un ejemplo, consulte "El operador condicional y sintaxis alternativa" en la página 117.

## Utilización de operadores en un documento

En el siguiente ejemplo, se utiliza el método `Math.round()` para redondear los cálculos a un número arbitrario de cifras decimales. Este método redondea el valor del parámetro `x` hacia arriba o hacia abajo con el entero más próximo y devuelve el valor resultante. Tras modificar ligeramente el código ActionScript, podrá hacer que Flash redondee los números a un número determinado de cifras decimales.

En el siguiente ejemplo, se utilizan operadores de división y multiplicación para calcular la puntuación de un usuario sobre el número de respuestas correctas dividido por el número total de preguntas formuladas. La puntuación del usuario puede multiplicarse por un número y mostrarse para obtener una puntuación entre 0% y 100%. Posteriormente, se utiliza el operador de suma para concatenar la puntuación del usuario en una cadena que se muestra en el panel Salida.

### Para utilizar operadores en ActionScript:

1. Cree un nuevo documento de Flash.
2. Escriba el siguiente código ActionScript en el fotograma 1 de la línea de tiempo principal:

```
var correctAnswers:Number = 11;
var totalQuestions:Number = 13;
//redondear al entero más próximo
//var score:Number = Math.round(correctAnswers / totalQuestions * 100);
//redondear a dos cifras decimales
var score:Number = Math.round(correctAnswers / totalQuestions * 100 *
  100) / 100;
trace("You got " + correctAnswers + " out of " + totalQuestions + "
  answers correct, for a score of " + score + "%.");
```

3. Seleccione Control > Probar película.

   El panel Salida muestra el siguiente texto:

   ```
   You got 11 out of 13 answers correct, for a score of 84.62%.
   ```

   Al llamar a `Math.round()` en este ejemplo, la puntuación se redondea con el entero más próximo (85) y se muestra en el panel Salida. Si multiplica el número por 100 antes de llamar a `Math.round()` y, seguidamente, divide por 100, hará que Flash redondee a 2 cifras decimales. Esto proporciona una puntuación más precisa.
4. Pruebe a cambiar el valor de la variable `correctAnswers` a 3 y seleccione Control > Probar película para probar de nuevo el archivo SWF.

Si está creando una aplicación de prueba, puede que desee crear una serie de preguntas cuya respuesta sea verdadero/falso o de varias opciones empleando los componentes RadioButton y Label. Una vez que los usuarios terminen de responder a cada una de las preguntas y hagan clic en el botón submit, podrá comparar las respuestas con una clave de respuestas y luego calcular la puntuación.

CAPÍTULO 5

# Funciones y métodos

5

Conocer las funciones es importante a la hora de escribir código ActionScript, crear clases y utilizar métodos. Existen diversos tipos de funciones con los que deberá trabajar. En este capítulo, conocerá las funciones y los métodos: aprenderá a utilizarlos en sus aplicaciones cuando emplee clases incorporadas, así como a escribirlos. En el Capítulo 6, "Clases", creará clases personalizadas, para lo cual deberá utilizar funciones con frecuencia. También aprenderá a escribir funciones en archivos de clases de ActionScript.

Se pueden utilizar funciones del código para añadir interactividad, animaciones y otros efectos a las aplicaciones. En este capítulo se tratan los tipos de funciones que se pueden escribir en las aplicaciones de Flash. Para obtener información sobre la definición de funciones y métodos, así como ejercicios en los que se escriben y utilizan funciones y métodos de Flash, consulte los siguientes temas:



## Funciones y métodos

Los métodos y funciones son bloques de código de ActionScript que pueden volver a utilizarse en cualquier parte de un archivo SWF. Puede escribir funciones en el archivo FLA o en un archivo de código ActionScript externo y, seguidamente, llamar a la función desde cualquier lugar de los documentos. Los métodos son simplemente funciones ubicadas dentro de una definición de clase de ActionScript. Puede definir funciones para que éstas ejecuten una serie de sentencias en función de los valores que se le han pasado. Sus funciones también pueden devolver valores. Una vez que se ha definido una función, podrá llamarla desde cualquier línea de tiempo, incluida la línea de tiempo de un archivo SWF que se ha cargado.

Si pasa valores a una función como parámetros, la función puede realizar cálculos empleando los valores facilitados. Cada función tiene sus propias características y algunas necesitan que se les pasen determinados tipos o números de valores. Si pasa más parámetros de los que requiere la función, la función ignora los valores sobrantes. Si no pasa un parámetro requerido, la función asigna el tipo de datos `undefined` a los parámetros vacíos. Esto puede provocar errores durante la ejecución. Una función también puede devolver valores (consulte "Devolución de valores de funciones" en la página 190).

| NOTA | Para llamar a una función, la definición de ésta debe encontrase en un fotograma al que haya llegado la cabeza lectora. |
|---|---|

Una función correctamente escrita puede compararse a una "caja negra". Si la función incluye comentarios acerca de sus entradas, salidas y su finalidad, no es necesario que la persona que utilice la función comprenda exactamente cómo funciona internamente.

La sintaxis básica de una *función con nombre* sencilla es:

```
function traceMe() {
  trace("su mensaje");
}
traceMe();
```

Para obtener información sobre la escritura de funciones con nombre, consulte "Escritura de funciones con nombre" en la página 175.

La sintaxis básica de una función con nombre sencilla que se crea según el ejemplo anterior al pasar un parámetro, `yourMessage`, es:

```
function traceMe(yourMessage:String) {
  trace(yourMessage);
}
traceMe("¿Qué tal?");
```

De forma alternativa, si desea pasar varios parámetros, podría utilizar el siguiente código:

```
var yourName:String = "Ester";
var yourAge:String = "65";
var favSoftware:String = "Flash";
function traceMe(favSoftware:String, yourName:String, yourAge:String) {
  trace("Me llamo " + yourName + ", me gusta" + favSoftware + " y tengo " +
  yourAge + ".");
}
traceMe(favSoftware,yourName,yourAge);
```

Para más información sobre cómo pasar parámetros, consulte "Paso de parámetros a una función" en la página 187.

Existen numerosos tipos de funciones que puede crear. Para más información sobre la escritura de funciones, así como los vínculos a las secciones sobre cómo escribir determinados tipos de funciones, consulte "Tipos de funciones y métodos" en la página 171. Para ver un ejemplo en el que se comparan métodos y funciones, consulte "Aspectos básicos de los métodos" en la página 192.

| NOTA | Para obtener información sobre la escritura de código con el asistente de script, consulte "Utilización del asistente de script para escribir ActionScript" en la página 362, "Creación de un evento startDrag/stopDrag con el Asistente de script" en la página 366 y el tutorial ActionScript: Utilización del modo de asistente de script (que comienza con "Apertura del documento inicial" en la página 229). |
|---|---|

Para más información sobre funciones y métodos, consulte los siguientes temas:

- "Tipos de funciones y métodos" en la página 171

## Tipos de funciones y métodos

Las funciones pertenecientes a una clase se conocen como los *métodos* de dicha clase. Puede utilizar distintos tipos de funciones en las aplicaciones, incluidas las funciones incorporadas, con nombre, definidas por el usuario, anónimas, callback, constructoras y literales. En las siguientes secciones se incluye información sobre cómo definir estas funciones.

También puede escribir funciones en un archivo de clase de ActionScript. Estas funciones se utilizan como métodos en los scripts. En el siguiente ejemplo, la clase Person muestra un método constructor, métodos de clase, métodos de instancia y métodos de descriptores de acceso (captadores y definidores). Los comentarios de este ejemplo de código muestran dónde tienen lugar estos métodos.

| NOTA | Para obtener información sobre la escritura de archivos de clase, como el siguiente, consulte el Capítulo 6, "Clases", en la página 195. |
|---|---|

```
class Person {
  public static var numPeople:Number = 0;

  // miembros de instancia
  private var _speed:Number;

  // constructor
  public function Person(speed:Number) {
    Person.numPeople++;
    this._speed = speed;
  }
```

```
  // métodos estáticos
  public static function getPeople():Number {
    return Person.numPeople;
  }

  // métodos de instancia
  public function walk(speed:Number):Void {
    this._speed = speed;
  }
  public function run():Void {
    this._speed *= 2;
  }
  public function rest():Void {
    this._speed = 0;
  }

  // captadores/definidores (métodos de descriptores de acceso)
  public function get speed():Number {
    return this._speed;
  }
}
```

Para una demostración completa sobre la forma de escribir métodos como los del ejemplo de código anterior, consulte el Capítulo 6, “Clases”, en la página 195. Los métodos que usted utiliza en el código podrían pertenecer a una clase que está incorporada en el lenguaje ActionScript. MovieClip y Math son ejemplos de clases de nivel superior que pueden utilizarse en una aplicación. Cuando utiliza métodos de estas clases en el código, se trata de funciones escritas en la clase incorporada (similar al ejemplo de código anterior). Como alternativa, puede utilizar los métodos de una clase personalizada escrita por usted.

Las funciones que no pertenecen a una clase se conocen como *funciones de nivel superior* (a veces se denominan *funciones predefinidas* o *incorporadas*), lo que significa que pueden llamarse sin un constructor. Ejemplos de funciones que están incorporadas en el nivel superior del lenguaje ActionScript son `trace()` y `setInterval()`.

Para añadir al código una llamada a una función de nivel superior, basta con añadir una única línea de código en el panel Script del panel Acciones. Por ejemplo, escriba lo siguiente:

```
trace("mi mensaje");
```

Al probar el archivo SWF con esta única línea de código, se llamará a la función de nivel superior `trace()` y aparecerá texto en el panel Salida.

Recuerde: cuando desee asignar un método a una propiedad, debe omitir los paréntesis después del nombre del método ya que está pasando una referencia a la función:

```
my_mc.myMethod = aFunction;
```

Sin embargo, cuando desee invocar un método en el código, deberá incluir los paréntesis tras el nombre del método:

```
my_mc.myMethod();
```

> **NOTA** Para más información sobre funciones de nivel superior, consulte "Funciones incorporadas y de nivel superior" en la página 173.

También puede definir funciones de muchas otras formas. Para más información sobre cada tipo de función, consulte las secciones siguientes:

- "Funciones incorporadas y de nivel superior" en la página 173
- "Escritura de funciones con nombre" en la página 175
- "Escritura de funciones anónimas y callback" en la página 176
- "Literales de función" en la página 179
- "Referencia y llamada a las funciones definidas por el usuario" en la página 181
- "Funciones constructoras" en la página 179

Para obtener información sobre la escritura y utilización de funciones y métodos, consulte las siguientes secciones relacionadas. Para obtener información sobre la utilización de funciones, consulte "Utilización de funciones en Flash" en la página 183. Para obtener información sobre la utilización de métodos, consulte "Aspectos básicos de los métodos" en la página 192.

> **NOTA** Para obtener información sobre la escritura de código con el asistente de script, consulte "Utilización del asistente de script para escribir ActionScript" en la página 362, "Creación de un evento startDrag/stopDrag con el Asistente de script" en la página 366 y el tutorial ActionScript: Utilización del modo de asistente de script (que comienza con "Apertura del documento inicial" en la página 229).

## Funciones incorporadas y de nivel superior

Como ya se ha tratado en la sección "Funciones y métodos" en la página 169, una función es un bloque de código de ActionScript que puede volver a utilizarse en cualquier parte de un archivo SWF. Si pasa valores a una función como parámetros, la función actúa sobre esos valores. Una función también puede devolver valores.

Puede utilizar funciones que están incorporadas en el lenguaje ActionScript. Las funciones pueden ser de nivel superior, como se describe en "Tipos de funciones y métodos" en la página 171, o estar dentro de una clase incorporada, como Math o MovieClip, que utiliza como método en la aplicación.

Las funciones incorporadas en ActionScript se utilizan para realizar determinadas tareas y acceder a información. Por ejemplo, puede obtener el número de milisegundos que se ha estado reproduciendo el archivo SWF utilizando `getTimer()`. O bien puede obtener el número de la versión de Flash Player en la que se aloja el archivo empleando `getVersion()`. Las funciones que pertenecen a un objeto se denominan *métodos*. Las funciones que no pertenecen a un objeto se denominan *funciones de nivel superior* y se encuentran en las subcategorías de la categoría Funciones globales del panel Acciones.

Algunas funciones incorporadas necesitan que se les pasen determinados valores. Si se pasan a la función más argumentos de los que necesita, se pasarán por alto los valores. Si no se le pasa un argumento necesario, los argumentos vacíos se asignarán al tipo de datos `undefined`, que puede causar errores en tiempo de ejecución.

| NOTA | Para llamar a una función, la definición de ésta debe encontrase en un fotograma al que haya llegado la cabeza lectora. |
|---|---|

Las funciones incorporadas son fáciles de usar. Para llamar a una función, utilice el nombre de función e indique los parámetros necesarios para la misma. (Para obtener información sobre los parámetros necesarios, consulte la entrada de la función en *Referencia del lenguaje ActionScript 2.0*). Por ejemplo, añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
trace("mi mensaje");
```

Al probar el archivo SWF, aparece `mi mensaje` en el panel Salida. Otros dos ejemplos de funciones de nivel superior son `setInterval()` y `getTimer()`. El siguiente ejemplo muestra cómo utilizar conjuntamente estas dos funciones. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
function myTimer():Void {
  trace(getTimer());
}
var intervalID:Number = setInterval(myTimer, 100);
```

Este código crea un temporizador sencillo mediante `getTimer()` y utiliza las funciones de nivel superior `setInterval()` y `trace()` para mostrar el número de milisegundos desde que comenzó a reproducirse el archivo SWF en Flash Player.

La llamada a una función de nivel superior es igual que la llamada a una *función definida por el usuario*. Para más información, consulte "Referencia y llamada a las funciones definidas por el usuario" en la página 181. Para obtener información sobre cada función, consulte su entrada en *Referencia del lenguaje ActionScript 2.0*.

## Escritura de funciones con nombre

Una función con nombre es un tipo de función que se crea con frecuencia en el código ActionScript para realizar todo tipo de acciones. Al crear un archivo SWF, se compilan primero las funciones con nombre, lo que significa que puede hacer referencia a la función en cualquier parte del código, siempre que la función se haya definido en el fotograma actual o anterior. Por ejemplo, si se define una función en el fotograma 2 de una línea de tiempo, no puede acceder a dicha función en el fotograma 1 de dicha línea.

El formato estándar para las funciones con nombre es el siguiente:

```
function functionName(parameters) {
  // bloque de función
}
```

Este código contiene las siguientes partes:

- `functionName` es el nombre exclusivo de la función. Todos los nombres de funciones de un documento deben ser exclusivos.
- `parameters` contiene uno o varios parámetros que se pasan a la función. Los parámetros se denominan a veces *argumentos*. Para más información sobre parámetros, consulte "Paso de parámetros a una función" en la página 187.
- `// bloque de función` contiene todo el código ActionScript que ejecuta la función. Esta parte contiene las sentencias que "hacen cosas". Aquí puede colocar el código que desea ejecutar. El comentario `// bloque de función` es un marcador de posición para el código del bloque de función.

### Para utilizar una función con nombre:

1. Cree un nuevo documento de Flash denominado **namedFunc.fla**.
2. Importe un archivo de sonido corto a la biblioteca; para ello, seleccione Archivo > Importar > Importar a biblioteca y elija el archivo de sonido.
3. Haga clic con el botón derecho del ratón en el archivo de sonido y seleccione Vinculación.
4. Introduzca **mySoundID** en el cuadro de texto Identificador.
5. Seleccione el fotograma 1 de la línea de tiempo y añada el siguiente código en el panel Acciones:

```
function myMessage() {
  trace("mySoundID completado");
}
var my_sound:Sound = new Sound();
my_sound.attachSound("mySoundID");
my_sound.onSoundComplete = myMessage;
my_sound.start();
```

En este código crea una función con nombre denominada `myMessage`, que utilizará posteriormente en el script para llamar a una función `trace()`.

6. Seleccione Control > Probar película para probar el archivo SWF.

La sentencia `function` le permite crear su propia función en ActionScript. Recuerde que los parámetros son opcionales; no obstante, aunque no incluya parámetros, deberá incluir los corchetes. El contenido incluido entre las llaves ({}) se conoce como el *bloque de función*.

Puede escribir funciones en la línea de tiempo principal o en archivos de código ActionScript externos, incluidos los archivos de clases.

También puede escribir funciones constructoras en archivos de clases empleando este formato (no obstante, el nombre de la función coincide con la clase). Para más información sobre funciones constructoras, consulte "Escritura de la función constructora" en la página 239. Consulte también el Capítulo 6, "Clases", en la página 195 para más información y ejemplos sobre la escritura de funciones en clases.

## Escritura de funciones anónimas y callback

Una función con nombre es aquella a la que se hace referencia en el script antes o después de definirla, mientras que una *función anónima* es una función sin nombre que hace referencia a ella misma; se hace referencia a la función anónima al crearla. Al escribir código ActionScript, creará numerosas funciones anónimas.

Las funciones anónimas se utilizan habitualmente al manipular controladores de eventos. Para escribir una función anónima, puede almacenarse un literal de función dentro de una variable. Por consiguiente, posteriormente podrá hacer referencia a la función en el código. En el siguiente ejemplo se muestra cómo escribir una función anónima.

### Para escribir una función anónima:

1. Cree un clip de película en el escenario y luego seleccione el clip.
2. Abra el inspector de propiedades y escriba **my_mc** en el cuadro de texto Nombre de instancia.
3. Seleccione el fotograma 1 de la línea de tiempo y, en el panel Acciones, introduzca el siguiente código:

```
var myWidth = function () {
  trace(my_mc._width);
};
//posteriormente en el código, puede añadir
myWidth();
```

4. Seleccione Control > Probar película.

   La anchura del clip de película se muestra en el panel Salida.

También puede crear una función dentro de un objeto, como XML o una instancia LoadVars. Puede asociar una función anónima a un determinado evento para crear una *función callback*. Una función llama a una función callback tras producirse un determinado evento, por ejemplo, después de que algo termine de cargarse (`onLoad()`) o finalice su animación (`onMotionFinished()`).

Por ejemplo, hay ocasiones en las que tendrá que escribir código ActionScript para que gestione datos que se cargan en un archivo SWF del servidor. Tras finalizar la carga de datos en un archivo SWF, puede acceder a los datos desde dicha ubicación. Es importante utilizar código ActionScript para comprobar si los datos se han cargado completamente. Puede utilizar funciones callback para enviar una señal que indique que los datos se han cargado en el documento.

En la siguiente función callback, en la que se carga un documento XML remoto, se asocia una función anónima al evento `onLoad()`. `XML.load()` y la función callback se emplean como se muestra en el siguiente ejemplo. Escriba el código siguiente en el fotograma 1 de la línea de tiempo:

```
var my_xml:XML = new XML();
my_xml.onLoad = function(success:Boolean):Void {
  trace(success);
};
my_xml.load("http://www.helpexamples.com/crossdomain.xml");
```

En el fragmento de código anterior se observa que el controlador de eventos `onLoad()` utiliza una función anónima para gestionar el evento `onLoad()`.

Para más información sobre funciones callback, consulte el Capítulo 9, “Gestión de eventos”, en la página 305.

También puede utilizar funciones anónimas con la función `setInterval()`, como se muestra en el siguiente código, que emplea `setInterval()` para llamar a la función anónima aproximadamente cada 1000 milisegundos (1 segundo):

```
setInterval(function() {trace("interval");}, 1000);
```

Puede utilizar funciones con nombre en lugar de funciones anónimas. Las funciones con nombre suelen ser más fáciles de leer y comprender (salvo en algunas circunstancias, como las funciones callback). También puede hacer referencia por adelantado a una función con nombre, lo que significa que hace referencia a ella antes de que tenga lugar la función en la línea de tiempo.

No se puede hacer referencia a una función anónima en cualquier parte del código (a menos que la haya asignado a una variable), como sí es el caso de las funciones con nombre. Por ejemplo, supongamos que tiene funciones anónimas en el fotograma 5 del archivo FLA, como las siguientes:

```
//con un clip de película denominado my_mc que se extiende por la línea de
  tiempo
stop();
var myWidth = function () {
  trace(my_mc._width);
};
```

Si coloca el siguiente código en el fotograma 1, no puede hacer referencia a la función:

```
myWidth();
```

De forma similar, no funciona el siguiente código ubicado en cualquier fotograma:

```
myWidth();
var myWidth:Function = function () {
  trace(my_mc._width);
};
```

Sin embargo, este código funciona correctamente:

```
var myWidth:Function = function () {
  trace(my_mc._width);
};
myWidth();
```

**NOTA**

También puede colocar `myWidth()` en cualquier fotograma que esté después del fotograma que incluye la función `myWidth`.

Al definir una función con nombre, funciona la llamada a ésta en un script de fotograma, aún cuando el código equivalente no funcione con una función anónima:

```
// el código siguiente sí funciona porque se está llamando a una función con
  nombre:
myWidth();
function myWidth() {
  trace("foo");
}

// el código siguiente no funciona porque se está llamando a una función
  anónima:
myWidth();
var myWidth:Function = function () {
  trace("foo");
};
```

Para más información, consulte "Escritura de funciones con nombre" en la página 175.

| NOTA | Para obtener información sobre la escritura de código con el asistente de script, consulte "Utilización del asistente de script para escribir ActionScript" en la página 362, "Creación de un evento startDrag/stopDrag con el Asistente de script" en la página 366 y el tutorial ActionScript: Utilización del modo de asistente de script (que comienza con "Apertura del documento inicial" en la página 229). |
|---|---|

## Literales de función

Un *literal de función* es una función sin nombre que se declara en una expresión en lugar de en una sentencia. Los literales de función son útiles si se desea usar una función temporalmente o para utilizar una función en el código en el que podría emplearse una expresión. La sintaxis de un literal de función es la siguiente:

```
function (param1, param2, etc) {
  // sentencias
};
```

Por ejemplo, el siguiente código utiliza un literal de función como expresión:

```
var yourName:String = "Ester";
setInterval(function() {trace(yourName);}, 200);
```

| NOTA | Al volver a definir un literal de función, la nueva definición de función sustituye a la anterior. |
|---|---|

Puede almacenar un literal de función en una variable para acceder a él en un punto posterior del código. Para ello, deberá utilizar una *función anónima*. Para más información, consulte "Escritura de funciones anónimas y callback" en la página 176.

## Funciones constructoras

El constructor de una clase es una función especial a la que se llama automáticamente cuando se crea una instancia de una clase mediante la palabra clave `new` (por ejemplo, `var my_xml:XML = new XML();`). La función constructora tiene el mismo nombre que la clase que la contiene. Por ejemplo, una clase Person personalizada que crease, incluiría la función constructora siguiente:

```
public function Person(speed:Number) {
  Person.numPeople++;
  this._speed = speed;
}
```

A continuación, podría crear una nueva instancia mediante:

```
var myPerson:Person = new Person();
```

| NOTA | Si no declara de forma explícita ninguna función constructora (es decir, si no crea una función cuyo nombre coincida con el nombre de la clase), el compilador crea automáticamente una función constructora vacía. |
|---|---|

Una clase sólo puede contener una función constructora; ActionScript 2.0 no admite funciones constructoras sobrecargadas. Además, una función constructora no puede tener un tipo de retorno. Para más información sobre la escritura de funciones constructoras en archivos de clase, consulte "Escritura de la función constructora" en la página 239.

## Definición de funciones globales y de línea de tiempo

En "Funciones y métodos" en la página 169, se han examinado los diferentes tipos de funciones disponibles en Flash. Al igual que las variables, las funciones están asociadas a la línea de tiempo del clip de película que las define y, para llamarlas, debe utilizarse una ruta de destino. Al igual que con las variables, puede utilizarse el identificador `_global` para declarar una función global que esté disponible para todas las líneas de tiempo y ámbitos sin utilizar una ruta de destino. Para definir una función global, debe anteponer el identificador `_global` al nombre de la función, como se muestra en el ejemplo siguiente:

```
_global.myFunction = function(myNum:Number):Number {
   return (myNum * 2) + 3;
};
trace(myFunction(5)) // 13
```

Para obtener información sobre `_global` y el ámbito, consulte "Variables y ámbito" en la página 354.

Para definir una función de línea de tiempo, utilice la sentencia `function` seguida del nombre de la función, los parámetros que va a pasar a la función y las sentencias de ActionScript que indican qué acción lleva a cabo la función.

En el ejemplo siguiente se muestra una función denominada `areaOfCircle` con el parámetro `radius`:

```
function areaOfCircle(radius:Number):Number {
  return (Math.PI * radius * radius);
}
trace (areaOfCircle(8));
```

También puede definir funciones de muchas otras formas. Para más información sobre cada tipo de función, consulte las secciones siguientes:

- "Funciones incorporadas y de nivel superior" en la página 173
- "Escritura de funciones con nombre" en la página 175
- "Escritura de funciones anónimas y callback" en la página 176
- "Literales de función" en la página 179
- "Funciones constructoras" en la página 179
- "Referencia y llamada a las funciones definidas por el usuario" en la página 181

Para más información sobre la asignación de nombres a las funciones, consulte "Asignación de nombre a las funciones" en la página 183. Para ver un ejemplo detallado de uso de funciones en un archivo de clase externo, consulte "Utilización de funciones en Flash" en la página 183 y el Capítulo 6, "Clases", en la página 195.

**NOTA**

Para obtener información sobre la escritura de código con el asistente de script, consulte "Utilización del asistente de script para escribir ActionScript" en la página 362, "Creación de un evento startDrag/stopDrag con el Asistente de script" en la página 366 y el tutorial ActionScript: Utilización del modo de asistente de script (que comienza con "Apertura del documento inicial" en la página 229).

## Referencia y llamada a las funciones definidas por el usuario

Las *funciones definidas por el usuario* son aquellas que crea usted mismo para usarlas en aplicaciones, a diferencia de las funciones en las clases incorporadas, que realizan funciones predefinidas. Deberá asignar nombre a las funciones usted mismo y añadir sentencias en el bloque de función. Las secciones anteriores tratan la escritura de funciones como, por ejemplo, las funciones con nombre, anónimas y callback. Para obtener información sobre la asignación de nombre a las funciones, consulte "Asignación de nombre a las funciones" en la página 183 y, para obtener información sobre la utilización de funciones, consulte "Utilización de funciones en Flash" en la página 183.

Puede utilizar una ruta de destino para llamar a una función en cualquier línea de tiempo y desde cualquier línea de tiempo, inclusive desde la línea de tiempo de un archivo SWF cargado. Para llamar a una función, escriba la ruta de destino del nombre de la función, si es necesario, y pase los parámetros necesarios entre paréntesis. Existen diversas formas de sintaxis para las funciones definidas por el usuario. En el código siguiente se utiliza una ruta para llamar a la función `initialize()`, que se ha definido en la línea de tiempo actual y que no requiere ningún parámetro:

```
this.initialize();
```

En el ejemplo siguiente se utiliza una ruta relativa para llamar a la función `list()` que se ha definido en el clip de película `functionsClip`:

```
this._parent.functionsClip.list(6);
```

Para obtener información sobre la escritura de funciones con nombre, consulte "Escritura de funciones con nombre" en la página 175. Para más información sobre parámetros, consulte "Paso de parámetros a una función" en la página 187.

También puede definir sus propias funciones con nombre. Por ejemplo, la siguiente función con nombre `helloWorld()` ha sido definida por el usuario:

```
function helloWorld() {
  trace("Hello world!");
};
```

En el siguiente ejemplo se muestra cómo utilizar una función definida por el usuario en un archivo FLA.

### Para crear y llamar a una función definida por el usuario sencilla:

1. Cree un nuevo documento de Flash y guárdelo como **udf.fla**.
2. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

   ```
   function traceHello(name:String):Void {
     trace("hello, " + name + "!");
   }
   traceHello("world"); // hello, world!
   ```

   El código anterior crea una función definida por el usuario denominada `traceHello()` que toma un argumento, `name`, y traza un mensaje de saludo. Para llamar a la función definida por el usuario, se puede llamar a `traceHello` desde la misma línea de tiempo que la definición de función y pasar un único valor de cadena.
3. Seleccione Control > Probar película para probar el documento de Flash.

Para más información sobre funciones con nombre, consulte "Escritura de funciones con nombre" en la página 175. Las clases contienen muchas funciones definidas por el usuario. Para obtener información sobre la escritura de funciones en archivos de clases, consulte "Utilización de funciones en Flash" en la página 183. Asimismo, consulte las siguientes secciones del Capítulo 6, "Clases": "Utilización de métodos y propiedades de un archivo de clase" en la página 215, "Métodos y propiedades (miembros) públicos, privados y estáticos" en la página 217 y "Miembros de clase" en la página 221.

## Asignación de nombre a las funciones

Los nombres de funciones deben empezar por una letra minúscula. Los nombres de las funciones deben describir el valor que devuelve la función, en el caso de que devuelvan un valor. Por ejemplo, si la función devuelve el título de una canción, podría denominar a la función `getCurrentSong()`.

Establezca un estándar para agrupar funciones similares (funciones relacionadas entre sí por su funcionalidad), porque ActionScript no permite la sobrecarga. En el contexto de la programación orientada a objetos (OOP), la *sobrecarga* hace referencia a la capacidad de hacer que las funciones se comporten de forma distinta según los tipos de datos que se pasen.

Al igual que con las variables, no puede utilizar caracteres especiales y el nombre del método no puede empezar por un número. Para más información, consulte "Convenciones de asignación de nombre" en la página 777. Para más información sobre la asignación de nombre a los métodos, consulte "Asignación de nombre a los métodos" en la página 194.

## Utilización de funciones en Flash

En esta sección se muestra cómo utilizar funciones en una aplicación. Algunos de los siguientes ejemplos de código utilizan código ActionScript que reside en el archivo FLA, mientras que otros ejemplos de código sitúan las funciones en un archivo de clase para su comparación. Para más información y ejemplos sobre la utilización de funciones en un archivo de clase, consulte el Capítulo 6, "Clases", en la página 195. Para obtener información detallada e instrucciones sobre la forma de escribir funciones para un archivo de clase, consulte "Ejemplo: Escritura de clases personalizadas" en la página 233.

Para reducir la cantidad de trabajo que debe realizar, así como el tamaño del archivo SWF, intente reutilizar bloques de código siempre que sea posible. Una forma de reutilizar código es llamar a una función varias veces en lugar de crear código distinto cada vez. Las funciones pueden ser fragmentos genéricos de código, de forma que es posible utilizar los mismos bloques de código para fines ligeramente distintos en un archivo SWF. La reutilización del código permite crear aplicaciones eficaces y reduce al mínimo la cantidad de código ActionScript que debe escribirse, por lo que el tiempo de desarrollo es menor.

Puede crear funciones en un archivo FLA o en un archivo de clase, o bien escribir código ActionScript que resida en un componente basado en código. Los siguientes ejemplos muestran cómo crear funciones en la línea de tiempo y en un archivo de clase.

| SUGERENCIA | Al empaquetar el código en archivos de clases o componentes basados en código, puede compartir, distribuir o reutilizar bloques de código fácilmente. Los usuarios pueden instalar el componente, arrastrarlo al escenario y utilizar el código almacenado en el archivo, como el flujo de trabajo de los componentes basados en código disponibles en Flash (Ventana › Bibliotecas comunes › Clases). |
|---|---|

En el siguiente ejemplo se muestra cómo crear y llamar a una función en un archivo FLA.

### Para crear y llamar a una función en un archivo FLA:

1. Cree un nuevo documento de Flash y guárdelo como **basicFunction.fla**.
2. Elija Ventana > Acciones para abrir el panel Acciones.
3. Escriba el siguiente código ActionScript en el panel Script:

```
function helloWorld() {
  // aquí van las sentencias
  trace("Hello world!");
};
```

   Este código ActionScript define la función (definida por el usuario, con nombre) denominada `helloWorld()`. Si prueba el archivo SWF en estos momentos, no ocurrirá nada. Por ejemplo, no verá la sentencia `trace` en el panel Salida. Para ver la sentencia `trace`, debe llamar a la función `helloWorld()`.

4. Escriba la siguiente línea de código ActionScript después de la función:

```
helloWorld();
```

   Este código llama a la función `helloWorld()`.

5. Seleccione Control > Probar película para probar el archivo FLA.

   El panel Salida muestra este texto: `Hello world!`

Para obtener información sobre cómo pasar valores (parámetros) a una función, consulte "Paso de parámetros a una función" en la página 187.

Existen diversas formas de escribir funciones en la línea de tiempo principal. Y lo que es más importante: puede utilizar funciones con nombre y funciones anónimas. Por ejemplo, puede utilizar la siguiente sintaxis al crear funciones:

```
function myCircle(radius:Number):Number {
  return (Math.PI * radius * radius);
}
trace(myCircle(5));
```

Las funciones anónimas suelen ser más difíciles de leer. Compare el código siguiente con el código anterior.

```
var myCircle:Function = function(radius:Number):Number {
  // aquí va el bloque de función
  return (Math.PI * radius * radius);
};
trace(myCircle(5));
```

También puede colocar funciones en archivos de clases al utilizar ActionScript 2.0, como se muestra en el siguiente ejemplo:

```
class Circle {
  public function area(radius:Number):Number {
    return (Math.PI * Math.pow(radius, 2));
  }
  public function perimeter(radius:Number):Number {
    return (2 * Math.PI * radius);
  }
  public function diameter(radius:Number):Number {
    return (radius * 2);
  }
}
```

Para más información sobre la escritura de funciones en un archivo de clase, consulte el Capítulo 6, “Clases”, en la página 195.

Como puede ver en el ejemplo de código anterior, no es preciso que coloque las funciones en la línea de tiempo. En el ejemplo siguiente también se incluyen funciones en un archivo de clase. Esta es una práctica recomendable al crear aplicaciones grandes utilizando ActionScript 2.0, ya que le permite reutilizar el código fácilmente en varias aplicaciones. Si desea reutilizar las funciones en otras aplicaciones, puede importar la clase existente en lugar de volver a escribir el código desde cero o duplicar las funciones en la nueva aplicación.

### Para crear funciones en un archivo de clase:

1. Cree un nuevo documento de ActionScript y guárdelo como **Utils.as**.
2. Escriba el siguiente código ActionScript en el panel Script:

```
class Utils {
  public static function randomRange(min:Number, max:Number):Number {
    if (min > max) {
      var temp:Number = min;
      min = max;
      max = temp;
    }
    return (Math.floor(Math.random() * (max - min + 1)) + min);
  }
```

```
    public static function arrayMin(num_array:Array):Number {
      if (num_array.length == 0) {
        return Number.NaN;
      }
      num_array.sort(Array.NUMERIC | Array.DESCENDING);
      var min:Number = Number(num_array.pop());
      return min;
    }
    public static function arrayMax(num_array:Array):Number {
      if (num_array.length == 0) {
        return undefined;
      }
      num_array.sort(Array.NUMERIC);
      var max:Number = Number(num_array.pop());
      return max;
    }
}
```

3. Seleccione Archivo > Guardar para guardar el archivo de ActionScript.
4. Cree un nuevo documento de Flash y guárdelo como **classFunctions.fla** en el mismo directorio que Utils.as.
5. Elija Ventana > Acciones para abrir el panel Acciones.
6. Escriba el siguiente código ActionScript en el panel Script:

```
var randomMonth:Number = Utils.randomRange(0, 11);
var min:Number = Utils.arrayMin([3, 3, 5, 34, 2, 1, 1, -3]);
var max:Number = Utils.arrayMax([3, 3, 5, 34, 2, 1, 1, -3]);
trace("month: " + randomMonth);
trace("min: " + min); // -3
trace("max: " + max); // 34
```

7. Seleccione Control > Probar película para probar los documentos. El panel Salida muestra este texto:

```
month: 7
min: -3
max: 34
```

**NOTA**

Para obtener información sobre la escritura de código con el asistente de script, consulte "Utilización del asistente de script para escribir ActionScript" en la página 362, "Creación de un evento startDrag/stopDrag con el Asistente de script" en la página 366 y el tutorial ActionScript: Utilización del modo de asistente de script (que comienza con "Apertura del documento inicial" en la página 229).

## Utilización de variables en funciones

Las variables locales son herramientas muy valiosas para organizar el código y hacer que sea más fácil de entender. Cuando una función utiliza variables locales, la función puede ocultar sus variables de todos los demás scripts del archivo SWF; las variables locales se invocan en el ámbito del cuerpo de la función y desaparecen cuando se sale de la función. Flash también trata los parámetros pasados a una función como variables locales.

| NOTA | En una función también pueden utilizarse variables regulares. No obstante, si modifica las variables regulares, es aconsejable utilizar comentarios de script para documentar los cambios. |
|---|---|

**Para utilizar variables en funciones:**

1. Cree un nuevo documento de Flash y guárdelo como **flashvariables.fla**.
2. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
var myName:String = "Ester";
var myAge:String = "65";
var myFavSoftware:String = "Flash";
function traceMe(yourFavSoftware:String, yourName:String,
  yourAge:String) {
  trace("Me llamo " + yourName + ", me gusta " + yourFavSoftware + " y
  tengo " + yourAge + ".");
}
traceMe(myFavSoftware, myName, myAge);
```

3. Seleccione Control > Probar película para probar el documento de Flash.

Para más información sobre cómo pasar parámetros, consulte "Paso de parámetros a una función" en la página 187. Para más información sobre variables y datos, consulte el Capítulo 10, "Datos y tipos de datos", en la página 327.

## Paso de parámetros a una función

Los parámetros, también conocidos como *argumentos*, son los elementos sobre los que una función ejecuta su código. (En este manual, los términos *parámetro* y *argumento* pueden utilizarse indistintamente). Puede pasar parámetros (valores) a una función. Luego puede utilizar dichos parámetros para procesar la función. Los valores se utilizan dentro del bloque de función (las sentencias contenidas en la función).

En determinados casos, los parámetros son obligatorios, en otros, opcionales. Puede que tenga incluso parámetros obligatorios y opcionales en una misma función. Si no pasa suficientes parámetros a una función, Flash establece los valores de parámetros que faltan como `undefined`, lo que puede causar resultados inesperados en el archivo SWF.

La siguiente función, denominada `myFunc()`, toma el parámetro `someText`:

```
function myFunc(someText:String):Void {
  trace(someText);
}
```

Tras pasar el parámetro, puede pasar un valor a la función al llamar a ésta. Este valor se traza en el panel Salida de la siguiente forma:

```
myFunc("Esto es lo que se traza");
```

Al llamar a la función, debería pasar siempre el número especificado de parámetros a menos que la función compruebe los valores no definidos y establezca los valores predeterminados correspondientes. La función sustituye los valores pasados de los parámetros en la definición de función, si falta algún parámetro, Flash establece su valor en `undefined`. Al escribir código ActionScript, debe pasar parámetros a las funciones con frecuencia.

También puede pasar varios parámetros a una función, que pueden ser tan simples como el siguiente:

```
var birthday:Date = new Date(1901, 2, 3);
trace(birthday);
```

Cada parámetro se separa de los demás mediante una coma delimitadora. Muchas funciones incorporadas en el lenguaje ActionScript tienen varios parámetros. Por ejemplo, el método `startDrag()` de la clase MovieClip toma cinco parámetros, `lockCenter`, `left`, `top`, `right` y `bottom`:

```
startDrag(lockCenter:Boolean, left:Number, top:Number, right:Number,
  bottom:Number):Void
```

### Para pasar un parámetro a una función:

1. Cree un nuevo documento de Flash y guárdelo como **parameters.fla**.
2. Añada el código siguiente al fotograma 1 de la línea de tiempo:

   ```
   function traceMe(yourMessage:String):Void {
     trace(yourMessage);
   }
   traceMe("¿Qué tal?");
   ```

   Las primeras líneas de código crean una función definida por el usuario denominada `traceMe()`, que toma un único parámetro, `yourMessage`. La última línea de código llama a la función `traceMe()` y pasa el valor de la cadena "`¿Qué tal?`".
3. Seleccione Control > Probar película para probar el documento de Flash.

En el siguiente ejemplo se muestra cómo pasar varios parámetros a una función.

### Para pasar varios parámetros a una función:

1. Cree un nuevo documento de Flash y guárdelo como **functionTest.fla**.
2. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
function getArea(width:Number, height:Number):Number {
     return width * height;
}
```

   La función `getArea()` toma dos parámetros, `width` y `height`.

3. Escriba el siguiente código después de la función:

```
var area:Number = getArea(10, 12);
trace(area); // 120
```

   La llamada a la función `getArea()` asigna los valores 10 y 12 a width y height respectivamente y se guarda el valor de retorno en la instancia `area`. Posteriormente se trazan los valores que se han guardado en la instancia `area`.

4. Seleccione Control > Probar película para probar el archivo SWF.

   Se verá `120` en el panel Salida.

   Los parámetros de la función `getArea()` son similares a los valores de una variable local; existen mientras se llama a la función y dejan de existir cuando se cierra la función.

En el siguiente ejemplo, el código ActionScript devuelve el valor `NaN` (no un número) si no se pasan suficientes parámetros a la función `addNumbers()`.

### Para pasar un número variable de parámetros a una función:

1. Cree un nuevo documento de Flash y guárdelo como **functionTest2.fla**.
2. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
function addNumbers(a:Number, b:Number, c:Number):Number {
  return (a + b + c);
}
trace(addNumbers(1, 4, 6)); // 11
trace(addNumbers(1, 4)); // NaN (Not a Number), c equals undefined
trace(addNumbers(1, 4, 6, 8)); // 11
```

   Si no pasa suficientes parámetros a la función `addNumbers`, los argumentos que faltan se asignan al valor predeterminado de `undefined`. Si, por el contrario, pasa demasiados parámetros, se ignoran los que sobran.

3. Seleccione Control > Probar película para probar el documento de Flash.

   Flash muestra los siguientes valores: 11, NaN, 11.

# Devolución de valores de funciones

La sentencia `return` se utiliza para devolver valores desde las funciones. La sentencia `return` especifica el valor que devuelve la función. La sentencia `return` devuelve el resultado de una evaluación como valor de la función en la que se ejecuta una expresión. La sentencia `return` devuelve de inmediato su resultado al código que realiza la llamada.

Para más información, consulte `%{return statement}%` en la *Referencia del lenguaje ActionScript 2.0*.

El uso de la sentencia `return` se rige por las siguientes reglas:

- Si especifica un tipo de devolución que no sea Void para una función, debe incluir una sentencia `return` seguida del valor devuelto en la función.
- Si especifica el tipo de devolución Void, no es necesario que incluya una sentencia `return`, pero si la incluye, ésta no deberá ir seguida de ningún valor.
- Con independencia del tipo de devolución, puede utilizar una sentencia `return` para salir de una función.
- Si no especifica ningún tipo de devolución, la inclusión de una sentencia `return` es opcional.

Por ejemplo, la siguiente función devuelve el cuadrado del parámetro `myNum` y especifica que el valor devuelto debe ser un número:

```
function sqr(myNum:Number):Number {
  return myNum * myNum;
}
```

Algunas funciones realizan una serie de tareas sin devolver un valor. En el siguiente ejemplo se devuelve el valor procesado. Dicho valor se captura en una variable, que posteriormente se utiliza en la aplicación.

### Para devolver un valor y capturarlo en una variable:

1. Cree un nuevo documento de Flash y guárdelo como **return.fla**.
2. Añada el código siguiente al fotograma 1 de la línea de tiempo:

   ```
   function getArea(width:Number, height:Number):Number {
        return width * height;
   }
   ```

   La función `getArea()` toma dos parámetros, `width` y `height`.

3. Escriba el siguiente código después de la función:

```
var area:Number = getArea(10, 12);
trace(area); // 120
```

La llamada a la función `getArea()` asigna los valores 10 y 12 a width y height respectivamente y se guarda el valor de retorno en la instancia `area`. Posteriormente se trazan los valores que se han guardado en la instancia `area`.

4. Seleccione Control > Probar película para probar el archivo SWF.

Se verá `120` en el panel Salida.

Los parámetros de la función `getArea()` son similares a los valores de una variable local; existen mientras se llama a la función y dejan de existir cuando se cierra la función.

## Funciones anidadas

Puede llamar a una función desde dentro de otra función. Esto le permite anidar funciones de modo que éstas realicen determinadas tareas en Flash.

Por ejemplo, puede anidar funciones en la línea de tiempo principal para realizar tareas específicas en una cadena. Escriba el código siguiente en el fotograma 1 de la línea de tiempo:

```
var myStr:String = "El pollo es amarillo.";
trace("Original string: " + myStr);
function formatText():Void {
  changeString("Poner el pollo en el microondas.");
  trace("Changed string: " + myStr);
}
function changeString(newtext:String):Void {
  myStr = newtext;
}
// Llama a la función.
formatText();
```

Seleccione Control > Probar película para probar la función anidada. Las funciones `formatText()` y `changeString()` se aplican a la cadena cuando se llama a la función `formatText()`.

# Aspectos básicos de los métodos

Los métodos son funciones asociadas a una clase. La clase podría ser una clase personalizada, o bien una clase incorporada que forma parte del lenguaje ActionScript. Para obtener información sobre la comparación de métodos en funciones, consulte "Funciones y métodos" en la página 169 y "Tipos de funciones y métodos" en la página 171.

Por ejemplo, `sortOn()` es un método incorporado asociado con la clase Array (`sortOn` es una función de la clase Array predefinida incorporada en Flash).

### Para utilizar el método sortOn() en un archivo FLA:

1. Cree un nuevo documento de Flash y guárdelo como **methods.fla**.
2. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
var userArr:Array = new Array();
userArr.push({firstname:"George", age:39});
userArr.push({firstname:"Dan", age:43});
userArr.push({firstname:"Socks", age:2});
userArr.sortOn("firstname");
var userArrayLenth:Number = userArr.length;
var i:Number;
for (i = 0; i < userArrayLenth; i++) {
  trace(userArr[i].firstname);
}
```

   Se utiliza el método `sortOn()` de la clase Array para crear un nuevo objeto Array denominado `userArr`. La matriz se rellena con tres objetos que contienen un nombre y una edad, tras lo cual la matriz se ordena en función del valor de la propiedad `firstname` de cada objeto. Finalmente, se reproduce sobre cada elemento de la matriz y se muestran los nombres en el panel Salida ordenados alfabéticamente por la primera letra.

3. Seleccione Control > Probar película para probar el archivo SWF.

   Este código muestra la siguiente información en el panel Salida:

```
Dan
George
Socks
```

Como se mostró en la sección "Escritura de funciones con nombre" en la página 175, cuando se escribe el siguiente código en el fotograma 1 de la línea de tiempo, el código ActionScript define una función denominada `eatCabbage()`.

```
function eatCabbage() {
    trace("tastes bad");
}
eatCabbage();
```

Sin embargo, si escribe la función `eatCabbage()` dentro de un nombre de clase y, por ejemplo, llama a `eatCabbage()` en el archivo FLA, `eatCabbage()` se considera como método.

Los siguientes ejemplos muestran cómo crear métodos dentro de una clase.

### Para comparar métodos y funciones:

1. Cree un nuevo archivo ActionScript, seleccione Archivo > Guardar como y guárdelo como **EatingHabits.as**.
2. Escriba el siguiente código ActionScript en la ventana Script:

```
class EatingHabits {
  public function eatCabbage():Void {
    trace("tastes bad");
  }
}
```

3. Guarde los cambios en EatingHabits.as.
4. Cree un nuevo documento de Flash, seleccione Archivo > Guardar como, asígnele el nombre **methodTest.fla** y guarde este archivo en el mismo directorio que EatingHabits.as.
5. Escriba el siguiente código ActionScript en el fotograma 1 de la línea de tiempo:

```
var myHabits:EatingHabits = new EatingHabits();
myHabits.eatCabbage();
```

   Al utilizar este código ActionScript, se llama al método `eatCabbage()` de la clase EatingHabits.

> **NOTA** Al utilizar métodos de cualquier clase incorporada (además de la clase personalizada escrita anteriormente en este procedimiento), se utiliza un *método* en la línea de tiempo.

6. Añada el siguiente código tras la línea de código ActionScript anterior:

```
function eatCarrots():Void {
    trace("tastes good");
}
eatCarrots();
```

   En este código, se escribe y se llama a la función `eatCarrots()`.
7. Seleccione Control > Probar película para probar el archivo SWF.

## Asignación de nombre a los métodos

Debe utilizar verbos para denominar a los métodos, mezclando mayúsculas y minúsculas en las palabras concatenadas, y asegurarse de que la primera letra está en minúscula. Por ejemplo, podría denominar los siguientes métodos:

```
sing();
boogie();
singLoud();
danceFast();
```

En la mayoría de los nombres de métodos se utilizan verbos porque ejecutan una operación en un objeto. Al igual que con las variables, no puede utilizar caracteres especiales y el nombre del método no puede empezar por un número. Para más información, consulte "Convenciones de asignación de nombre" en la página 777.

CAPÍTULO 6

# Clases

6

En este capítulo se describe cómo utilizar y escribir clases mediante ActionScript 2.0. Las clases constituyen la base de ActionScript 2.0, ya que han cobrado una importancia aun mayor de la que tenían en versiones anteriores de Macromedia Flash. A lo largo de este capítulo tendrá la oportunidad de describir lo importante que son las clases en Flash.

Este capítulo comienza con la definición de diversos términos fundamentales y la relación de éstos con las clases y la programación orientada a objetos (OOP). Seguidamente, estudiaremos un archivo de clase de muestra, lo que le permitirá conocer cómo funciona cada sección del archivo de clase y cómo está organizada la clase. En el resto del capítulo se muestra cómo crear clases personalizadas y cómo utilizarlas en los documentos de Flash. Aprenderá en qué consiste la ruta de clases de Flash y cómo debe documentarse una clase para que otras personas que lean o utilicen el código entiendan fácilmente el código y el objetivo global de la clase.

En esta sección se incluyen ejemplos de código que puede utilizar para familiarizarse con la creación de clases en ActionScript 2.0. Al término de este capítulo, debería poder escribir un archivo de clase típico, ser capaz de reconocer clases de Flash y leer con soltura los archivos de clases escritos por otras personas.

Si no está familiarizado con la creación de scripts en ActionScript 2.0, consulte el Capítulo 4, “Principios básicos de la sintaxis y el lenguaje”, en la página 75 y el Capítulo 19, “Recomendaciones y convenciones de codificación para ActionScript 2.0”, en la página 775.

Para más información sobre el trabajo con clases personalizadas e incorporadas, consulte los temas siguientes:

# Programación orientada a objetos y Flash

ActionScript 2.0 es un lenguaje orientado a objetos. Al igual que ActionScript, los lenguajes OOP se basan en el concepto de *clases* e *instancias*. Una clase define todas las propiedades que distinguen a una serie de objetos. Por ejemplo, una clase User representa a una serie de usuarios que están utilizando una aplicación. A continuación, tiene lugar la *creación de la instancia* de la clase que, para la clase User, es uno de los usuarios concretos (uno de sus miembros). La creación de la instancia produce una *instancia* de la clase User, que posee todas las propiedades de esta clase.

Las clases también se consideran como *tipos de datos* o *plantillas* que pueden crearse para definir un nuevo tipo de objeto. Por ejemplo, si necesita el tipo de datos Lettuce (lechuga) en la aplicación, puede escribir la clase Lettuce. Esto define el objeto Lettuce, tras lo cual podrá asignar los métodos Lettuce (`wash()`) y propiedades (`leafy` o `bugs`). Para definir una clase, se utiliza la palabra clave `class` en un archivo de script externo. Puede crear un archivo de script externo en la herramienta de edición de Flash seleccionando Archivo > Nuevo y Archivo ActionScript.

Flash Player 8, que está disponible tanto en Flash Basic 8 como en Flash Professional 8, añade diversas funciones nuevas al lenguaje ActionScript, como son los efectos de filtro, la carga y descarga de archivos y la API externa. Como siempre, ActionScript 2.0 ofrece diversos conceptos de OOP y palabras clave de gran potencia (como, por ejemplo, `class`, `interface` y `package`) que habitualmente se incluyen en otros lenguajes de programación, como es el caso de Java. El lenguaje de programación le permite crear estructuras de programa reutilizables, escalables, sólidas y sostenibles. También puede disminuir el tiempo de desarrollo al proporcionar a los usuarios ayuda para la codificación e información de depuración detallada. Puede utilizar ActionScript 2.0 para crear objetos y establecer herencias, así como para crear clases personalizadas y ampliar las clases de nivel superior e incorporadas de Flash. En este capítulo aprenderá a crear clases y a utilizar clases personalizadas.

Flash Basic 8 y Flash Professional 8 incluyen aproximadamente 65 clases de nivel superior e incorporadas que ofrecen desde tipos de datos básicos o "simples" (Array, Boolean, Date, etc.) hasta errores y eventos personalizados, además de formas de cargar contenido externo (XML, imágenes, datos binarios sin formato, etc.). También puede escribir sus propias clases personalizadas e integrarlas en los documentos de Flash, o incluso ampliar las clases de nivel superior y añadir su propia funcionalidad o modificar la funcionalidad existente. Por ejemplo, en la sección "Miembros de clase" en la página 221 de este capítulo se muestra cómo crear una clase personalizada Person que contiene propiedades personalizadas para el nombre y la edad de la persona. Posteriormente, podrá considerar esta clase personalizada como un nuevo tipo de datos en los documentos y crear una nueva instancia de la clase mediante el operador `new`.

Para más información sobre el trabajo con OOP, consulte los siguientes temas:



## Ventajas de utilizar clases

En OOP, una clase define una categoría de objetos. Una clase describe las propiedades (datos) y los métodos (comportamientos) de un objeto, del mismo modo que un plano arquitectónico describe las características de un edificio. Una clase personalizada se escribe en un archivo ActionScript (AS) externo que puede importar a la aplicación al compilar el archivo FLA.

Las clases pueden resultar muy útiles al crear aplicaciones Flash de gran tamaño, ya que permiten organizar gran parte de la complejidad de la aplicación en archivos de clases externos. Al mover gran parte de la lógica a una clase personalizada, no solamente resulta más fácil reutilizar el código, sino que también puede "ocultar" algunos de los métodos y propiedades de otras partes del código ActionScript. Esto contribuye a impedir que alguien pueda acceder a información confidencial o cambiar datos que no deben cambiarse.

Al utilizar una clase, también puede ampliar las clases existentes y añadir nueva funcionalidad o modificar la funcionalidad existente. Por ejemplo, si crea tres clases muy similares, puede escribir una clase básica y luego escribir otras dos clases que amplíen la clase básica. Estas dos clases pueden añadir métodos y propiedades adicionales, de modo que no necesita crear tres archivos de clases que repitan el mismo código y la misma lógica.

Otra ventaja del uso de clases es la capacidad de reutilización del código. Por ejemplo, si crea una clase personalizada que crea una barra de progreso personalizada empleando la interfaz de programación de aplicaciones (API) de dibujo, puede guardar la clase de barra de progreso en la ruta de clases y reutilizar el mismo código en todos los documentos de Flash mediante la importación de la clase personalizada. Para más información sobre la configuración de la ruta de clases, consulte "Importación de archivos de clases" en la página 209 y "Configuración y modificación de la ruta de clases" en la página 211.

# Paquetes

Al crear clases, los archivos de clases de ActionScript se organizan en *paquetes*. Un paquete es un directorio que contiene uno o más archivos de clases y que reside en un directorio de rutas de clases designado (consulte "Importación de archivos de clases" en la página 209 y "Configuración y modificación de la ruta de clases" en la página 211). Un paquete puede, a su vez, contener otros paquetes, denominados *subpaquetes*, cada uno de ellos con sus propios archivos de clase.

Al igual que las variables, los nombres de paquete deben ser identificadores; es decir, que el primer carácter puede ser una letra, un carácter de subrayado (_) o un símbolo de dólar ($), y los caracteres siguientes pueden ser una letra, un número, un carácter de subrayado o un símbolo de dólar. Hay algunas formas de asignar nombre a los paquetes que son preferibles a otras como, por ejemplo, la recomendación de que evite utilizar caracteres de subrayado o símbolos del dólar. Para más información sobre la asignación de nombre a los paquetes, consulte "Asignación de nombre a paquetes" en la página 786.

Los paquetes suelen utilizarse para organizar clases relacionadas. Por ejemplo, puede tener tres clases relacionadas, Square, Circle y Triangle, que se definen en Square.as, Circle.as y Triangle.as. Suponga que ha guardado los archivos ActionScript en un directorio especificado en la ruta de clases, como se muestra en el siguiente ejemplo:

```
// En Square.as:
class Square {}

// En Circle.as:
class Circle {}

// En Triangle.as:
class Triangle {}
```

Dado que estos tres archivos de clase están relacionados, puede decidir ponerlos en un paquete (directorio) denominado Shapes. En este caso, el nombre de clase calificado contendría la ruta del paquete, así como el nombre de clase simple. Para las rutas de paquetes se utiliza la sintaxis con punto, en la que cada punto indica un subdirectorio.

Por ejemplo, si ha colocado todos los archivos ActionScript que definen una forma en el directorio Shapes, necesitará modificar el nombre de cada archivo de clase para que refleje la nueva ubicación, del modo siguiente:

```
// En Shapes/Square.as:
class Shapes.Square {}

// En Shapes/Circle.as:
class Shapes.Circle {}

// En Shapes/Triangle.as:
class Shapes.Triangle {}
```

Para hacer referencia a una clase que reside en un directorio del paquete, puede especificar su nombre de clase completo o importar el paquete mediante la sentencia `import`. Para más información, consulte "Utilización de paquetes" en la página 199.

## Comparación de clases y paquetes

En OOP, una clase define una categoría de objetos. Las clases son básicamente tipos de datos que pueden crearse si se desea definir un nuevo tipo de objeto en la aplicación. Una clase describe las *propiedades* (datos) y el *comportamiento* (métodos) de un objeto, del mismo modo que un plano arquitectónico describe las características de un edificio. Las propiedades (variables definidas dentro de una clase) y los métodos de una clase se conocen colectivamente como *miembros* de la clase. Para utilizar las propiedades y los métodos definidos por una clase, primero debe crear una instancia de dicha clase (salvo en el caso de clases que tengan miembros estáticos (consulte "Miembros de clase (estáticos)" en la página 271, como la clase de nivel superior Math, y "Métodos y propiedades estáticos" en la página 219). La relación entre una instancia y su clase es similar a la relación entre una casa y sus planos.

En Flash, los paquetes son directorios que contienen uno o más archivos de clase y que residen en la ruta de archivos determinada. Puede colocar los archivos de clases personalizadas relacionadas en un mismo directorio. Por ejemplo, puede tener tres clases relacionadas denominadas SteelWidget, PlasticWidget y WoodWidget que se definen en SteelWidget.as, PlasticWidget.as y WoodWidget.as. Estas clases se organizarían en el paquete Widget. Para más información sobre paquetes, consulte "Utilización de paquetes" en la página 199 y "Creación y empaquetado de archivos de clases" en la página 236.

## Utilización de paquetes

Los paquetes son directorios que contienen uno o más archivos de clase y que residen en un directorio classspath determinado. Por ejemplo, el paquete flash.filters es un directorio del disco duro que contiene varios archivos de clase para cada tipo de filtro (como BevelFilter, BlurFilter, DropShadowFilter, entre otros) en Flash 8.

| NOTA | Para utilizar la sentencia `import`, debe especificar ActionScript 2.0 y Flash Player 6 o posterior en la ficha Flash del cuadro de diálogo Configuración de publicación del archivo FLA. |
|---|---|

La sentencia `import` permite acceder a las clases sin especificar sus nombres completos. Por ejemplo, si desea utilizar la clase BlurFilter en un script, debe hacer referencia a ésta con el nombre completo (flash.filters.BlurFilter) o importarla; si lo hace, puede hacer referencia a ésta con el nombre de clase (BlurFilter). El siguiente código ActionScript muestra las diferencias entre utilizar la sentencia `import` y los nombres de clase completos.

Si no realiza la importación de la clase BlurFilter, el código necesita utilizar el nombre de la clase completa (nombre del paquete seguido del nombre de la clase) para poder utilizar el filtro:

```
// sin importación
var myBlur:flash.filters.BlurFilter = new flash.filters.BlurFilter(10, 10,
  3);
```

El mismo código, escrito con una sentencia `import`, le permite acceder a BlurFilter con sólo el nombre de la clase en lugar de tener siempre que utilizar el nombre completo. De este modo, se ahorra tener que escribir y se reducen las posibilidades de cometer errores:

```
// con la importación
import flash.filters.BlurFilter;
var myBlur:BlurFilter = new BlurFilter(10, 10, 3);
```

Si tuviese que importar varias clases dentro de un paquete (como BlurFilter, DropShadowFilter y GlowFilter), podría utilizar uno de los dos métodos para importar cada clase. El primer método para importar varias clases consiste en importar cada clase utilizando una sentencia `import` independiente, como se ve en el siguiente fragmento:

```
import flash.filters.BlurFilter;
import flash.filters.DropShadowFilter;
import flash.filters.GlowFilter;
```

La utilización de sentencias `import` independientes para cada clase dentro de un paquete puede resultar una tarea lenta y con muchas posibilidades de escribir errores. El segundo método para importar clases dentro de un paquete consiste en utilizar una importación con comodín que importe todas las clases dentro de un determinado nivel de un paquete. El siguiente código ActionScript muestra un ejemplo de una importación con comodín:

```
import flash.filters.*; // importa cada clase dentro del paquete
  flash.filters
```

La sentencia `import` sólo se aplica al script actual (fotograma u objeto) en el que se llama. Por ejemplo, supongamos que importa todas las clases del paquete macr.util en el fotograma 1 de un documento de Flash. En dicho fotograma, puede hacer referencia a las clases del paquete por sus nombres de clase en lugar de los nombres completos. Sin embargo, si deseara utilizar el nombre de clase en otro script de fotograma, necesitaría hacer referencia a las clases de dicho paquete por sus nombres completos o añadir una sentencia `import` al otro fotograma que importa las clases de dicho paquete.

Al utilizar sentencias `import`, también es importante advertir que las clases se importan solamente para el nivel especificado. Por ejemplo, si importa todas las clases del paquete mx.transitions, sólo se importan aquellas clases dentro del directorio /transitions/, no todas las clases dentro de los subdirectorios (como las clases del paquete mx.transitions.easing).

| SUGERENCIA | Si importa una clase pero no la utiliza en el script, la clase no se exporta como parte del archivo SWF. Eso significa que puede importar grandes paquetes sin preocuparse del tamaño del archivo SWF; el código de bytes asociado con una clase se incluye en un archivo SWF únicamente si dicha clase se utiliza realmente. |
|---|---|

## Valores y tipos de datos

Los datos, valores y tipos son importantes al comenzar a escribir y utilizar clases. En el Capítulo 10, “Datos y tipos de datos”, en la página 327, ya ha conocido los datos y tipos. Al utilizar clases, recuerde que los tipos de datos describen el tipo de información que puede contener una variable o elemento de ActionScript, como son Boolean, Number y String. Para más información, consulte “Tipos de datos” en la página 328.

Las expresiones tienen valores, mientras que los valores y las propiedades tienen *tipos*. Los valores que puede establecer y obtener de una propiedad de la clase deben ser compatibles con dicha propiedad. La compatibilidad de tipos supone que el tipo de un valor es compatible con el tipo en el que se utiliza, como en el siguiente ejemplo:

```
var myNum:Number = 10;
```

Para más información sobre “strict data typing”, consulte “Asignación de tipos de datos y “strict data typing”” en la página 337.

## Conceptos básicos sobre la programación orientada a objetos

En las siguientes secciones, abordaremos algunos de los términos empleados en este capítulo antes de comenzar a escribir código ActionScript. Esta breve introducción a los principios en los que se basa el desarrollo de programas orientados a objetos le ayudará a seguir los ejemplos y las secciones de este capítulo y del resto del manual. Estos principios se describen con mayor profundidad en el resto de este capítulo, junto con detalles sobre la forma de aplicarlos en Flash 8.

En las siguientes secciones se utiliza un gato (cat) como analogía para demostrar su semejanza con los conceptos de OOP.

## Objetos

Piense en un objeto del mundo real, como por ejemplo un gato. Podría decirse que un gato tiene *propiedades* (o estados) como nombre, edad y color; también tiene comportamientos como dormir, comer y ronronear. En el mundo de la OOP, los objetos también tienen propiedades y comportamientos. Al utilizar las técnicas orientadas a objetos, se puede tomar como modelo un objeto del mundo real (como un gato) o un objeto más abstracto (como un proceso químico).

| NOTA | La palabra *comportamientos* se utiliza aquí de manera genérica y no se refiere al panel Comportamientos del entorno de edición de Flash. |
|---|---|

Para más información sobre objetos, consulte "Tipo de datos Object (objeto)" en la página 335.

## Instancias y miembros de clase

Siguiendo con la analogía de un gato, piense que hay gatos de distintos colores, edades y nombres, con maneras diferentes de comer y ronronear. Pero a pesar de sus diferencias individuales, todos los gatos pertenecen a la misma categoría o, en términos de OOP, a la misma clase: la clase formada por los gatos (cats). En la terminología de OOP, se dice que cada gato individual es una *instancia* de la clase Cat (gato).

Del mismo modo que en OOP, una clase define un modelo para un tipo de objeto. Las características y los comportamientos que pertenecen a una clase se conocen conjuntamente como *miembros* de dicha clase. Las características (en el ejemplo del gato, el nombre, la edad y el color) se denominan *propiedades* de la clase, y se representan como variables; los comportamientos (jugar, dormir) se denominan *métodos* de la clase, y se representan como funciones.

Para más información sobre instancias y miembros de clases, consulte "Miembros de clase" en la página 221 y "Utilización de miembros de clase" en la página 224.

## Herencia

Una de las principales ventajas de la OOP es que se pueden crear *subclases* de una clase (es decir, *ampliar* la clase); la subclase heredará todas las propiedades y los métodos de la clase. La subclase normalmente define métodos y propiedades adicionales o anula los métodos o propiedades definidos en la superclase. Las subclases también pueden suplantar a (proporcionar sus propias definiciones para) los métodos heredados de una superclase.

Una de las principales ventajas que aporta el uso de una estructura superclase/subclase es que resulta más fácil reutilizar código similar entre diversas clases. Por ejemplo, puede crear una superclase denominada Animal que contenga características y comportamientos comunes a todos los animales. Seguidamente, tendría que crear diversas subclases que heredaran de la superclase Animal y que añadieran características y comportamientos específicos de ese tipo de animal.

Podría crear una clase Cat que heredara de otra clase. Por ejemplo, podría crear una clase Mammal (mamífero) que defina determinadas propiedades y comportamientos comunes a todos los mamíferos. Luego podría crear una subclase Cat (gato) que se ampliara a la clase Mammal. A su vez, otra subclase, pongamos por caso, la clase Siamese (siamés), podría ampliar (*subclase*) la clase Cat y, así, sucesivamente.

La escritura de subclases le permite reutilizar el código. En lugar de volver a crear todo el código común a las dos clases, simplemente puede ampliar una clase existente.

| SUGERENCIA | En una aplicación compleja, determinar la estructura jerárquica de las clases constituye una parte importante del proceso de diseño. Asegúrese de que determina esta jerarquía antes de comenzar a programar. |
|---|---|

Para más información sobre herencia y subclases, consulte el Capítulo 7, “Herencia”, en la página 275.

## Interfaces

En OOP, las *interfaces* pueden describirse como plantillas de definiciones de clases y clases que implementan interfaces necesarias para implementar dicha plantilla de métodos. Siguiendo la analogía del gato, una interfaz es similar a un modelo de un gato: indica las partes que necesita, pero no necesariamente cómo se montan dichas partes o cómo funcionan.

Las interfaces se pueden utilizar para añadir estructura y facilitar el mantenimiento de las aplicaciones. Dado que ActionScript 2.0 sólo admite la ampliación desde una única superclase, puede utilizar las interfaces como una forma de herencia múltiple limitada.

Una interfaz también puede considerarse como un "contrato de programación" que puede utilizarse para aplicar relaciones entre clases que de otro modo no estarían relacionadas. Por ejemplo, suponga que está trabajando con un equipo de programadores y que cada uno de ellos está trabajando en una parte (clase) distinta de la misma aplicación. Al diseñar la aplicación, se acordaron una serie de métodos que utilizan las distintas clases para comunicarse. Por lo tanto, puede crear una interfaz que declare estos métodos, sus parámetros y sus tipos de devolución. Todas las clases que implementen esta interfaz deben proporcionar definiciones de dichos métodos; de lo contrario, se produce un error en el compilador.

Para más información sobre herencia, consulte el Capítulo 7, "Herencia", en la página 275. Para más información sobre interfaces, consulte el Capítulo 8, "Interfaces", en la página 289.

## Encapsulado

En el diseño elegante orientado a objetos, los objetos son vistos como "cajas negras" que contienen o *encapsulan* funcionalidad. Un programador debe poder interactuar con un objeto conociendo sólo sus propiedades, métodos y eventos (su interfaz de programación), sin conocer los detalles de su implementación. Este enfoque permite a los programadores pensar con niveles superiores de abstracción y ofrece un marco organizativo para la creación de sistemas complejos.

El encapsulado es la razón por la cual ActionScript 2.0 incluye, por ejemplo, control de acceso para miembros, de manera que los detalles de la implementación puedan ser privados e invisibles en el código externo a un objeto. El código situado fuera del objeto se ve forzado a interactuar con la interfaz de programación de objetos en lugar de con los detalles de la implementación (que pueden estar ocultos en métodos y propiedades privadas). Este enfoque aporta importantes ventajas; por ejemplo, permite al creador del objeto cambiar la implementación del objeto sin necesidad de realizar cambios en el código externo al objeto, es decir, siempre y cuando la interfaz de programación no cambie.

Para más información sobre encapsulado, consulte "Utilización de encapsulado" en la página 232.

## Polimorfismo

La OOP le permite expresar diferencias entre clases individuales mediante la técnica conocida como *polimorfismo*, que permite que las clases sustituyan métodos de sus superclases y definan implementaciones especializadas de dichos métodos. En Flash, las subclases pueden definir implementaciones especializadas de métodos heredados de su superclase pero no pueden acceder a la implementación de la superclase como en los demás lenguajes de programación.

Por ejemplo, puede comenzar con una clase llamada Mammal que tenga los métodos `play()` y `sleep()`. Puede crear las subclases Cat, Monkey y Dog para ampliar la clase Mammal. Las subclases sustituyen el método `play()` de la clase Mammal para reflejar los hábitos de estos tipos de animales en particular. Monkey implementa el método `play()` para columpiarse de árbol en árbol; Cat implementa el método `play()` para abalanzarse sobre una madeja; Dog implementa el método `play()` para buscar una pelota. Dado que la funcionalidad `sleep()` es similar entre los animales, utilizaría la implementación de la superclase.

Para más información sobre polimorfismo, consulte el Capítulo 7, "Herencia", en la página 275 y "Utilización de polimorfismo en una aplicación" en la página 283.

# Escritura de archivos de clases personalizadas

En el siguiente ejemplo se examinan las partes de un archivo de clase. Aprenderá a escribir una clase y a modificarla para ampliar las formas en que puede utilizarla con Flash. Conocerá las diversas partes de una clase y la importación de clases, así como información relacionada con la utilización de archivos de clases personalizadas en Flash.

Comenzaremos con una clase muy simple. En el siguiente ejemplo se muestra la organización de una clase simple denominada UserClass.

Para definir una clase, se utiliza la palabra clave `class` en un archivo de script externo (es decir, no en un script que se esté escribiendo en el panel Acciones). La estructura de clases es también relevante para los archivos de interfaz. Esta estructura se ilustra a continuación, tras la cual se crea una clase.

- El archivo de clase empieza con comentarios de documentación que incluyen una descripción general del código, además de información del autor y de la versión.
- Añada las sentencias `import` (si resulta aplicable).
- Escriba una sentencia de paquete, declaración de clase o declaración de interfaz de la siguiente forma:

  ```
  class UserClass {...}
  ```

- Incluya los comentarios de implementación de clase o interfaz que sean necesarios. En estos comentarios, añada información que sea relevante para toda la clase o interfaz.
- Añada todas las variables estáticas. Escriba primero las variables de clases públicas y, a continuación, las variables de clases privadas.
- Añada variables de instancia. Escriba primero las variables de miembros públicas y, a continuación, las variables de miembros privadas.

- Añada la sentencia constructora, como la del siguiente ejemplo:
  ```
  public function UserClass(username:String, password:String) {...}
  ```
- Escriba los métodos. Agrupe los métodos por la funcionalidad y no por la accesibilidad o el ámbito. Esta forma de organizar los métodos ayuda a mejorar la legibilidad y claridad del código.
- Escriba los métodos getter/setter (captador/definidor) en el archivo de clase.

En el siguiente ejemplo se muestra una clase sencilla de ActionScript denominada User.

### Para crear archivos de clases:

1. Seleccione Archivo > Nuevo, elija Archivo ActionScript y haga clic en Aceptar.
2. Seleccione Archivo > Guardar como y asigne al nuevo archivo el nombre **User.as**.
3. Escriba el siguiente código ActionScript en la ventana Script:

```
/**
  User class
  author: John Doe
  version: 0.8
  modified: 08/21/2005
  copyright: Macromedia, Inc.

  Este código define una clase User personalizada que permite crear
  nuevos usuarios y especificar información de inicio de sesión del
  usuario.
*/

class User {
  // variables de instancias privadas
  private var __username:String;
  private var __password:String;

  // sentencia constructora
  public function User(p_username:String, p_password:String) {
    this.__username = p_username;
    this.__password = p_password;
  }

  public function get username():String {
    return this.__username;
  }
  public function set username(value:String):Void {
    this.__username = value;
  }

  public function get password():String {
    return this.__password;
  }
  public function set password(value:String):Void {
    this.__password = value;
  }
}
```

4. Guarde los cambios en el archivo de clase.

   El fragmento de código anterior comienza con un *comentario de documentación* estandarizado que especifica el nombre de la clase, el autor, la versión, la fecha en la que se modificó por última vez, la información de copyright y una breve descripción de lo que hace la clase.

   La sentencia constructora de la clase User toma dos parámetros: `p_username` y `p_password`, que se copian en las variables de instancias privadas de la clase `__username` y `__password`. El resto del código de la clase define las propiedades getter (captador) y setter (definidor) para las variables de instancias privadas. Si desea crear una propiedad de sólo lectura, deberá definir una función getter, pero no una función setter. Por ejemplo, si desea asegurarse de que un nombre de usuario no pueda cambiarse después de que se haya definido, deberá eliminar la función setter `username` del archivo de la clase User.

5. Seleccione Archivo > Nuevo y seleccione Documento de Flash.

6. Seleccione Archivo > Guardar como y asigne al archivo el nombre **user_test.fla**. Guarde el archivo en el mismo directorio que User.as.

7. Escriba el siguiente código ActionScript en el fotograma 1 de la línea de tiempo:

```
import User;
var user1:User = new User("un1", "pw1");
trace("Before:");
trace("\t username = " + user1.username); // un1
trace("\t password = " + user1.password); // pw1
user1.username = "1nu";
user1.password = "1wp";
trace("After:");
trace("\t username = " + user1.username); // 1nu
trace("\t password = " + user1.password); // 1wp
```

   Dado que la clase User que ha creado anteriormente es muy básica, el código ActionScript del documento de Flash es también muy sencillo. La primera línea de código importa la clase personalizada User en el documento de Flash. La importación de la clase User le permite utilizar la clase como tipo de datos personalizado.

   Una sola instancia de la clase User se define y asigna a una variable denominada `user1`. Se asigna un valor al objeto User `user1` y se define el `username` (nombre de usuario) `un1` y la `password` (contraseña) `pw1`. Las dos sentencias `trace` siguientes muestran el valor actual de `user1.username` y `user1.password` mediante las funciones getter (captador) de la clase User, que en ambos casos devuelven cadenas. Los dos líneas siguientes utilizan las funciones setter (definidor) de la clase User para establecer nuevos valores para las variables `username` y `password`. Finalmente, se trazan los valores de `username` y `password` en el panel Salida. Las sentencias `trace` muestran los valores modificados que se establecen empleando las funciones setter.

8. Guarde el archivo FLA y luego seleccione Control > Probar película para probar los archivos.

   El resultado de las sentencias `trace` se ve en el panel Salida. En los siguientes ejemplos, estos archivos se utilizan en una aplicación.

Un archivo de ejemplo en su disco duro muestra la forma de crear un menú dinámico con datos XML y un archivo de clase personalizado. El ejemplo llama al constructor `XmlMenu()` de ActionScript y le pasa dos parámetros: la ruta al archivo de menú XML y una referencia a la línea de tiempo actual. El resto de la funcionalidad reside en un archivo de clase personalizado, XmlMenu.as.

El archivo de origen de muestra, xmlmenu.fla, se puede encontrar en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\XML_Menu.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/XML_Menu.

# Utilización de clases personalizadas en una aplicación

En "Escritura de archivos de clases personalizadas" en la página 205, hemos creado un archivo de clase personalizado. En las siguientes secciones, utilizará ese archivo de clase en una aplicación. Como mínimo, este flujo de trabajo de creación de clases incluye los siguientes pasos:

1. Definir una clase en un archivo de clase ActionScript externo. Para obtener información sobre la definición y escritura de un archivo de clase, consulte "Escritura de archivos de clases personalizadas" en la página 205.
2. Guardar el archivo de clase en un directorio de rutas de clases designado (una ubicación en donde Flash busque las clases) o en el mismo directorio que el archivo FLA de la aplicación. Para más información sobre la configuración de la ruta de clases, consulte "Configuración y modificación de la ruta de clases" en la página 211. Para ver una comparación y obtener más información sobre la importación de archivos de clases, consulte "Importación de archivos de clases" en la página 209.
3. Crear una instancia de la clase en otro script, ya sea en un documento FLA o en un archivo de script externo, o crear una subclase basada en la clase original. Para más información sobre la creación de una instancia de una clase, consulte "Creación de instancias de clases en un ejemplo" en la página 250.

En las siguientes secciones de este capítulo se incluyen ejemplos de código que puede utilizar para familiarizarse con la creación de clases en ActionScript 2.0. Si no está familiarizado con ActionScript 2.0, consulte el Capítulo 10, “Datos y tipos de datos”, en la página 327 y el Capítulo 4, “Principios básicos de la sintaxis y el lenguaje”, en la página 75.

Para más información sobre el trabajo con clases personalizadas, consulte los siguientes temas:


- “Importación de archivos de clases” en la página 209
- “Utilización de un archivo de clase en Flash” en la página 214
- “Utilización de métodos y propiedades de un archivo de clase” en la página 215
- “Miembros de clase” en la página 221
- “Métodos getter (captador) y setter (definidor)” en la página 226
- “Cómo resuelve el compilador las referencias de clases” en la página 214
- “Clases dinámicas” en la página 229
- “Utilización de encapsulado” en la página 232
- “Utilización de la palabra clave this en clases” en la página 233


Un archivo de ejemplo en su disco duro muestra la forma de crear un menú dinámico con datos XML y un archivo de clase personalizado. El ejemplo llama al constructor `XmlMenu()` de ActionScript y le pasa dos parámetros: la ruta al archivo de menú XML y una referencia a la línea de tiempo actual. El resto de la funcionalidad reside en un archivo de clase personalizado, XmlMenu.as.

El archivo de origen de muestra, xmlmenu.fla, se puede encontrar en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\XML_Menu.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/XML_Menu.

## Importación de archivos de clases

Para utilizar una clase o una interfaz que ha definido, Flash debe localizar los archivos ActionScript externos que contienen la definición de interfaz o clase a fin de que pueda importar el archivo. La lista de directorios en los que Flash busca las definiciones de clases, interfaces, funciones y variables se denomina *ruta de clases*. Flash tiene tipos de rutas de clases: una ruta de clases global y una ruta de clases de documento:

- La **ruta de clases global** es una ruta de clases que comparten todos los documentos de Flash. Se establece en el cuadro de diálogo Preferencias (Edición > Preferencias (Windows) o Flash > Preferencias (Macintosh), seleccione ActionScript de la lista Categoría y haga clic en Configuración de ActionScript 2.0).

- La **ruta de clases de documento** es una ruta de clases que se define específicamente para un único documento de Flash. Se establece en el cuadro de diálogo Configuración de publicación (Archivo > Configuración de publicación, seleccione la ficha Flash y haga clic en el botón Configuración).

Al importar archivos de clases, se aplican las siguientes reglas:

- Las sentencias `import` pueden encontrarse en las siguientes ubicaciones:
  - En cualquier lugar antes de la definición de clases en los archivos de clase
  - En cualquier lugar de scripts de fotogramas o de objetos
  - En cualquier lugar de archivos ActionScript que incluya en una aplicación (con la sentencia `#include`).
- Las definiciones empaquetadas individuales se importan mediante la siguiente sintaxis:
  ```
  import flash.display.BitmapData;
  ```
- Puede importar paquetes enteros utilizando la sintaxis de comodín:
  ```
  import flash.display.*;
  ```

También puede incluir código ActionScript en un archivo de documento de Flash (FLA) mediante una sentencia `include`. Las siguientes reglas se aplican a la sentencia `include`:

- Las sentencias `include` son básicamente una copia pegada del contenido situado dentro del archivo ActionScript incluido.
- Las sentencias `include` dentro de los archivos de clase ActionScript son relativos al subdirectorio que contiene el archivo.
- Una sentencia `include` de un archivo FLA sólo puede incluir código que sea válido dentro de archivos FLA, lo que también es aplicable a cualquier otro lugar en el que puedan situarse sentencias `include`. Por ejemplo, si hay una sentencia `include` dentro de una definición de clase, sólo pueden estar presentes las definiciones de propiedades y métodos en el archivo ActionScript incluido:

```
// Foo.as
class Foo {
  #include "FooDef.as"
}

// FooDef.as:
var fooProp;
function fooMethod() {}
trace("Foo"); // Las definiciones de clase no admiten este tipo de
  declaración.
```

Para más información sobre la sentencia `include`, consulte %{directiva #include}% en *Referencia del lenguaje ActionScript 2.0*. Para más información sobre rutas de clases, consulte "Configuración y modificación de la ruta de clases" en la página 211.

## Configuración y modificación de la ruta de clases

Para utilizar una clase o una interfaz que ha definido, Flash debe localizar los archivos ActionScript externos que contienen la definición de interfaz o clase. La lista de directorios en los que Flash busca las definiciones de interfaces y clases se denomina *ruta de clases*.

Al crear un archivo de clase ActionScript, es necesario guardar el archivo en uno de los directorios especificados en la ruta de clases o en un subdirectorio de ésta. (Si lo desea, puede modificar la ruta de clases de forma que incluya la ruta de directorio deseada). De lo contrario, Flash no podrá resolver, es decir, localizar la clase o interfaz especificada en el script. Los subdirectorios que se crean dentro de un directorio de rutas de clases se denominan paquetes y permiten organizar las clases. (Para más información sobre paquetes, consulte "Creación y empaquetado de archivos de clases" en la página 236.)

Flash tiene dos tipos de rutas de clases: una *ruta de clases global* y una *ruta de clases de documento*. La ruta de clases global es una ruta de clases que comparten todos los documentos de Flash. La ruta de clases de documento es una ruta de clases que se define específicamente para un único documento de Flash.

La ruta de clases global se aplica a los archivos externos ActionScript y FLA y se configura en el cuadro de diálogo Preferencias (Edición > Preferencias (Windows) o Flash > Preferencias (Macintosh), seleccione ActionScript de la lista Categoría y haga clic en Configuración de ActionScript 2.0). Puede establecer la ruta de clases de documento en el cuadro de diálogo Configuración de publicación del documento de Flash (Archivo > Configuración de publicación, seleccione Flash y haga clic en el botón Configuración).

**NOTA**

Al hacer clic en el botón Revisar sintaxis situado encima del panel Script mientras se edita un archivo ActionScript, el compilador sólo consulta la ruta de clases global. Los archivos ActionScript no están asociados a archivos FLA en modo de edición y carecen de ruta de clases propia.

### Utilización de una ruta de clases global

La ruta de clases global es una ruta de clases que comparten todos los documentos de Flash.

La ruta de clases global se puede modificar a través del cuadro de diálogo Preferencias. Para modificar la configuración de ruta de clases de documentos, utilice el cuadro de diálogo Configuración de publicación para el archivo FLA. En ambos casos, puede añadir rutas de directorios absolutas (por ejemplo, `C:/my_classes`) y rutas de directorios relativas (por ejemplo, `../my_classes` o "."). El orden de los directorios en el cuadro de diálogo refleja el orden en el que se buscan.

De manera predeterminada, la ruta de clase global contiene una ruta absoluta y una ruta relativa. La ruta absoluta se expresa mediante $(LocalData)/Classes en el cuadro de diálogo Preferencias. A continuación se muestra la ubicación de la ruta absoluta:

- Windows: Disco duro\Documents and Settings*usuario*\Configuración local\Datos de programa\Macromedia\Flash 8*idioma*\Configuration\Classes.
- Macintosh: Disco duro/Users/*usuario*/Library/Application Support/Macromedia/Flash 8/*idioma*/Configuration/Classes.

| NOTA | No elimine la ruta de clases global absoluta. Flash utiliza esta ruta de clases para obtener acceso a clases incorporadas. Si elimina por error esta ruta de clases, restáurela añadiendo $(LocalData)/Classes como nueva ruta de clases. |
|---|---|

La parte de ruta relativa de la ruta de clases global se expresa mediante un solo punto (.) y señala al directorio de documentos actual. Tenga en cuenta que las rutas de clases relativas pueden señalar a distintos directorios, según la ubicación del documento que se está compilando o publicando.

Puede seguir estos pasos para añadir una ruta de clases global o editar una ruta de clases existente.

### Para modificar la ruta de clases global:

1. Seleccione Edición > Preferencias (Windows) o Flash > Preferencias (Macintosh) y abra el cuadro de diálogo Preferencias.
2. Haga clic en ActionScript en la columna izquierda y, seguidamente, haga clic en el botón Configuración de ActionScript 2.0.
3. Haga clic en el botón Buscar ruta para ir al directorio que desea añadir.
4. Vaya a la ruta que desea añadir y haga clic en Aceptar.

### Para eliminar un directorio de la ruta de clases:

1. Seleccione la ruta en la lista de ruta de clases.
2. Haga clic en el botón Quitar ruta seleccionada.

| NOTA | No elimine la ruta de clases global absoluta. Flash utiliza esta ruta de clases para obtener acceso a clases incorporadas. Si elimina por error esta ruta de clases, puede restaurarla añadiendo $(LocalData)/Classes como nueva ruta de clases. |
|---|---|

Para más información sobre importación de paquetes, consulte "Utilización de paquetes" en la página 199.

## Utilización de una ruta de clases de documento

La ruta de clases de documentos sólo es aplicable a archivos FLA. La ruta de clases de documento se establece en el cuadro de diálogo Configuración de publicación para un archivo FLA concreto (Archivo > Configuración de publicación, haga clic en la ficha Flash y seleccione Configuración de ActionScript 2.0). De forma predeterminada, la ruta de clases de documentos está vacía. Al crear un archivo FLA en un directorio, dicho directorio se convierte en un directorio designado de ruta de clases.

Al crear clases, es posible que algunas veces desee almacenarlas en un directorio que posteriormente pueda añadir a la lista de directorios de ruta de clases global en los siguientes casos:

- Si dispone de un conjunto de clases de utilidad que utilizan todos sus proyectos
- Si desea revisar la sintaxis del código (haga clic en el botón Revisar sintaxis) dentro del archivo ActionScript externo

La creación de un directorio evita la pérdida de clases personalizadas si alguna vez desinstala Flash y vuelve a instalarlo, especialmente si se elimina o sobrescribe el directorio predeterminado de la ruta de clases global, ya que perdería todas las clases almacenadas en dicho directorio.

Por ejemplo, podría crear un directorio como el siguiente para sus clases personalizadas:

- Windows: Disco duro\Documents and Settings*usuario*\clases personalizadas.
- Macintosh: Disco duro/Users/*usuario*/clases personalizadas.

Posteriormente, añadiría esta ruta a la lista de rutas de clases globales (consulte "Utilización de una ruta de clases global" en la página 211).

Cuando Flash intenta resolver referencias de clases en un script FLA, primero busca la ruta de clases del documento especificada para dicho archivo FLA. Si Flash no encuentra la clase en esa ruta de clases o ésta está vacía, la busca en la ruta de clases global. Si Flash no encuentra la clase en la ruta de clases global, se produce un error de compilador.

### Para modificar la ruta de clase en el documento:

1. Seleccione Archivo > Configuración de publicación para abrir el cuadro de diálogo Configuración de publicación.
2. Haga clic en la ficha Flash.
3. Haga clic en el botón Configuración situado junto al menú emergente Versión de ActionScript.

4. Puede escribir manualmente una ruta de archivo o hacer clic en el botón Buscar ruta para ir al directorio que desea añadir a la ruta de clases.

| NOTA | Para editar un directorio de ruta de clases existente, seleccione la ruta en la lista de rutas de clases, haga clic en el botón Buscar ruta, busque el directorio que desee añadir y haga clic en Aceptar. |
|---|---|

| NOTA | Para eliminar un directorio de la ruta de clases, seleccione la ruta en la lista de rutas de clases y haga clic en el botón Quitar ruta seleccionada (-). |
|---|---|

Para más información sobre paquetes, consulte "Paquetes" en la página 198.

## Cómo resuelve el compilador las referencias de clases

Cuando Flash intenta resolver referencias de clases en un script FLA, primero busca la ruta de clases del documento especificada para dicho archivo FLA. Si no se encuentra la clase en dicha ruta de clases, o si la ruta de clases está vacía, Flash busca la ruta de clases global. Si no se encuentra la clase en la ruta de clases global, se produce un error de compilador.

En Flash Professional, al hacer clic en el botón Revisar sintaxis durante la edición de un archivo ActionScript, el compilador busca sólo en la ruta de clases global; los archivos ActionScript no se asocian a los FLA en modo de edición y carecen de una ruta de clases propia.

## Utilización de un archivo de clase en Flash

Para crear una instancia de una clase de ActionScript, utilice el operador `new` para invocar la función constructora de la clase. La función constructora tiene siempre el mismo nombre que la clase y devuelve una instancia de la clase, que normalmente se asigna a una variable. Por ejemplo, si utilizara la clase User de "Escritura de archivos de clases personalizadas" en la página 205, escribiría el siguiente código para crear un nuevo objeto User:

```
var firstUser:User = new User();
```

| NOTA | En algunos casos, no es necesario crear una instancia de una clase para utilizar sus propiedades y métodos. Para más información sobre miembros (estáticos) de clase, consulte "Miembros de clase (estáticos)" en la página 271 y "Métodos y propiedades estáticos" en la página 219. |
|---|---|

Utilice el operador de punto (.) para acceder al valor de una propiedad en una instancia. Escriba el nombre de la instancia a la izquierda del punto y el nombre de la propiedad en el lado derecho. Por ejemplo, en la siguiente sentencia, `firstUser` es la instancia y `username` es la propiedad:

```
firstUser.username
```

También puede utilizar las clases de nivel superior o las clases incorporadas que conforman el lenguaje ActionScript en un documento de Flash. Por ejemplo, el siguiente código crea un nuevo objeto Array y luego muestra su propiedad `length`:

```
var myArray:Array = new Array("apples", "oranges", "bananas");
trace(myArray.length); // 3
```

Para más información sobre el uso de clases personalizadas en Flash, consulte "Ejemplo: Utilización de archivos de clases personalizadas en Flash" en la página 248. Para obtener información sobre la función constructora, consulte "Escritura de la función constructora" en la página 239.

## Utilización de métodos y propiedades de un archivo de clase

En OOP, los miembros (propiedades o métodos) de una clase pueden ser miembros de la instancia o miembros de la clase. Los miembros de instancias se crean para cada instancia de la clase; se definen para el prototipo de la clase cuando se inicializan en la definición de clase. Por contra, los miembros de clases se crean una vez por clase. (Los miembros de clases también se denominan miembros estáticos.)

Las propiedades son atributos que definen un objeto. Por ejemplo, `length` es una propiedad de todas las matrices que especifica el número de elementos de la matriz. Los métodos son funciones que se asocian a una clase. Para más información sobre funciones y métodos, consulte el Capítulo 5, "Funciones y métodos", en la página 169.

En el siguiente ejemplo se muestra cómo crear un método en un archivo de clase:

```
class Sample {
  public function myMethod():Void {
    trace("myMethod");
  }
}
```

A continuación, invocaremos dicho método en el documento. Para invocar un método de instancia o acceder a una propiedad de instancia, debe hacer referencia a una instancia de la clase. En el siguiente ejemplo, `picture01`, que es una instancia de la clase personalizada Picture (disponible en el siguiente ejercicio), invoca el método `showInfo()`:

```
var img1:Picture = new Picture("http://www.helpexamples.com/flash/images/
  image1.jpg");
// Invoca el método showInfo().
img1.showInfo();
```

En el siguiente ejemplo se muestra cómo escribir una clase personalizada Picture que contenga diversos datos sobre una foto.

### Para utilizar las clases Picture y PictureClass en un archivo FLA:

1. Seleccione Archivo > Nuevo y, a continuación, Archivo ActionScript. Guarde el documento como **Picture.as** y haga clic en Aceptar.

   Deberá guardar la clase personalizada Picture en este documento.

2. Escriba el siguiente código ActionScript en la ventana Script:

```
/**
  Picture class
  author: John Doe
  version: 0.53
  modified: 6/24/2005
  copyright: Macromedia, Inc.

  La clase Picture se utiliza como contenedor de una imagen y su URL.
*/

class Picture {
  private var __infoObj:Object;

  public function Picture(src:String) {
    this.__infoObj = new Object();
    this.__infoObj.src = src;
  }

  public function showInfo():Void {
    trace(this.toString());
  }
  private function toString():String {
    return "[Picture src=" + this.__infoObj.src + "]";
  }

  public function get src():String {
    return this.__infoObj.src;
  }
  public function set src(value:String):Void {
    this.__infoObj.src = value;
  }
}
```

3. Guarde el archivo de ActionScript.
4. Seleccione Archivo > Nuevo y haga clic en Documento de Flash para crear un nuevo archivo FLA. Guárdelo como **picture_test.fla** en el mismo directorio en el que guardó el archivo de clase Picture.
5. Escriba el siguiente código ActionScript en el fotograma 1 de la línea de tiempo:

```
var picture1:Picture = new Picture("http://www.helpexamples.com/flash/
  images/image1.jpg");
picture1.showInfo();
this.createEmptyMovieClip("img_mc", 9);
img_mc.loadMovie(picture1.src);
```

6. Guarde el documento de Flash.
7. Seleccione Control > Probar película para probar el documento.

   El panel Salida muestra este texto:

```
[Picture src=http://www.helpexamples.com/flash/images/image1.jpg]
```

Un archivo de ejemplo en su disco duro muestra la forma de crear un menú dinámico con datos XML y un archivo de clase personalizado. El ejemplo llama al constructor `XmlMenu()` de ActionScript y le pasa dos parámetros: la ruta al archivo de menú XML y una referencia a la línea de tiempo actual. El resto de la funcionalidad reside en un archivo de clase personalizado, XmlMenu.as.

El archivo de origen de muestra, xmlmenu.fla, se puede encontrar en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\XML_Menu.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/XML_Menu.

## Métodos y propiedades (miembros) públicos, privados y estáticos

Al escribir archivos de clases de ActionScript en un archivo de script externo, podrá crear cuatro tipos de métodos y propiedades: públicos, privados, estáticos y estáticos privados. Estos métodos y propiedades definen la forma en que Flash puede acceder a las variables y le permiten especificar qué partes del código pueden acceder a determinados métodos o propiedades.

Al crear aplicaciones basadas en clases, con independencia del tamaño de la aplicación, es particularmente importante tener en cuenta si un método o propiedad debe ser privado o público. Al tener en cuenta este aspecto, conseguirá que el código sea lo más seguro posible. Por ejemplo, si está creando una clase User, puede que desee impedir que las personas que usen la clase puedan cambiar un ID de usuario. Al establecer la propiedad de la clase (a veces denominada *miembro de instancia*) como `private`, podrá limitar el acceso a la propiedad al código situado dentro de la clase o las subclases de dicha clase, lo que significa que ningún usuario podrá cambiar la propiedad directamente.

### Métodos y propiedades públicos

La palabra clave `public` especifica que una variable o función está disponible para cualquier origen de llamada. Dado que las variables y funciones son públicas de forma predeterminada, la palabra clave `this` se utiliza principalmente por motivos de estilo y legibilidad, lo que indica que la variable existe en el ámbito actual. Por ejemplo, quizá desee utilizar la palabra clave `this` para mantener la coherencia de un bloque de código que contenga además variables privadas o estáticas. La palabra clave `this` se puede utilizar con la palabra clave pública o privada.

La siguiente clase Sample (de muestra) ya tiene un método público denominado `myMethod()`:

```
class Sample {
  private var ID:Number;
  public function myMethod():Void {
    this.ID = 15;
    trace(this.ID); // 15
    trace("myMethod");
  }
}
```

Si desea añadir una propiedad `public`, utilice la palabra “public” en lugar de “private”, como puede observarse en el siguiente código de muestra:

```
class Sample {
  private var ID:Number;
  public var email:String;
  public function myMethod():Void {
    trace("myMethod");
  }
}
```

Dado que la propiedad `email` es pública, puede cambiarla dentro de la clase Sample o directamente dentro de un FLA.

## Métodos y propiedades privados

La palabra clave `private` especifica que una variable o función está únicamente disponible para la clase que la declara o define o para las subclases de dicha clase. De manera predeterminada, una variable o función es pública y está disponible para cualquier origen de llamada. Utilice la palabra clave `this` si desea restringir el acceso a una variable o función, como puede observarse en el siguiente ejemplo:

```
class Sample {
  private var ID:Number;
  public function myMethod():Void {
    this.ID = 15;
    trace(this.ID); // 15
    trace("myMethod");
  }
}
```

Si desea añadir una propiedad privada a la clase anterior, simplemente deberá usar la palabra clave `private` antes de la palabra clave `var`.

Si intenta acceder a la propiedad privada `ID` desde fuera de la clase Sample, obtendrá un error de compilación y un mensaje en el panel Salida. El mensaje indica que el miembro es privado y no permite el acceso.

## Métodos y propiedades estáticos

La palabra clave `static` especifica que una variable o función se crea únicamente una vez por cada clase, en lugar de crearse en cada objeto basado en dicha clase. Puede acceder a un miembro de clase estático sin crear una instancia de la clase. Los métodos y propiedades estáticos pueden establecerse en el ámbito público o privado.

Los miembros estáticos, también conocidos como *miembros de clase*, se asignan a la clase, no a una instancia de la clase. Para invocar un método de la clase o para acceder a una propiedad de la clase, debe hacer referencia al nombre de la clase en lugar de a una instancia concreta de la misma, como se muestra en el siguiente código:

```
trace(Math.PI / 8); // 0.392699081698724
```

Si escribe sólo esta línea de código en el panel Script del panel Acciones, se ve la sentencia trace resultante en el panel Salida.

Por ejemplo, en el anterior ejemplo de la clase Sample, podría crear una variable estática para supervisar cuántas instancias de la clase se crean, como se muestra en el siguiente código:

```
class Sample {
  public static var count:Number = 0;
  private var ID:Number;
  public var email:String;
  public function Sample() {
    Sample.count++;
    trace("count updated: " + Sample.count);
  }
  public function myMethod():Void {
    trace("myMethod");
  }
}
```

Cada vez que crea una instancia nueva de la clase Sample, el método constructor traza el número total de instancias de clase Sample definidas hasta el momento.

Algunas de las clases de ActionScript del más alto nivel tienen miembros de clase (o miembros estáticos), como ya hemos visto anteriormente en esta sección al llamar a la propiedad `Math.PI`. Se accede o se invoca a los miembros de clase (propiedades y métodos) desde el propio nombre de la clase, no desde una instancia de la misma. Por consiguiente, no se crea una instancia de la clase para utilizar estas propiedades y métodos.

Por ejemplo, la clase Math del más alto nivel consta sólo de propiedades y métodos estáticos. Para llamar a cualquiera de sus métodos, no debe crear una instancia de la clase Math. En su lugar, simplemente llame a los métodos en la propia clase Math. El siguiente código llama al método `sqrt()` de la clase Math:

```
var squareRoot:Number = Math.sqrt(4);
trace(squareRoot); // 2
```

El código siguiente invoca el método `max()` de la clase Math, que determina, entre dos números, cuál de ellos es el mayor:

```
var largerNumber:Number = Math.max(10, 20);
trace(largerNumber); // 20
```

Para más información sobre la creación de miembros de clases, consulte “Miembros de clase” en la página 221 y “Utilización de miembros de clase” en la página 224.

Un archivo de ejemplo en su disco duro muestra la forma de crear un menú dinámico con datos XML y un archivo de clase personalizado. El ejemplo llama al constructor `XmlMenu()` de ActionScript y le pasa dos parámetros: la ruta al archivo de menú XML y una referencia a la línea de tiempo actual. El resto de la funcionalidad reside en un archivo de clase personalizado, XmlMenu.as.

El archivo de origen de muestra, xmlmenu.fla, se puede encontrar en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\XML_Menu.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/XML_Menu.

## Miembros de clase

La mayoría de los miembros (métodos y propiedades) descritos hasta ahora en este capítulo son de un tipo denominado *miembros de instancias*. Por cada miembro de instancia, existe una copia exclusiva de dicho miembro en cada instancia de la clase. Por ejemplo, la variable de miembro `email` de la clase Sample tiene un miembro de instancia, porque cada persona tiene una dirección de correo electrónico diferente.

Otro tipo de miembro es un *miembro de clase*. Sólo hay una copia de un miembro de clase, que se utiliza para toda la clase. Todas las variables declaradas dentro de una clase, pero fuera de una función, son propiedades de la clase. En el siguiente ejemplo, la clase Person tiene dos propiedades, `age` y `username`, de tipo Number y String, respectivamente:

```
class Person {
  public var age:Number;
  public var username:String;
}
```

Asimismo, todas las funciones declaradas dentro de una clase se consideran un método de la clase. En el ejemplo de la clase Person, puede crear un método denominado `getInfo()`:

```
class Person {
  public var age:Number;
  public var username:String;
  public function getInfo():String {
    // Definición del método getInfo()
  }
}
```

En el fragmento de código anterior, el método `getInfo()` de la clase Person, así como las propiedades `age` y `username`, son miembros de instancia públicos. La propiedad `age` no sería un buen miembro de clase, dado que cada persona tiene una edad diferente. Sólo deberían ser miembros de clase las propiedades y métodos compartidos por todos los individuos de la clase.

Supongamos que desea que todas las clases tengan una variable `species` que indique el nombre latino de cada especie a la que representa la clase. Por cada objeto Person, la especie es *Homo sapiens*. Sería un despilfarro almacenar una copia exclusiva de la cadena `"Homo sapiens"` por cada instancia de la clase, por lo que este miembro debe ser un miembro de clase.

Los miembros de clases se declaran con la palabra clave `static`. Por ejemplo, puede declarar el miembro de clase species con el siguiente código:

```
class Person {
  public static var species:String = "Homo sapiens";
  // ...
}
```

También puede declarar métodos de una clase para que sean estáticos, como se muestra en el siguiente código:

```
public static function getSpecies():String {
  return Person.species;
}
```

Los métodos estáticos sólo pueden acceder a las propiedades estáticas, no a las propiedades de instancia. Por ejemplo, el siguiente código provoca un error de compilador porque el método de la clase `getAge()` hace referencia a la variable de instancia `age`:

```
class Person {
  public var age:Number = 15;
  // ...
  public static function getAge():Number {
    return age; /* **Error**: No se puede acceder a variables de instancia
  en funciones estáticas. */
  }
}
```

Para resolver este problema, podría hacer que el método fuera un método de instancia o hacer que la variable fuera una variable de clase.

Para más información sobre los miembros de clase (también denominados propiedades estáticas), consulte "Métodos y propiedades estáticos" en la página 219.

Un archivo de ejemplo en su disco duro muestra la forma de crear un menú dinámico con datos XML y un archivo de clase personalizado. El ejemplo llama al constructor `XmlMenu()` de ActionScript y le pasa dos parámetros: la ruta al archivo de menú XML y una referencia a la línea de tiempo actual. El resto de la funcionalidad reside en un archivo de clase personalizado, XmlMenu.as.

El archivo de origen de muestra, xmlmenu.fla, se puede encontrar en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\XML_Menu.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/XML_Menu.

## Utilización del patrón de diseño Singleton

Una forma habitual de usar miembros de clase es con el patrón de diseño Singleton. Un *patrón de diseño* define un enfoque formal a la hora de estructurar el código. Normalmente, podría estructurar un patrón de diseño como solución para un problema de programación común. Existen muchos patrones de diseño establecidos, como Singleton. El patrón de diseño Singleton garantiza que una clase tenga sólo una instancia y ofrece una forma de acceder globalmente a la instancia. Para obtener información detallada sobre el patrón de diseño Singleton, consulte www.macromedia.com/devnet/mx/coldfusion/articles/design_patterns.html.

Muchas veces hay situaciones en las que necesita un objeto exactamente de un tipo concreto en un sistema. Por ejemplo, en un juego de ajedrez sólo hay un tablero, al igual que en cada país sólo hay una capital. Aunque sólo haya un objeto, debe encapsular la funcionalidad de este objeto en una clase. Sin embargo, puede que necesite administrar y acceder a la única instancia de dicho objeto. Puede hacerlo empleando una variable global, aunque las variables globales no suelen ser muy convenientes para la mayoría de los proyectos. Un enfoque más adecuado consiste en hacer que la clase administre la única instancia del objeto en sí misma empleando miembros de clase. En el siguiente ejemplo se muestra el uso de un patrón de diseño Singleton típico, donde la instancia Singleton se crea sólo una vez.

### Para utilizar el patrón de diseño Singleton:

1. Seleccione Archivo > Nuevo y, a continuación, Archivo ActionScript. Guarde el documento como **Singleton.as**.
2. Escriba el siguiente código ActionScript en la ventana Script:

```
/**
  Singleton class
  author: John Doe
  version: 0.53
  modified: 6/24/2008
  copyright: Macromedia, Inc.
*/

class Singleton {
  private static var instance:Singleton = null;
  public function trackChanges():Void {
    trace("tracking changes.");
  }
  public static function getInstance():Singleton {
    if (Singleton.instance == null) {
      trace("creating new Singleton.");
      Singleton.instance = new Singleton();
    }
    return Singleton.instance;
  }
}
```

3. Guarde el documento Singleton.as.
4. Seleccione Archivo > Nuevo y elija Documento de Flash para crear un nuevo archivo FLA; a continuación, guárdelo como **singleton_test.fla** en el mismo directorio en el que ha guardado el archivo de clase Singleton.
5. Escriba el siguiente código ActionScript en el fotograma 1 de la línea de tiempo:

```
Singleton.getInstance().trackChanges(); // rastreo de cambios.

var s:Singleton = Singleton.getInstance(); // rastreo de cambios.
s.trackChanges();
```

6. Guarde el documento de Flash.
7. Seleccione Control > Probar película para probar el documento.

El objeto Singleton no se crea hasta que es necesario (es decir, hasta que otro código lo pida mediante una llamada al método `getInstance()`). Esto normalmente se conoce como *creación perezosa* y puede contribuir a alcanzar una mayor eficiencia en el código en numerosas circunstancias.

Recuerde que no debe utilizar en la aplicación ni muy pocos archivos de clase ni demasiados, ya que al hacerlo puede generar archivos de clase mal diseñados, lo que no resultaría beneficioso para el rendimiento de la aplicación ni para el flujo de trabajo. Debe intentar siempre utilizar archivos de clase en lugar de colocar código en otros lugares (como las líneas de tiempo); sin embargo, evite crear muchas clases que sólo cuenten con una pequeña cantidad de funcionalidad o solamente algunas clases que gestionen mucha funcionalidad. En estos dos casos indicaría un mal diseño.

## Utilización de miembros de clase

Una forma de emplear miembros (estáticos) de clase es mantener información de estado sobre una clase y sus instancias. Por ejemplo, suponga que desea mantener un seguimiento del número de instancias que se han creado a partir de una clase concreta. Una forma fácil de hacerlo es utilizar una propiedad de clase que se incremente cada vez que se cree una nueva instancia.

En el siguiente ejemplo, creará una clase denominada Widget que define un único contador de instancias estáticas denominado `widgetCount`. Cada vez que se crea una nueva instancia de la clase, el valor de `widgetCount` se incrementa en `1` y el valor actual de `widgetCount` se muestra en el panel Salida.

### Para crear un contador de instancias mediante una variable de clase:

1. Seleccione Archivo > Nuevo, elija Archivo ActionScript y haga clic en Aceptar.
2. Escriba el siguiente código en la ventana Script:

```
class Widget {
  //Inicialice la variable de clase
  public static var widgetCount:Number = 0;
  public function Widget() {
    Widget.widgetCount++;
    trace("Creating widget #" + Widget.widgetCount);
  }
}
```

   La variable `widgetCount` se declara como estática y, por lo tanto, sólo se inicializa a 0 una vez. Cada vez que se llama a la sentencia constructora de la clase Widget, se añade 1 a `widgetCount` y luego se muestra el número de la instancia actual que se está creando.

3. Guarde el archivo como **Widget.as**.
4. Seleccione Archivo > Nuevo y elija Documento de Flash para crear un nuevo archivo FLA. Guárdelo como **widget_test.fla** en el mismo directorio que Widget.as.
5. En widget_test.fla, escriba el siguiente código en el fotograma 1 de la línea de tiempo:

```
// Antes de crear instancias de la clase,
// Widget.widgetCount es cero (0).
trace("Widget count at start: " + Widget.widgetCount); // 0
var widget1:Widget = new Widget(); // 1
var widget2:Widget = new Widget(); // 2
var widget3:Widget = new Widget(); // 3
trace("Widget count at end: " + Widget.widgetCount); // 3
```

6. Guarde los cambios en widget_test.fla.
7. Seleccione Control > Probar película para probar el archivo.

   Flash muestra la siguiente información en el panel Salida:

```
Widget count at start: 0
Creating widget # 1
Creating widget # 2
Creating widget # 3
Widget count at end: 3
```

# Métodos getter (captador) y setter (definidor)

Los métodos getter y setter son métodos de acceso, lo que significa que generalmente son una interfaz pública para cambiar miembros de clases privadas. Los métodos getter y setter se utilizan para definir una propiedad. A los métodos getter y setter se accede como propiedades situadas fuera de la clase, aunque las defina dentro de la clase como métodos. Dichas propiedades situadas fuera de la clase tienen un nombre diferente al nombre de la propiedad de la clase.

El uso de los métodos getter y setter aporta diversas ventajas, como la posibilidad de crear miembros con funcionalidad sofisticada a los que puede acceder como propiedades. También le permiten crear propiedades de sólo lectura o sólo escritura.

Aunque los métodos getter y setter son útiles, debe tener cuidado de no hacer un uso *abusivo* de ellos, ya que, entre otros problemas, pueden dificultar el mantenimiento del código en determinadas situaciones. Asimismo, proporcionan acceso a la implementación de la clase, como miembros públicos. En OOP, no se aconseja el acceso directo a las propiedades de una clase.

Al escribir clases, se recomienda que las variables de instancia sean privadas siempre que sea posible y que se añadan los correspondientes métodos getter y setter. La razón de ello es que existen casos en los que es posible que no desee permitir que los usuarios cambien determinadas variables en las clases. Por ejemplo, si tiene un método estático privado que rastrea el número de instancias creadas para una clase específica, no querrá que un usuario modifique el contador utilizando el código. Dicha variable sólo deberá incrementar su valor cuando se llame la sentencia constructora. En este caso, podría crear una variable de instancia privada y permitir un método getter solamente para la variable de contador, lo que significa que los usuarios sólo podrán recuperar el valor actual utilizando el método getter y no podrán establecer nuevos valores empleando el método setter. La creación de un método getter sin método setter es una forma sencilla de convertir determinadas variables de la clase en variables de sólo lectura.

## Utilización de métodos getter y setter

La sintaxis de los métodos getter y setter es la siguiente:

- Un método getter no toma ningún parámetro y siempre devuelve un valor.
- Un método setter siempre toma un parámetro y nunca devuelve ningún valor.

Las clases normalmente definen métodos getter que proporcionan acceso de lectura y métodos setter que proporcionan acceso de escritura a una propiedad determinada. Por ejemplo, imagine una clase que contiene una propiedad llamada `userName`:

```
private var userName:String;
```

En lugar de permitir que las instancias de la clase accedan directamente a esta propiedad (por ejemplo, `user.userName = "Buster"`), la clase puede utilizar dos métodos, `getUserName()` y `setUserName()`, que se implementarían como se muestra en el siguiente ejemplo.

### Para utilizar los métodos getter y setter:

1. Seleccione Archivo > Nuevo, elija Archivo ActionScript y haga clic en Aceptar.
2. Escriba el siguiente código en la ventana Script:

```
class Login {
  private var __username:String;
  public function Login(username:String) {
    this.__username = username;
  }
  public function getUserName():String {
    return this.__username;
  }
  public function setUserName(value:String):Void {
    this.__username = value;
  }
}
```

3. Guarde el documento ActionScript como **Login.as**.

   Como puede ver, `getUserName()` devuelve el valor actual de `userName` y `setUserName()` establece el valor de `userName` en el parámetro string pasado al método.

4. Seleccione Archivo > Nuevo y elija Documento de Flash para crear un nuevo archivo FLA. Guárdelo como **login_test.fla** en el mismo directorio que Login.as.
5. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
var user:Login = new Login("RickyM");

// se llama al método getUserName()
var userName:String = user.getUserName();
trace(userName); // RickyM

// se llama al método setUserName()
user.setUserName("EnriqueI");
trace(user.getUserName()); // EnriqueI
```

6. Seleccione Control > Probar película para probar el archivo.

   Flash muestra la siguiente información en el panel Salida:

```
RickyM
EnriqueI
```

Sin embargo, si desea utilizar una sintaxis más concisa, puede utilizar métodos getter y setter implícitos. Los métodos getter y setter implícitos permiten acceder directamente a las propiedades de clases a la vez que siguen las recomendaciones de la programación orientada a objetos (OOP).

Para definir estos métodos, utilice los atributos de los métodos `get` y `set`. Debe crear métodos que obtengan o establezcan el valor de una propiedad, así como añadir la palabra clave `get` o `set` antes del nombre del método, como se muestra en el siguiente ejemplo.

| NOTA | Los métodos getter/setter implícitos son la abreviatura sintáctica del método `Object.addProperty()` utilizado en ActionScript 1.0. |
|---|---|

### Para utilizar los métodos getter y setter implícitos:

1. Seleccione Archivo > Nuevo, elija Archivo ActionScript y haga clic en Aceptar.
2. Escriba el siguiente código en la ventana Script:

```
class Login2 {
  private var __username:String;
  public function Login2(username:String) {
    this.__username = username;
  }
  public function get userName():String {
    return this.__username;
  }
  public function set userName(value:String):Void {
    this.__username = value;
  }
}
```

3. Guarde el documento ActionScript como **as Login2.as**.

   Recuerde que un método getter no debe tomar ningún parámetro. Un método setter debe aceptar exactamente un parámetro necesario. Un método setter puede tener el mismo nombre que un método getter en el mismo ámbito. Los métodos getter y setter no pueden tener los mismos nombres que otras propiedades. Por ejemplo, en el código de ejemplo anterior ha definido los métodos getter y setter denominados `userName`; en este caso, no podrá tener también una propiedad denominada `userName` en la misma clase.
4. Seleccione Archivo > Nuevo y elija Documento de Flash para crear un nuevo archivo FLA. Guárdelo como **login2_test.fla** en el mismo directorio que Login2.as.
5. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
var user:Login2 = new Login2("RickyM");

// Se llama al método "get"
var userNameStr:String = user.userName;
trace(userNameStr); // RickyM

// Se llama al método "set"
user.userName = "EnriqueI";
trace(user.userName); // EnriqueI
```

   A diferencia de los métodos más corrientes, los métodos getter y setter se invocan sin ningún paréntesis ni argumentos. La invocación de los métodos getter y setter se realiza de la misma forma que lo haría con una propiedad por el mismo nombre.

6. Guarde el documento de Flash y seleccione Control > Probar película para probar el archivo.

   Flash muestra la siguiente información en el panel Salida:

```
RickyM
EnriqueI
```

| NOTA | No puede utilizar atributos de métodos getter y setter en declaraciones de métodos de interfaz. |
|---|---|

# Clases dinámicas

Al añadir la palabra clave `dynamic` a una definición de clase, se especifica que los objetos basados en la clase especificada pueden añadir y acceder a las propiedades dinámicas en tiempo de ejecución. Sólo debe crear clases dinámicas si necesita esta funcionalidad específicamente.

La verificación de tipos en clases dinámicas es menos estricto que en clases no dinámicas porque los miembros a los que se accede dentro de la definición de clase y en instancias de la clase no se comparan con los definidos en el ámbito de la clase. Sin embargo, pueden verificarse los tipos return y parameter de las funciones de miembros de clase.

Para obtener información sobre la creación de clases dinámicas, consulte "Creación de clases dinámicas" en la página 229.

## Creación de clases dinámicas

De forma predeterminada, las propiedades y los métodos de una clase están fijos. Es decir, una instancia de una clase no puede crear ni acceder a propiedades ni métodos que la clase no haya declarado o definido originalmente. Por ejemplo, imagine una clase Person que defina dos propiedades, `userName` y `age`.

### Para crear una clase que no es dinámica:

1. Seleccione Archivo > Nuevo, elija Archivo ActionScript y haga clic en Aceptar.
2. Escriba el siguiente código ActionScript en la ventana Script:

```
class Person {
  public var userName:String;
  public var age:Number;
}
```

   Si en otro script se crea una instancia de la clase Person y se intenta acceder a una propiedad de la clase que no existe, el compilador generará un error.
3. Guarde el archivo en el disco duro como **Person.as**.

4. Seleccione Archivo > Nuevo, elija Documento de Flash para crear un nuevo archivo FLA y luego haga clic en Aceptar.
5. Seleccione Archivo > Guardar como, asígnele el nombre **person_test.fla** y guarde el archivo en el mismo directorio que la clase Person creada anteriormente.
6. Añada el siguiente código para crear una nueva instancia de la clase Person (`firstPerson`) e intente asignar un valor a una propiedad denominada `hairColor` (que no existe en la clase Person):

```
var firstPerson:Person = new Person();
firstPerson.hairColor = "blue"; // Error. No hay ninguna propiedad que
  lleve por nombre 'hairColor'.
```

7. Guarde el documento de Flash.
8. Seleccione Control > Probar película para probar el código.

   Este código causa un error de compilador porque la clase Person no declara una propiedad denominada `hairColor`. En la mayoría de los casos, esto es exactamente lo que desea que suceda. Los errores de compilador pueden parecer poco deseables, pero son de gran ayuda para los programadores: los mensajes de error de calidad le ayudan a escribir código correcto al señalar errores en una etapa temprana del proceso de codificación.

Sin embargo, en algunos casos, puede que durante la ejecución desee añadir y acceder a propiedades o métodos de una clase que no se han establecido en la definición de clase original. El modificador de clase dynamic le permite hacer justamente esto.

### Para crear una clase dinámica:

1. Seleccione Archivo > Nuevo, elija Archivo ActionScript y haga clic en Aceptar.
2. Seleccione Archivo > Guardar como y asigne al archivo el nombre **Person2.as**. Guarde el archivo en el disco duro.
3. Escriba el siguiente código en la ventana Script:

```
dynamic class Person2 {
  public var userName:String;
  public var age:Number;
}
```

   Este código ActionScript añade la palabra clave `dynamic` a la clase Person en el ejemplo anterior. Las instancias de la clase Person2 pueden añadir y acceder a las propiedades y métodos no definidos en esta clase.
4. Guarde los cambios en el archivo ActionScript.
5. Seleccione Archivo > Nuevo, elija Documento de Flash para crear un nuevo archivo FLA y luego haga clic en Aceptar.

6. Seleccione Archivo > Guardar como y asigne al nuevo archivo el nombre **person2_test.fla**. Guárdelo en el mismo directorio que Person2.as.
7. Añada el siguiente código para crear una nueva instancia de la clase Person2 (`firstPerson`) y asigne un valor a una propiedad denominada `hairColor` (que no existe en la clase Person2).

```
var firstPerson:Person2 = new Person2();
firstPerson.hairColor = "blue";
trace(firstPerson.hairColor); // blue (azul)
```

8. Guarde los cambios en el archivo person2_test.fla.
9. Seleccione Control > Probar película para probar el código.

   Dado que la clase personalizada de Flash es dinámica, puede añadir métodos y propiedades a la clase durante la ejecución (cuando se reproduce el archivo SWF). Al probar el código, el texto `blue` debe aparecer en el panel Salida.

Al desarrollar aplicaciones, no es conveniente que las clases sean dinámicas a no ser que sea necesario. Una de las razones para no usar clases dinámicas es que la comprobación de tipos en clases dinámicas es menos estricta que en clases no dinámicas porque los miembros a los que se accede dentro de la definición de clase y en instancias de la clase no se comparan con los definidos en el ámbito de la clase. Sin embargo, pueden verificarse los tipos return y parameter de las funciones de miembros de clase.

Las subclases de clases dinámicas son también dinámicas, con una excepción. Las subclases de la clase MovieClip no son dinámicas de manera predeterminada, aunque la clase MovieClip en sí misma sí lo es. Esta implementación le proporciona un mayor control sobre las subclases de la clase MovieClip, dado que puede optar por hacer que sus clases sean dinámicas o no:

```
class A extends MovieClip {}        // A no es dinámica
dynamic class B extends A {}        // B es dinámica
class C extends B {}                // C es dinámica
class D extends A {}                // D no es dinámica
dynamic class E extends MovieClip{} // E es dinámica
```

Para más información sobre subclases, consulte el Capítulo 7, “Herencia”, en la página 275.

## Utilización de encapsulado

En el diseño elegante orientado a objetos, los objetos son vistos como "cajas negras" que contienen o *encapsulan* funcionalidad. Un programador debe poder interactuar con un objeto conociendo sólo sus propiedades, métodos y eventos (su interfaz de programación), sin conocer los detalles de su implementación. Este enfoque permite a los programadores pensar con niveles superiores de abstracción y ofrece un marco organizativo para la creación de sistemas complejos.

El encapsulado es la razón por la cual ActionScript 2.0 incluye, por ejemplo, control de acceso para miembros, de manera que los detalles de la implementación puedan ser privados e invisibles en el código externo a un objeto. El código situado fuera del objeto se ve forzado a interactuar con la interfaz de programación de objetos en lugar de con los detalles de la implementación. Este enfoque aporta importantes ventajas; por ejemplo, permite al creador del objeto cambiar la implementación del objeto sin necesidad de realizar cambios en el código externo al objeto, siempre y cuando la interfaz de programación no cambie.

Un ejemplo de encapsulado en Flash sería la configuración de todos los miembros y variables de clase como privadas y forzar a aquellas personas que utilizaran sus clases a acceder a dichas variables empleando métodos getter y setter. Al encapsular de esta forma, tendrá la garantía de que, si en un momento posterior necesita cambiar la estructura de las variables, sólo tendrá que cambiar el comportamiento de las funciones getter y setter en lugar de forzar a todos los desarrolladores a cambiar la forma en que acceden a las variables de la clase.

En el siguiente código se muestra cómo podría modificar la clase Person de los ejemplos anteriores, establecer sus miembros de instancia como privados y definir métodos getter y setter para los miembros de instancia privados:

```
class Person {
  private var __userName:String;
  private var __age:Number;
  public function get userName():String {
    return this.__userName;
  }
  public function set userName(value:String):Void {
    this.__userName = value;
  }
  public function get age():Number {
    return this.__age;
  }
  public function set age(value:Number):Void {
    this.__age = value;
  }
}
```

## Utilización de la palabra clave this en clases

Utilice la palabra clave `this` como prefijo dentro de las clases para métodos y variables de miembros. Aunque no es necesaria, la palabra clave `this` hace que resulte sencillo saber que una propiedad o método pertenece a una clase cuando tiene un prefijo; sin la palabra clave, no es posible determinar si la propiedad o el método pertenece a la superclase.

También es posible utilizar un prefijo de nombre de clase en variables estáticas y métodos, e incluso en una clase. Esto ayuda a completar las referencias que se crean, lo que mejora la legibilidad del código. En función del entorno de codificación que utilice, la adición de prefijos también podría accionar sugerencias para el código.

**NOTA**

En opinión de algunos desarrolladores, no es necesario añadir estos prefijos. Macromedia recomienda añadir la palabra clave `this` como prefijo, ya que puede facilitar la legibilidad y ayudarle a escribir un código limpio al proporcionar contexto para los métodos y variables.

# Ejemplo: Escritura de clases personalizadas

Ahora que ya conoce los principios básicos de un archivo de clase y su contenido, es el momento de que conozca algunas de las directrices generales para la creación de un archivo de clase. El primer ejemplo de este capítulo muestra cómo escribir clases y empaquetarlas. El segundo ejemplo muestra cómo utilizar dichos archivos de clase con un archivo FLA.

**ATENCIÓN**

El código ActionScript de los archivos externos se compila en un archivo SWF cuando se publica, se exporta o se depura un archivo FLA. Por lo tanto, si realiza modificaciones en un archivo externo deberá guardarlo y volver a compilar todos los archivos FLA que lo utilizan.

Como se ha descrito en "Escritura de archivos de clases personalizadas" en la página 205, una clase consta de dos partes principales: la *declaración* y el *cuerpo*. La declaración de la clase consta como mínimo de la sentencia class seguida de un identificador para el nombre de clase y, a continuación, llaves de apertura y cierre ({}). Todo lo que se encuentra dentro de las llaves es el cuerpo de la clase, como se muestra en el siguiente ejemplo:

```
class className {
  // Cuerpo de la clase
}
```

Recuerde: sólo puede definir clases en archivos ActionScript. Por ejemplo, no puede definir una clase en un script de fotograma en un archivo FLA. Por consiguiente, debe crear un nuevo archivo para este ejemplo.

En su forma más básica, una declaración de clase consta de la palabra clave `class`, seguida del nombre de la clase (en este caso, Person) y después llaves de apertura y cierre (`{}`). Todo lo que está entre llaves se denomina el cuerpo de la clase y es donde se definen las propiedades y los métodos de la clase.

Al final de este ejemplo, el orden básico de los archivos de clase es el siguiente:

- Comentarios de documentación
- Declaración de clase
- Función constructora
- Cuerpo de la clase

En este capítulo no se escriben subclases. Para más información sobre herencia y subclases, consulte el Capítulo 7, “Herencia”, en la página 275

Este ejemplo contiene los siguientes temas:


- “Directrices generales para la creación de una clase” en la página 235
- “Creación y empaquetado de archivos de clases” en la página 236
- “Escritura de la función constructora” en la página 239
- “Adición de métodos y propiedades” en la página 241
- “Control del acceso de miembros en las clases” en la página 244
- “Documentación de las clases” en la página 246


Un archivo de ejemplo en su disco duro muestra la forma de crear un menú dinámico con datos XML y un archivo de clase personalizado. El ejemplo llama al constructor `XmlMenu()` de ActionScript y le pasa dos parámetros: la ruta al archivo de menú XML y una referencia a la línea de tiempo actual. El resto de la funcionalidad reside en un archivo de clase personalizado, XmlMenu.as.

El archivo de origen de muestra, xmlmenu.fla, se puede encontrar en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\XML_Menu.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/XML_Menu.

## Directrices generales para la creación de una clase

Los siguientes puntos son directrices que deberá seguir al escribir archivos de clases personalizadas. Le ayudarán a escribir clases correctas y bien formadas. Podrá practicar estas directrices en los ejemplos siguientes.

- En general, incluya una sola declaración por línea y no coloque la misma declaración o distintos tipos de declaración en una sola línea. Aplique formato a las declaraciones como se muestra en el siguiente ejemplo:

```
private var SKU:Number; // número de SKU (identificación) del producto
private var quantity:Number; // cantidad del producto
```

- Inicialice las variables locales al declararlas, a menos que el valor inicial se determine mediante un cálculo. Para más información sobre la inicialización de variables, consulte "Adición de métodos y propiedades" en la página 241.
- Declare las variables antes de utilizarlas por primera vez (incluidos los bucles). Por ejemplo, el siguiente código establece una declaración previa de la variable de repetidor de bucle (`i`) antes de utilizarla en el bucle`for`:

```
var my_array:Array = new Array("one", "two", "three");
var i:Number;
for (i = 0 ; i < my_array.length; i++) {
  trace(i + " = " + my_array[i]);
}
```

- Evite utilizar declaraciones locales que oculten declaraciones de nivel superior. Por ejemplo, no declare una variable dos veces, como se muestra en el siguiente ejemplo:

```
// código incorrecto
var counter:Number = 0;
function myMethod() {
  var counter:Number;
  for (counter = 0; counter <= 4; counter++) {
    // sentencias;
  }
}
```

  Este código declara la misma variable dentro de un bloque interno.

- No asigne muchas variables a un solo valor de una sentencia, porque es difícil de leer, como puede observarse en las siguientes muestras de código ActionScript:

```
// forma incorrecta
xPos = yPos = 15;
```

  o bien

```
// forma incorrecta
class User {
  private var m_username:String, m_password:String;
}
```

- Cree variables de instancia públicas o clases o variables de miembro estáticas públicas sólo cuando sea estrictamente necesario. Asegúrese de que estas variables sean públicas de manera explícita antes de crearlas de esta forma.
- Defina como privadas la mayoría de las variables de miembros a menos que haya un buen motivo para hacerlas públicas. Desde el punto de vista del diseño, es mucho mejor definir variables de miembros privadas y permitir el acceso a las variables únicamente a través de un grupo reducido de funciones getter y setter.

## Asignación de nombre a los archivos de clases

Los nombres de clase deben ser identificadores; es decir, que el primer carácter debe ser una letra, un carácter de subrayado (_) o un símbolo de dólar ($), y los caracteres siguientes deben ser una letra, un número, un carácter de subrayado o un símbolo de dólar. Se recomienda utilizar solamente letras en los nombres de clases.

El nombre de la clase debe coincidir exactamente con el nombre del archivo ActionScript que la contiene, incluidas mayúsculas y minúsculas. En el siguiente ejemplo, si crea una clase llamada Rock, el archivo ActionScript que contiene la definición de clase debe llamarse Rock.as:

```
// En el archivo Rock.as
class Rock {
  // Cuerpo de la clase Rock
}
```

En la siguiente sección asignará nombre y creará una definición de clase. Consulte la sección "Creación y empaquetado de archivos de clases" en la página 236 para crear, asignar nombre y empaquetar archivos de clase. Para más información sobre la asignación de nombre a archivos de clases, consulte "Asignación de nombre a clases y objetos" en la página 784.

# Creación y empaquetado de archivos de clases

En esta sección, crearemos, asignaremos nombre y empaquetaremos los archivos de clases para este ejemplo ("Ejemplo: Escritura de clases personalizadas" en la página 233). En secciones siguientes muestran cómo escribir archivos de clases completos (aunque sencillos). Para más información sobre paquetes, consulte "Paquetes" en la página 198, "Comparación de clases y paquetes" en la página 199 y "Utilización de paquetes" en la página 199.

Al crear un archivo de clase, decida dónde desea guardar el archivo. En los siguientes pasos, guardará el archivo de clase y el archivo FLA de la aplicación que utiliza el archivo de clase en el mismo directorio para mayor simplicidad. No obstante, si desea comprobar la sintaxis, también tendrá que indicar a Flash cómo encontrar el archivo. Generalmente, al crear una aplicación, tendría que añadir el directorio en el que están guardados la aplicación y los archivos de clase en la ruta de clases de Flash. Para más información sobre rutas de clases, consulte "Configuración y modificación de la ruta de clases" en la página 211.

Los archivos de clases también se denominan archivos ActionScript (AS). Los archivos AS se crean en la herramienta de edición de Flash o mediante un editor externo. Hay varios editores externos, como Macromedia Dreamweaver y Macromedia Flex Builder, que permiten crear archivos AS.

| NOTA | El nombre de la clase (ClassA) debe coincidir exactamente con el nombre del archivo AS que la contiene (ClassA.as). Esto es sumamente importante, puesto que si estos dos nombres no coinciden, incluidas mayúsculas y minúsculas, la clase no se compilará. |
|---|---|

### Para crear un archivo de clase y una declaración de clase:

1. Seleccione Archivo > Nuevo, elija Documento de Flash para crear un documento FLA y luego haga clic en Aceptar.
2. Seleccione Archivo > Guardar como, asigne al nuevo archivo el nombre **package_test.fla** y guarde el documento de Flash en el directorio actual.

   En un paso posterior deberá añadir contenido a este documento de Flash.
3. Seleccione Archivo > Nuevo, elija Archivo ActionScript y haga clic en Aceptar.
4. Seleccione Archivo > Guardar como y cree un subdirectorio nuevo llamado **com**.
   A continuación, realice lo siguiente:
   a. En el subdirectorio com, cree un nuevo subdirectorio llamado **macromedia**.
   b. En el subdirectorio macromedia, cree un nuevo subdirectorio llamado **utils**.
   c. Guarde el documento ActionScript actual en el directorio utils y asigne al archivo el nombre **ClassA.as**.
5. Escriba el siguiente código en la ventana Script:

   ```
   class com.macromedia.utils.ClassA {
   }
   ```

   El código anterior crea una nueva clase denominada ClassA en el paquete com.macromedia.utils.
6. Guarde el documento ActionScript ClassA.as.
7. Seleccione Archivo > Nuevo, elija Archivo ActionScript y haga clic en Aceptar.

8. Seleccione Archivo > Guardar como, asigne al nuevo archivo el nombre **ClassB.as** y guárdelo en el mismo directorio que ClassA.as, creado en el paso anterior.
9. Escriba el siguiente código en la ventana Script:

```
class com.macromedia.utils.ClassB {
}
```

   El código anterior crea una nueva clase denominada ClassB en el paquete com.macromedia.utils.
10. Guarde los cambios en ambos archivos de clases, ClassA.as y ClassB.as.

Los archivos de clases que se utilizan en un archivo FLA se importan a un archivo SWF al compilarlos. El código que escriba en un archivo de clase deberá seguir una determinada metodología y un orden concreto, tal y como se describe en las siguientes secciones.

Si va a crear múltiples clases personalizadas, use paquetes para organizar los archivos de clases. Un paquete es un directorio que contiene uno o más archivos de clases y que reside en un directorio de rutas de clases designado. El nombre de la clase debe estar completo en el archivo en el que está declarada; es decir, debe reflejar el directorio (paquete) en el que está almacenada. Para más información sobre rutas de clases, consulte "Configuración y modificación de la ruta de clases" en la página 211.

Por ejemplo, una clase denominada `com.macromedia.docs.YourClass` se almacena en el directorio `com/macromedia/docs`. La declaración de clase del archivo YourClass.as tiene el siguiente aspecto:

```
class com.macromedia.docs.YourClass {
  // la clase
}
```

**NOTA** En la siguiente sección, "Ejemplo: Escritura de clases personalizadas" en la página 233, escribirá la declaración de clase que refleja el directorio del paquete.

Por este motivo, es aconsejable planificar una estructura de paquete antes de empezar a crear clases. De otro modo, si decide mover archivos de clases después de crearlos, deberá modificar las sentencias de declaración de clases para reflejar su nueva ubicación.

### Para empaquetar los archivos de clases:

1. Decida el nombre que desea utilizar para el paquete.

   Los nombres de paquetes deben ser intuitivos y fácilmente identificables por parte de otros desarrolladores. Recuerde que el nombre del paquete también coincidirá con una estructura de directorios específica. Por ejemplo, las clases existentes en el paquete `com.macromedia.utils` deben colocarse en una carpeta com/macromedia/utils de la unidad de disco duro.

2. Cree la estructura de directorios que sea necesaria, una vez elegido el nombre del paquete.

   Por ejemplo, si el paquete se llama `com.macromedia.utils`, deberá crear la estructura de directorios com/macromedia/utils y colocar las clases en la carpeta utils.

3. Utilice el prefijo com.macromedia.utils para cualquier clase que cree en este paquete.

   Por ejemplo, si el nombre de la clase fuera ClassA, el nombre de clase completo tendría que ser `com.macromedia.utils.ClassA` dentro del archivo de clase `com/macromedia/utils/ClassA.as`.

4. Si cambia la estructura del paquete en un momento posterior, recuerde que no sólo deberá modificar la estructura de directorios, sino también el nombre del paquete dentro de cada archivo de clase, además de todas las sentencias de importación o referencias a una clase contenida en dicho paquete.

Para continuar escribiendo archivos de clases, consulte "Escritura de la función constructora" en la página 239.

## Escritura de la función constructora

Ya hemos visto cómo se escribe la declaración de clase en "Creación y empaquetado de archivos de clases" en la página 236. En esta parte del capítulo, escribiremos lo que se conoce como la *función constructora* del archivo de clase.

| NOTA | En posteriores secciones, aprenderá a escribir los comentarios, las sentencias y declaraciones. |
|---|---|

Los constructores son funciones que se utilizan para inicializar (*definir*) las propiedades y métodos de una clase. Por definición, los constructores son funciones incluidas en definiciones de clases que tienen el mismo nombre que la clase. Por ejemplo, el código siguiente define una clase Person e implementa una función constructora. En OOP, la función constructora inicializa cada instancia nueva de una clase.

El constructor de una clase es una función especial a la que se llama automáticamente cuando se crea una instancia de una clase mediante el operador `new`. La función constructora tiene el mismo nombre que la clase que la contiene. Por ejemplo, la clase Person que se creó contenía la función constructora siguiente:

```
//Función constructora de la clase Person
public function Person (uname:String, age:Number) {
  this.__name = uname;
  this.__age = age;
}
```

Tenga en cuenta los siguientes aspectos al escribir funciones constructoras:

- Si no se declara de forma explícita ninguna función constructora (es decir, si no se crea una función cuyo nombre coincida con el nombre de la clase), el compilador crea automáticamente una función constructora vacía.
- Una clase sólo puede contener una función constructora; ActionScript 2.0 no admite funciones constructoras sobrecargadas.
- Una función constructora no debe tener tipo de retorno.

El término *constructor* también se emplea habitualmente para crear un objeto (crear instancias del mismo) basado en una clase determinada. Las sentencias siguientes son llamadas a las funciones constructoras de la clase del más alto nivel Array y de la clase personalizada Person:

```
var day_array:Array = new Array("Sun", "Mon", "Tue", "Wed", "Thu", "Fri",
  "Sat");
var somePerson:Person = new Person("Tom", 30);
```

A continuación, añadirá una función especial denominada función constructora.

NOTA

El siguiente ejercicio es parte de "Ejemplo: Escritura de clases personalizadas" en la página 233. Si no desea seguir todos los pasos del ejercicio, puede descargar los archivos de clase en www.helpexamples.com/flash/learnas/classes/.

### Para añadir las funciones constructoras a los archivos de clases:

1. Abra la clase ClassA.as en la herramienta de edición de Flash.
2. Modifique el archivo de clase existente para que coincida con el siguiente código (los cambios para realizar aparecen en negrita):

   ```
   class com.macromedia.utils.ClassA {
     function ClassA() {
       trace("ClassA constructor");
     }
   }
   ```

   El código anterior define un método constructor para la clase ClassA. Este constructor traza una cadena sencilla en el panel Salida que le permitirá saber cuándo se ha creado una nueva instancia de la clase.
3. Abra la clase ClassB.as en la herramienta de edición de Flash.
4. Modifique el archivo de clase existente para que coincida con el siguiente código (los cambios para realizar aparecen en negrita):

   ```
   class com.macromedia.utils.ClassB {
     function ClassB() {
       trace("ClassB constructor");
     }
   }
   ```

5. Guarde ambos archivos ActionScript antes de continuar.

Para continuar escribiendo el archivo de clase, consulte "Adición de métodos y propiedades" en la página 241.

## Adición de métodos y propiedades

Para crear las propiedades de las clases ClassA y ClassB, utilice la palabra clave `var` para definir variables.

| NOTA | Los tres ejercicios siguientes forman parte de "Ejemplo: Escritura de clases personalizadas" en la página 233. Si no desea seguir todos los pasos del ejercicio, puede descargar los archivos de clase en www.helpexamples.com/flash/learnas/classes/. |
|---|---|

### Para añadir propiedades a las clases ClassA y ClassB:

1. Abra ClassA.as y ClassB.as en la herramienta de edición de Flash.
2. Modifique el archivo ActionScript ClassA.as para que coincida con el siguiente código (los cambios para realizar aparecen en negrita):

```
class com.macromedia.utils.ClassA {
  static var _className:String;
  function ClassA() {
    trace("ClassA constructor");
  }
}
```

   El bloque de código anterior añade una única variable estática, `_className`, que contiene el nombre de la clase actual.
3. Modifique la clase ClassB y añada la variable estática de manera que sea igual al código anterior.
4. Guarde ambos archivos ActionScript antes de continuar.

| SUGERENCIA | Por convención, las propiedades de clases se definen en la parte superior del cuerpo de la clase. Su definición en la parte superior hace que el código resulte más fácil de entender, aunque no es imprescindible. |
|---|---|

La sintaxis de signo de dos puntos posterior (por ejemplo, `var username:String` y `var age:Number`) se utiliza en las declaraciones de variables. Éste es un ejemplo de strict data typing. Cuando escribe una variable empleando el formato `var variableName:variableType`, el compilador de ActionScript se asegura de que todos los valores asignados a dicha variable coincidan con el tipo especificado. Si no se utiliza el tipo de datos correcto en el archivo FLA que importa esta clase, el compilador devolverá un error. Para más información sobre "strict data typing", consulte "Asignación de tipos de datos y "strict data typing"" en la página 337.

Los miembros de una clase constan de propiedades (declaraciones de variables) y métodos (definiciones de funciones). Debe declarar y definir las propiedades y los métodos dentro del cuerpo de la clase (entre llaves [{}]); de lo contrario, se produce un error durante la compilación. Para obtener información sobre miembros, consulte "Métodos y propiedades (miembros) públicos, privados y estáticos" en la página 217.

### Para añadir métodos a las clases ClassA y ClassB:

1. Abra ClassA.as y ClassB.as en la herramienta de edición de Flash.
2. Modifique el archivo de clase ClassA para que coincida con el siguiente código (los cambios para realizar aparecen en negrita):

```
class com.macromedia.utils.ClassA {
  static var _className:String;

  function ClassA() {
    trace("ClassA constructor");
  }
  function doSomething():Void {
    trace("ClassA - doSomething()");
  }
}
```

   El bloque de código en negrita crea un nuevo método en la clase que traza una cadena en el panel Salida.
3. En ClassA.as, seleccione Herramientas > Revisar sintaxis para revisar la sintaxis del archivo ActionScript.

   Si se indica algún error en el panel Salida, compare el código ActionScript de su script con el código completo escrito en el paso anterior. Si no puede resolver los errores del código, copie y pegue el código completo en la ventana Script antes de continuar.

4. Revise la sintaxis de ClassB.as como lo hizo con ClassA.as.

   Si aparecen errores en el panel Salida, copie y pegue el código completo en la ventana Script antes de continuar:

```
class com.macromedia.utils.ClassB {
  static var _className:String;

  function ClassB() {
    trace("ClassB constructor");
  }
  function doSomething():Void {
    trace("ClassB - doSomething()");
  }
}
```

5. Guarde ambos archivos ActionScript antes de continuar.

Puede inicializar propiedades en línea (es decir, al declararlas), con valores predeterminados, como se indica a continuación:

```
class Person {
  var age:Number = 50;
  var username:String = "John Doe";
}
```

Al inicializar propiedades en línea, la expresión situada en el lado derecho de una asignación debe ser una constante en tiempo de compilación. Es decir, la expresión no puede hacer referencia a nada que se haya establecido o definido durante la ejecución. Las constantes en tiempo de compilación incluyen literales de cadena, números, valores booleanos, null y undefined, así como funciones constructoras para las siguientes clases del más alto nivel: Array, Boolean, Number, Object y String.

### Para inicializar propiedades en línea:

1. Abra ClassA.as y ClassB.as en la herramienta de edición de Flash.
2. Modifique el archivo de clase ClassA para que coincida con el siguiente código ActionScript (los cambios para realizar aparecen en negrita):

```
class com.macromedia.utils.ClassA {
  static var _className:String = "ClassA";

  function ClassA() {
    trace("ClassA constructor");
  }
  function doSomething():Void {
    trace("ClassA - doSomething()");
  }
}
```

   La única diferencia entre el archivo de clase existente y el bloque de código anterior es que ahora hay un valor definido para la variable estática `_className`, “ClassA”.

3. Modifique el archivo de clase ClassB y añada la propiedad en línea, cambiando el valor a "ClassB".
4. Guarde ambos archivos ActionScript antes de continuar.

Esta regla sólo se aplica a las variables de instancia (variables que se copian en cada instancia de una clase), no a variables de clase (variables que pertenecen a la clase).

| NOTA | Al inicializar matrices en línea, sólo se crea una matriz para todas las instancias de la clase. |
|---|---|

Para continuar escribiendo el archivo de clase, consulte "Control del acceso de miembros en las clases" en la página 244.

## Control del acceso de miembros en las clases

De forma predeterminada, una clase puede acceder a cualquier propiedad o método de otra clase: todos los miembros de una clase son públicos. No obstante, en algunos casos quizá desee proteger datos o métodos de una clase para que otras clases no puedan acceder a ellos. Deberá hacer que estos miembros sean privados (que sólo estén disponibles para la clase que los declara o los define).

Especifique miembros públicos o privados con los atributos de miembro public o private. Por ejemplo, el siguiente código declara una variable privada (una propiedad) y un método privado (una función). La clase siguiente (LoginClass) define una propiedad privada denominada `userName` y un método privado llamado `getUserName()`:

```
class LoginClass {
  private var userName:String;
  private function getUserName():String {
    return this.userName;
  }
  // Constructor:
  public function LoginClass(user:String) {
    this.userName = user;
  }
}
```

Los miembros privados (propiedades y métodos) son accesibles sólo para la clase que define dichos miembros y las subclases de esa clase original. Las instancias de la clase original o las de las subclases de dicha clase no pueden acceder de forma privada a propiedades y métodos declarados; es decir, los miembros privados son accesibles sólo dentro de las definiciones de clase, no en el nivel de la instancia. En el siguiente ejemplo, cambiará el acceso de los miembros en los archivos de clases.

| NOTA | Este ejercicio forma parte de "Ejemplo: Escritura de clases personalizadas" en la página 233. Si no desea seguir todos los pasos del ejercicio, puede descargar los archivos de clase en www.helpexamples.com/flash/learnas/classes/. |
|---|---|

### Para controlar el acceso de los miembros:

1. Abra ClassA.as y ClassB.as en la herramienta de edición de Flash.
2. Modifique el archivo ActionScript ClassA.as para que su contenido coincida con el siguiente código ActionScript (los cambios para realizar aparecen en negrita):

```
class com.macromedia.utils.ClassA {
  private static var _className:String = "ClassA";

  public function ClassA() {
    trace("ClassA constructor");
  }
  public function doSomething():Void {
    trace("ClassA - doSomething()");
  }
}
```

   El código anterior establece ambos métodos (el constructor de ClassA y el método `doSomething()`) como públicos, lo que significa que los scripts externos pueden acceder a ellos. La variable estática `_className` se establece como privada, lo que significa que sólo puede accederse a la variable desde dentro de la clase y no desde scripts externos.
3. Modifique el archivo ActionScript ClassB.as y añada el mismo acceso de métodos y propiedades que para la clase ClassA.
4. Guarde ambos archivos ActionScript antes de continuar.

Una instancia de ClassA o ClassB no puede acceder a los miembros privados. Por ejemplo, el código siguiente, añadido al fotograma 1 de la línea de tiempo de un archivo FLA, daría como resultado un error de compilador que indicaría que el método es privado y es imposible acceder a él.

```
import com.macromedia.utils.ClassA;
var a:ClassA = new ClassA();
trace(a._className); // Error. El miembro es privado y no permite el acceso.
```

El control de acceso de los miembros es una función sólo de compilación; durante la ejecución, Flash Player no distingue entre los miembros privados y los públicos.

Para continuar escribiendo el archivo de clase, consulte "Documentación de las clases" en la página 246.

# Documentación de las clases

La utilización de comentarios en las clases e interfaces es una forma importante de documentarlas para otros usuarios. Por ejemplo, puede que desee distribuir los archivos de clases entre la comunidad de usuarios de Flash, o que esté trabajando con un equipo de diseñadores o desarrolladores que utilizarán los archivos de clases creados por usted en su trabajo o como parte de un proyecto en el que usted participa. La documentación ayuda a otros usuarios a entender el objetivo y los orígenes de la clase.

En un archivo de clase o interfaz típico hay dos tipos de comentarios: *comentarios de documentación* y *comentarios de implementación*. Los comentarios de documentación se utilizan para describir las especificaciones del código, no la implementación. Los comentarios de implementación se utilizan para convertir en comentario parte del código o para comentar la implementación de determinadas secciones del código. Los dos tipos de comentarios utilizan delimitadores ligeramente distintos. Los comentarios de documentación se delimitan con `/**` y `*/`, y los comentarios de implementación se delimitan con `/*` y `*/`.

| NOTA | Los comentarios de documentación no son una construcción del lenguaje en ActionScript 2.0. Sin embargo, constituyen una forma frecuente de estructurar los comentarios de un archivo de clase que puede utilizar en los archivos AS. |
|---|---|

Los comentarios de documentación se utilizan para describir interfaces, clases, métodos y constructores. Incluya un solo comentario de documentación por cada clase, interfaz o miembro y colóquelo inmediatamente antes de la declaración.

Si debe documentar información adicional que no cabe en los comentarios de documentación, utilice los comentarios de implementación (con el formato de comentarios en bloque o comentarios de una sola línea, como se describen en "Comentarios" en la página 95). Si añade comentarios de implementación, éstos deberán ir inmediatamente después de la declaración.

| NOTA | No incluya comentarios que no estén directamente relacionados con la clase que se está leyendo. Por ejemplo, no incluya comentarios que describan el paquete correspondiente. |
|---|---|

| NOTA | El siguiente ejercicio es parte de "Ejemplo: Escritura de clases personalizadas" en la página 233. Si no desea seguir todos los pasos del ejercicio, puede descargar los archivos de clase en www.helpexamples.com/flash/learnas/classes/. |
|---|---|

### Para documentar los archivos de clases:

1. Abra ClassA.as y ClassB.as en la herramienta de edición de Flash.
2. Modifique el archivo de clase ClassA y añada el nuevo código en la parte superior del archivo de clase (los cambios para realizar aparecen en negrita):

```
/**
  ClassA class
  version 1,1
  6/21/2005
  copyright Macromedia, Inc.
 */
class com.macromedia.utils.ClassA {
  private static var _className:String = "ClassA";

  public function ClassA() {
    trace("ClassA constructor");
  }
  public function doSomething():Void {
    trace("ClassA - doSomething()");
  }
}
```

   El código anterior añadió un comentario al principio del archivo de clase. Siempre es aconsejable añadir comentarios al código ActionScript y a los archivos de Flash, ya que éstos le permiten añadir información útil, como el autor de la clase, la fecha en que fue modificada por última vez, la información de copyright o cualquier posible error o problema que pueda existir en el archivo.
3. Añada un comentario similar en la parte superior del archivo ActionScript ClassB.as, cambiando el nombre de la clase y otros datos que considere oportunos.
4. Guarde ambos archivos ActionScript antes de continuar.

También podría añadir comentarios de bloque, de una sola línea o finales en el código de la clase. Para obtener información sobre la escritura de comentarios adecuados en el código, consulte “Escritura de comentarios adecuados” en la página 788. Para obtener información general sobre los comentarios, consulte “Comentarios de una sola línea” en la página 96, “Comentarios de varias líneas” en la página 97 y “Comentarios finales” en la página 98.

Para aprender a utilizar estos archivos de clases personalizadas en un archivo SWF, consulte “Ejemplo: Utilización de archivos de clases personalizadas en Flash” en la página 248.

# Ejemplo: Utilización de archivos de clases personalizadas en Flash

En este ejemplo se utilizan archivos de clases escritos en la sección titulada "Ejemplo: Escritura de clases personalizadas" en la página 233 o puede descargarlos en www.helpexamples.com/flash/learnas/classes/. Si ha realizado la sección "Ejemplo: Escritura de clases personalizadas" en la página 233, busque los archivos ClassA.as y ClassB.as en el disco duro.

Dado que el nombre del paquete del archivo de clase ClassA es `com.macromedia.utils.ClassA`, deberá asegurarse de que guarda los archivos de clases con la estructura de directorios adecuada. Cree una subcarpeta denominada com en el directorio actual. Dentro de la carpeta com, añada una nueva carpeta denominada macromedia. Añada un tercer y último directorio llamado utils dentro de la carpeta macromedia. Guarde ambos archivos de clases, ClassA.as y ClassB.as, dentro de la carpeta utils. Ya está preparado para continuar con este ejemplo.

Puede utilizar las clases personalizadas escritas en "Ejemplo: Escritura de clases personalizadas" en la página 233 con un archivo FLA. En este ejemplo, las clases personalizadas se utilizan para crear una pequeña aplicación en Flash. Las clases se compilan en el archivo SWF al publicar el documento, tras lo cual todo funciona correctamente. En los siguientes ejercicios, aprenderá el funcionamiento de las rutas de clases, la forma de utilizar los archivos de clase en la aplicación, así como el modo de importar clases y paquetes.

Para continuar con este ejemplo, prosiga con "Importación de clases y paquetes" en la página 248.

## Importación de clases y paquetes

Para hacer referencia a una clase de otro script, el nombre del paquete de la clase debe especificarse como prefijo del nombre de clase. La combinación del nombre de una clase y la ruta de paquete es el nombre de clase completo de la clase. Si una clase reside en un directorio de rutas de clases de nivel superior y no en un subdirectorio del directorio de rutas de clases, el nombre de clase es también el nombre de clase completo.

Para especificar las rutas de paquetes, utilice la notación con puntos (`.`) para separar nombres de directorios de paquetes. Las rutas de paquetes tienen una estructura jerárquica, es decir, cada punto representa un directorio anidado. Por ejemplo, supongamos que crea una clase denominada ClassName que reside en un paquete `com/macromedia/docs/learnAs2` en la ruta de clases. Para crear una instancia de dicha clase, puede especificar el nombre de clase completo.

También puede utilizar el nombre de clase completo para introducir las variables, como se muestra en el siguiente ejemplo:

```
var myInstance:com.macromedia.docs.learnAs2.ClassName = new
  com.macromedia.docs.learnAs2.ClassName();
```

Puede utilizar la sentencia `import` para importar paquetes a un script, lo que le permitirá utilizar un nombre abreviado de clase en lugar de su nombre completo. También puede utilizar el carácter comodín (*) para importar todas las clases de un paquete. Si utiliza el carácter comodín, no tendrá que utilizar el nombre de clase completo cada vez que utilice la clase.

Supongamos, por ejemplo, que ha importado clase anterior en un script utilizando la sentencia `import`, como se muestra en el siguiente ejemplo:

```
import com.macromedia.docs.learnAs2.util.UserClass;
```

Posteriormente, en el mismo script, podría hacer referencia a dicha clase mediante su nombre abreviado, como se muestra en el siguiente ejemplo:

```
var myUser:UserClass = new UserClass();
```

También puede utilizar el carácter comodín (*) para importar todas las clases de un paquete determinado. Supongamos, por ejemplo, que tiene un paquete llamado `com.macromedia.utils` que contiene dos archivos de clases ActionScript, ClassA.as y ClassB.as. En otro script, podría importar ambas clases de dicho paquete mediante el carácter comodín, como se muestra en el siguiente código:

```
import com.macromedia.utils.*;
```

En el siguiente ejemplo se muestra que posteriormente puede hacer referencia a cualquiera de las clases directamente en el mismo script:

```
var myA:ClassA = new ClassA();
var myB:ClassB = new ClassB();
```

La sentencia `import` sólo se aplica al script actual (fotograma u objeto) en el que se llama. Si una clase importada no se utiliza en un script, no se incluirá en el código de bytes del archivo SWF resultante ni estará disponible para los archivos SWF que pueda cargar el archivo FLA que contiene la sentencia `import`.

**NOTA**

El siguiente ejercicio es parte de "Ejemplo: Utilización de archivos de clases personalizadas en Flash" en la página 248 que es la continuación de los ejemplos de "Ejemplo: Escritura de clases personalizadas". Si necesita los archivos de clase ClassA y ClassB, puede descargarlos en www.helpexamples.com/flash/learnas/classes/.

**Para importar una clase o paquete:**

1. Abra el archivo denominado package_test.fla.
2. Escriba el siguiente código en la ventana Script:

```
import com.macromedia.utils.*;
var a = new ClassA(); // constructor de ClassA
var b = new ClassB(); // constructor de ClassB
```

   El bloque de código anterior comienza por importar cada una de las clases del paquete `com.macromedia.utils` utilizando el carácter comodín (*). A continuación, debe crear una nueva instancia de la clase ClassA, lo que provocará que el método constructor trace un mensaje en el panel Salida. También se crea una instancia de la clase ClassB, que envía mensajes de depuración al panel Salida.

3. Guarde los cambios en el documento de Flash antes de continuar.

Para continuar utilizando estos archivos de clases en un archivo de Flash, consulte "Creación de instancias de clases en un ejemplo" en la página 250.

## Creación de instancias de clases en un ejemplo

Las instancias son objetos que contienen todas las propiedades y métodos de una clase concreta. Por ejemplo, las matrices son instancias de la clase Array, de modo que puede utilizar cualquiera de los métodos o propiedades de la clase Array con cualquier instancia de matriz. O bien puede crear su propia clase, como, por ejemplo, UserSettings, y luego crear una instancia de la clase UserSettings.

Continuando con el ejemplo iniciado en "Ejemplo: Utilización de archivos de clases personalizadas en Flash" en la página 248, ha modificado el archivo FLA para importar las clases escritas de modo que no tenga que hacer siempre referencia a ellas por sus nombres completos.

El siguiente paso de este ejemplo ("Ejemplo: Utilización de archivos de clases personalizadas en Flash" en la página 248) consiste en crear una instancia de las clases ClassA y ClassB en un script, por ejemplo, un script de fotograma en un documento de Flash package_test.fla, y asignarlo a una variable. Para crear una instancia de una clase personalizada, utilice el operador `new`, tal y como haría al crear una instancia de una clase del más alto nivel de ActionScript (como la clase Date o Array). Puede hacer referencia a la clase empleando su nombre de clase completo o importar la clase (como se muestra en "Importación de clases y paquetes" en la página 248).

**NOTA**

El siguiente ejercicio es parte de **"Ejemplo: Utilización de archivos de clases personalizadas en Flash" en la página 248** que es la continuación de los ejemplos de **"Ejemplo: Escritura de clases personalizadas"**.

### Para crear una nueva instancia de las clases ClassA y ClassB:

1. Abra el archivo denominado **package_test.fla**.
2. Escriba el siguiente código en negrita en la ventana Script:

```
import com.macromedia.utils.*;
var a:ClassA = new ClassA(); // constructor de ClassA
a.doSomething(); // llamar al método doSomething() de ClassA
var b:ClassB = new ClassB(); // constructor de ClassB
b.doSomething(); // llamar al método doSomething() de ClassB
```

   Introducir los datos de los objetos en este ejemplo de código permite al compilador asegurarse de que no se intenta acceder a las propiedades o los métodos que no están definidos en la clase personalizada. Para más información sobre "strict data typing", consulte "Asignación de tipos de datos y "strict data typing"" en la página 337. La excepción a la escritura de datos de los objetos es si la clase se declara como dinámica con la palabra clave `dynamic`. Consulte "Creación de clases dinámicas" en la página 229.
3. Guarde los cambios en el archivo FLA antes de continuar.

Ahora debería tener una idea general de cómo crear y utilizar las clases en los documentos de Flash. Recuerde que también puede crear instancias de clases de nivel superior o incorporadas en ActionScript (consulte "Utilización de clases incorporadas" en la página 269).

Para continuar utilizando estos archivos de clases en un archivo de Flash, consulte "Asignación de una clase a símbolos en Flash" en la página 251.

# Asignación de una clase a símbolos en Flash

También puede asignar una clase a símbolos que puede utilizar en un archivo de Flash, como un objeto de clip de película del escenario.

### Para asignar una clase a un símbolo de clip de película:

1. Seleccione Archivo > Nuevo, elija Archivo ActionScript y haga clic en Aceptar.
2. Seleccione Archivo > Guardar como, asigne al archivo el nombre **Animal.as** y guárdelo en el disco duro.
3. Escriba el siguiente código en la ventana Script:

```
class Animal {
  public function Animal() {
    trace("Animal::constructor");
  }
}
```

   Este código ActionScript crea una nueva clase denominada Animal que tiene un método constructor que traza una cadena en el panel Salida.

4. Guarde los cambios en el archivo ActionScript.
5. Seleccione Archivo > Nuevo, elija Documento de Flash para crear un nuevo archivo FLA y luego haga clic en Aceptar.
6. Seleccione Archivo > Guardar como, asigne al archivo el nombre **animal_test.fla** y guárdelo en la misma carpeta que el archivo Animal.as creado en el paso 2.
7. Seleccione Insertar > Nuevo símbolo para abrir el cuadro de diálogo Crear un nuevo símbolo.
8. Especifique un nombre de símbolo de **animal** y seleccione la opción Clip de película.
9. Haga clic en el botón Avanzado en la esquina inferior derecha del cuadro de diálogo Crear un nuevo símbolo para activar más opciones.

   El botón Avanzado está disponible cuando se encuentra en el modo básico del cuadro de diálogo Crear un nuevo símbolo.
10. Haga clic en la casilla de verificación Exportar para ActionScript en la sección Vinculación.

    La activación de esta opción le permite asociar dinámicamente instancias de este símbolo a los documentos de Flash durante la ejecución.
11. Introduzca el valor del identificador **animal_id** y establezca Clase de AS 2.0 con el valor **Animal** (para que coincida con el nombre de clase especificado en el paso 3).
12. Active la casilla de verificación Exportar en primer fotograma y haga clic en Aceptar para aplicar los cambios y cerrar el cuadro de diálogo.
13. Guarde el documento de Flash y seleccione Control > Probar película.

    El panel Salida muestra el texto de la función constructora de la clase Animal.

| NOTA | Si necesita modificar las propiedades de Vinculación de Clip de película, haga clic con el botón derecho del ratón en el símbolo en la biblioteca del documento y seleccione Propiedades o Vinculación del menú contextual. |
|---|---|

# Compilación y exportación de clases

De forma predeterminada, las clases utilizadas por un archivo SWF se empaquetan y exportan en el primer fotograma del archivo SWF. También puede especificar un fotograma diferente en el que se empaquetan y exportan las clases. Esto es útil, por ejemplo, si un archivo SWF utiliza muchas clases que necesitan bastante tiempo para descargarse (por ejemplo, componentes). Si las clases se exportan en el primer fotograma, el usuario deberá esperar hasta que todo el código de clase se haya descargado para que aparezca el fotograma. Al especificar un fotograma posterior en la línea de tiempo, se puede visualizar una breve animación de carga en los primeros fotogramas de la línea de tiempo mientras se descarga el código de clase del fotograma posterior.

### Para especificar el fotograma de exportación para clases para un documento de Flash:

1. Seleccione Archivo > Nuevo y seleccione Documento de Flash. Guarde el nuevo documento como **exportClasses.fla**.
2. Cambie el nombre de la capa predeterminada a **content**, arrastre al escenario un componente ProgressBar del panel Componentes y asígnele el nombre de instancia **my_pb**.
3. Cree una nueva capa, arrástrela a la capa "content" y cámbiele el nombre por **actions**.
4. Añada el siguiente código ActionScript al fotograma 1 de la capa acciones de la línea de tiempo:

```
my_pb.indeterminate = true;
```

5. Cree un nuevo fotograma clave en el fotograma 2 de la capa "actions" y añada el siguiente código ActionScript:

```
var classesFrame:Number = 10;
if (_framesloaded < classesFrame) {
  trace(this.getBytesLoaded() + " of " + this.getBytesTotal() + " bytes
  loaded");
  gotoAndPlay(1);
} else {
  gotoAndStop(classesFrame);
}
```

6. Cree un nuevo fotograma clave en el fotograma 10 de la capa "actions" y añada el siguiente código ActionScript:

```
stop();
```

7. Cree un nuevo fotograma clave en el fotograma 10 de la capa "content" y arrastre varios componentes al escenario.
8. Haga clic con el botón derecho del ratón en cada componente (salvo en ProgressBar) del panel Biblioteca y seleccione Vinculación en el menú contextual para abrir el cuadro de diálogo Propiedades de vinculación.

9. En el cuadro de diálogo Propiedades de vinculación, asegúrese de que está seleccionada la opción Exportar para ActionScript, anule la selección de Exportar en primer fotograma y haga clic en Aceptar.
10. Seleccione Archivo > Configuración de publicación.
11. En el cuadro de diálogo Configuración de publicación, seleccione la ficha Flash.
12. Haga clic en el botón Configuración situado junto al menú emergente de la versión ActionScript para abrir el cuadro de diálogo Configuración de ActionScript.
13. En el cuadro de texto Fotograma de exportación para clases, introduzca el número del fotograma en el que desee exportar el código de clase (fotograma 10).

    Si el fotograma especificado no existe en la línea de tiempo, obtendrá un mensaje de error cuando publique el archivo SWF.
14. Haga clic en Aceptar para cerrar el cuadro de diálogo Configuración de ActionScript y, a continuación, haga clic en Aceptar para cerrar el cuadro de diálogo Configuración de publicación.
15. Seleccione Control > Probar película para probar el documento de Flash. Si los componentes se cargan demasiado deprisa, seleccione Ver > Simular descarga del archivo SWF. Flash simula la descarga del documento de Flash a una velocidad menor, lo que le permite ver la animación de los componentes de la barra de progreso mientras se descargan los archivos de clase.

Para más información sobre archivos ASO, consulte "Utilización de archivos ASO" en la página 254.

## Utilización de archivos ASO

Durante la compilación, Flash crea en ocasiones archivos con la extensión .aso en el subdirectorio /aso del directorio de ruta de clases global predeterminado (consulte "Configuración y modificación de la ruta de clases" en la página 211). La extensión .aso son las siglas de *ActionScript object* (ASO). Por cada archivo ActionScript 2.0 que se importa implícita o explícitamente y se compila correctamente, Flash genera un archivo ASO. El archivo contiene el código de bytes generado por el archivo ActionScript (AS) asociado. Por consiguiente, estos archivos contienen la forma compilada (el *código de bytes*) de un archivo de clase.

Flash sólo necesita generar un archivo ASO en las siguientes situaciones:

- Se ha modificado el archivo AS correspondiente.
- Se han modificado los archivos ActionScript que contienen las definiciones importadas o utilizadas por el correspondiente archivo ActionScript.
- Se han modificado los archivos incluidos por el correspondiente archivo ActionScript.

El compilador crea archivos ASO con el fin de generar cachés. Puede que observe que la primera compilación es más lenta que las compilaciones posteriores. Esto se debe a que sólo se recompilan en los archivos ASO los archivos AS que han cambiado. En el caso de los archivos AS que no han cambiado, el compilador lee directamente el código de bytes ya compilado del archivo ASO en lugar de volver a compilar el archivo AS.

El formato de archivo ASO es un formato intermedio desarrollado para uso interno exclusivamente. No es un formato de archivo documentado, ya que no está pensado para su redistribución.

Si experimenta problemas en los que parezca que Flash está compilando versiones anteriores de un archivo que ha editado, elimine los archivos ASO y vuelva a compilarlos. Si va a eliminar archivos ASO, elimínelos cuando Flash no esté realizando otras operaciones, como la revisión de sintaxis o la exportación de archivos SWF.

### Para eliminar archivos ASO:

Si está editando un archivo FLA y desea eliminar un archivo ASO, seleccione uno de estos métodos en el entorno de edición:

- Seleccione Control > Eliminar archivos ASO para eliminar los archivos ASO y continuar con la edición.
- Seleccione Control > Eliminar archivos ASO y probar película para eliminar los archivos ASO y probar la aplicación.

Si está editando un documento de ActionScript en la ventana Script:

- Seleccione Control > Eliminar archivos ASO para eliminar los archivos ASO y continuar con la edición.
- Seleccione Control > Eliminar archivos ASO y probar proyecto para eliminar los archivos ASO y probar la aplicación.

La cantidad de código que puede situar en una sola clase está limitada: el código de bytes para una definición de clase en un archivo SWF exportado no puede superar los 32.767 bytes. Si el código de bytes supera este límite, aparecerá un mensaje de advertencia.

Puede predecir el tamaño de la representación del código de bytes de una clase determinada, aunque las clases de hasta 1.500 líneas no suelen superar el límite.

Si una clase supera el límite, traslade parte del código a otra clase. En general, es recomendable que las clases sean relativamente breves.

# Clases y ámbito

Si transfiere código ActionScript a clases, es posible que deba cambiar la manera en que utiliza la palabra clave `this`. Por ejemplo, si tiene un método de clase que utiliza una función callback (como el método `onLoad()` de la clase LoadVars), puede ser difícil saber si la palabra clave `this` hace referencia a la clase o al objeto LoadVars. En esta situación, puede que sea necesario crear un puntero a la clase actual, como se muestra en el siguiente ejemplo.

### Para comprender el ámbito y los archivos de clases externas:

1. Seleccione Archivo > Nuevo, elija Archivo ActionScript y haga clic en Aceptar.
2. Escriba o pegue el siguiente código en la ventana Script:

```
/**
  Product class
  Product.as
*/
class Product {
  private var productsXml:XML;
  // constructor
  // targetXmlStr - cadena, contiene la ruta a un archivo XML
  function Product(targetXmlStr:String) {
    /* Crear una referencia local a la clase actual.
       Aunque esté dentro del controlador de eventos onLoad de XML,
       puede hacer referencia a la clase actual y no sólo al paquete XML.
    */
    var thisObj:Product = this;
    // Crear una variable local, que se utiliza para cargar el archivo
   XML.
    var prodXml:XML = new XML();
    prodXml.ignoreWhite = true;
    prodXml.onLoad = function(success:Boolean) {
      if (success) {
        /* Si el XML se carga y analiza correctamente,
           establezca la variable productsXml de la clase en el
           documento XML analizado y llame a la función init.
        */
        thisObj.productsXml = this;
        thisObj.init();
      } else {
        /* Error al cargar el archivo XML. */
        trace("error loading XML");
      }
    };
    // Iniciar carga del documento XML.
    prodXml.load(targetXmlStr);
  }
  public function init():Void {
    // Mostrar el paquete XML.
    trace(this.productsXml);
  }
}
```

Dado que se intenta hacer referencia a la variable de miembro privada de un controlador `onLoad`, la palabra clave `this` realmente hace referencia a la instancia `prodXml` y no a la clase Product, que es lo que podría esperarse. Por este motivo, debe crear un puntero al archivo de clase local para que pueda hacer referencia a la clase directamente desde el controlador `onLoad`. Ahora puede utilizar esta clase con un documento de Flash.

3. Guarde el código ActionScript anterior como **Product.as**.
4. Cree un nuevo documento de Flash denominado **testProduct.fla** en el mismo directorio.
5. Seleccione el fotograma 1 de la línea de tiempo principal.
6. Introduzca el siguiente código ActionScript en el panel Acciones:

```
var myProduct:Product = new Product("http://www.helpexamples.com/
  crossdomain.xml");
```

7. Seleccione Control > Probar película para comprobar este código en el entorno de prueba.

   El contenido del documento XML especificado aparece en el panel Salida.

Otro tipo de ámbito que puede encontrar al utilizar estas clases son las variables estáticas y las funciones estáticas. La palabra clave `static` especifica que una variable o función se crea únicamente una vez por cada clase, en lugar de crearse en cada instancia de dicha clase. Puede acceder a un miembro de clase estático sin crear una instancia de la clase utilizando la sintaxis `someClassName.username`. Para más información sobre variables y funciones estáticas, consulte "Métodos y propiedades (miembros) públicos, privados y estáticos" en la página 217 y "Utilización de miembros de clase" en la página 224.

Otra ventaja que aportan las variables estáticas es que no pierden su valor después de finalizar el ámbito de la variable. En el siguiente ejemplo se muestra cómo utilizar la palabra clave `static` para crear un contador que rastrea el número de instancias de la clase Flash que se han creado. Dado que la variable `numInstances` es estática, la variable se crea una sola vez para toda la clase, no una vez por cada instancia individual.

### Para utilizar la palabra clave static:

1. Seleccione Archivo > Nuevo, elija Archivo ActionScript y haga clic en Aceptar.
2. Escriba el siguiente código en la ventana Script:

```
class User {
  private static var numInstances:Number = 0;
  public function User() {
    User.numInstances++;
  }
  public static function get instances():Number {
    return User.numInstances;
  }
}
```

El código anterior define una clase User que rastrea el número de veces que se ha llamado al constructor. Una variable privada estática (`User.numInstances`) se incrementa dentro del método constructor.

3. Guarde el documento como **User.as**.
4. Seleccione Archivo > Nuevo, elija Documento de Flash para crear un nuevo archivo FLA y guárdelo en el mismo directorio que User.as.
5. Escriba el siguiente código ActionScript en el fotograma 1 de la línea de tiempo:

```
trace(User.instances); // 0
var user1:User = new User();
trace(User.instances); // 1
var user2:User = new User();
trace(User.instances); // 2
```

   La primera línea de código llama al método getter (captador) estático `instances()`, que devuelve el valor de la variable privada estática `numInstances`. El resto del código crea nuevas instancias de la clase User y muestra el valor actual devuelto por el método getter `instances()`.

6. Seleccione Control > Probar película para probar los documentos.

Para más información sobre el uso de la palabra clave `this` en las clases, consulte "Utilización de la palabra clave this en clases" en la página 233.

# Clases de nivel superior y clases incorporadas

Además de los principales elementos de lenguaje y construcciones de ActionScript (como los bucles `for` y `while`, por ejemplo) y los tipos de datos simples (números, cadenas y booleanos) descritos anteriormente en este manual (consulte el Capítulo 10, "Datos y tipos de datos", en la página 327 y Capítulo 4, "Principios básicos de la sintaxis y el lenguaje", en la página 75), ActionScript también proporciona una serie de clases incorporadas (*tipos de datos complejos*). Estas clases ofrecen diversas funciones para la creación de scripts. Ya ha utilizado clases del más alto nivel y otras clases incorporadas que forman parte del lenguaje ActionScript en capítulos anteriores y continuará utilizándolas en los restantes capítulos. Existen muchas clases que se entregan con Flash que permiten crear interactividad y funcionalidad en los archivos SWF. Puede incluso utilizarlas para crear aplicaciones complejas. Por ejemplo, puede utilizar la clase Math para incluir ecuaciones en las aplicaciones. O puede utilizar la clase BitmapData para crear píxeles y animaciones mediante scripts.

Las clases de nivel superior, enumeradas en "Clases de nivel superior" en la página 261, están escritas en Flash Player. En la caja de herramientas Acciones, estas clases se encuentran en el directorio Clases de ActionScript 2.0. Algunas de las clases de nivel superior están basadas en la especificación del lenguaje ECMAScript (ECMA-262) edición 3 y se conocen como *clases principales de ActionScript*. Entre ellas se incluyen las clases Array, Boolean, Date y Math. Para más información sobre paquetes, consulte "Utilización de paquetes" en la página 199.

Dispone de clases de ActionScript instaladas en el disco duro. Puede encontrar las carpetas de clases aquí:

- Windows: Disco duro\Documents and Settings*usuario*\Configuración local\Datos de programa\Macromedia\Flash 8*idioma*\Configuration\Classes.
- Macintosh: Disco duro/Users/*usuario*/Library/Application Support/Macromedia/Flash 8/*idioma*/Configuration/Classes.

Consulte el documento Readme ubicado en este directorio para obtener más información sobre su estructura.

Para comprender la distinción que existe entre las clases principales de ActionScript y las clases específicas de Flash, tenga en cuenta la distinción entre las clases JavaScript principales y las de la parte del cliente. Las clases JavaScript de la parte del cliente controlan el entorno del cliente (el contenido de la página Web y el navegador Web), mientras que las clases específicas de Flash controlan el aspecto y el comportamiento de una aplicación Flash en tiempo de ejecución.

El resto de las clases incorporadas de ActionScript son específicas de Macromedia Flash y del modelo de objetos de Flash Player. Ejemplos de estas clases son Camera, MovieClip y LoadVars. Otras clases están organizadas en paquetes, como flash.display. Todas estas clases se denominan a veces clases *incorporadas* (clases predefinidas que puede utilizar para añadir funcionalidad a las aplicaciones).

En las siguientes secciones se presentan las clases de ActionScript incorporadas y se describen las tareas fundamentales que se realizan con este tipo de clases. Para obtener información general sobre cómo utilizar las clases y los objetos en un entorno de programación orientada a objetos, consulte "Utilización de clases incorporadas" en la página 269. Los ejemplos de código que utilizan estas clases se incluyen en todo el manual *Aprendizaje de ActionScript 2.0 en Flash*.

Para obtener información sobre elementos del lenguaje (como constantes, operadores y directivas), consulte el Capítulo 4, "Principios básicos de la sintaxis y el lenguaje", en la página 75.

Para más información sobre las clases incorporadas y de nivel superior, consulte los temas siguientes:


- "Clases de nivel superior" en la página 261
- "El paquete flash.display" en la página 265
- "El paquete flash.external" en la página 265
- "El paquete flash.filters" en la página 265
- "El paquete flash.geom" en la página 267
- "El paquete flash.net" en la página 267
- "El paquete flash.text" en la página 268
- "El paquete mx.lang" en la página 268
- "Los paquetes System y TextField" en la página 268


## Otros elementos del lenguaje

Existen otros elementos del lenguaje que conforman ActionScript, aparte de las clases. Entre éstos se incluyen directivas, constantes, funciones globales, propiedades globales, operadores y sentencias. Para obtener información sobre la forma de utilizar cada uno de estos elementos del lenguaje, consulte los siguientes temas:


- Capítulo 4, "Principios básicos de la sintaxis y el lenguaje"
- Capítulo 5, "Funciones y métodos"


Podrá encontrar una lista de estos elementos en las siguientes secciones de *Referencia del lenguaje ActionScript 2.0*:

- %{directiva Compiler}%
- %{Constants}%
- %{Global Functions}%
- %{propiedad Global}%
- %{Operators}%
- %{Statements}%

# Clases de nivel superior

En el nivel superior se incluyen las clases y funciones globales de ActionScript, muchas de las cuales proporcionan funcionalidad básica para las aplicaciones. Entre las *clases principales*, tomadas directamente de ECMAScript, se encuentran Array, Boolean, Date, Error, Function, Math, Number, Object, String y System. Para más información sobre cada clase, consulte la siguiente tabla.

| NOTA | Las clases CustomActions y XMLUI sólo están disponibles en el entorno de edición de Flash. |
|---|---|

| Clase | Descripción |
|---|---|
| Accessibility | La clase Accessibility gestiona la comunicación entre los archivos SWF y las aplicaciones de lectura en pantalla. Los métodos de esta clase se utilizan con la propiedad `_accProps` global para controlar las propiedades accesibles de los clips de película, los botones y los campos de texto en tiempo de ejecución. Véase %{Accessibility}%. |
| Array | La clase Array representa a las matrices de ActionScript; todos los objetos de matriz son instancias de esta clase. La clase Array contiene métodos y propiedades para trabajar con objetos de matriz. Véase %{Array}%. |
| AsBroadcaster | Ofrece prestaciones de notificación y de administración de detector que pueden añadirse a otros objetos. Véase %{AsBroadcaster}%. |
| Boolean | Esta clase es un envolvente para los valores booleanos (`true` o `false`). Véase %{Boolean}%.›. |
| Button | Esta clase proporciona métodos, propiedades y controladores de eventos para trabajar con botones. Véase %{Button}%. Tenga en cuenta que la clase incorporada Button es distinta de la clase de componente Button, asociada con el componente de la versión 2, Button. |
| Camera | La clase Camera proporciona acceso a la cámara del usuario, si es que está instalada. Cuando se utiliza con Flash Communication Server, el archivo SWF puede capturar, difundir y registrar imágenes y vídeo desde la cámara de un usuario. Véase %{Camera}%. |
| Color | La clase Color le permite establecer el valor de color RGB y la transformación de color de instancias de clips de película y recuperar dichos valores una vez establecidos. La clase Color deja de admitirse en Flash Player 8 en favor de la clase ColorTransform. Para obtener información sobre las transformaciones de colores, consulte %{ColorTransform (flash.geom.ColorTransform)}%. |

| Clase | Descripción |
|---|---|
| ContextMenu | La clase ContextMenu permite controlar el contenido del menú contextual de Flash Player durante la ejecución. Puede asociar distintos objetos ContextMenu con objetos MovieClip, Button o TextField utilizando la propiedad `menu` disponible para dichas clases. También puede añadir elementos de menú personalizados a un objeto ContextMenu utilizando la clase ContextMenuItem. Véase %{ContextMenu}%. |
| ContextMenuItem | La clase ContextMenuItem permite crear nuevos elementos de menú que aparecen en el menú contextual de Flash Player. Los nuevos elementos de menú que se crean con esta clase se añaden al menú contextual de Flash Player utilizando la clase ContextMenu. Véase %{ContextMenuItem}%. |
| CustomActions | La clase CustomActions permite gestionar las acciones personalizadas que se registran con la herramienta de edición. Véase %{CustomActions}%. |
| Date | La clase Date muestra cómo se representan las fechas y horas en ActionScript y admite operaciones para la manipulación de fechas y horas. La clase Date también proporciona un medio de obtención de la fecha y la hora actuales del sistema operativo. Véase %{Date}%. |
| Error | Esta clase contiene información sobre los errores de tiempo de ejecución que se detectan en los scripts. Normalmente se utiliza la sentencia `throw` para generar una condición de error, la cual puede gestionarse utilizando una sentencia `try..catch..finally`. Véase %{Error}%. |
| Function | Es la representación en clase de todas las funciones de ActionScript, tanto las originales de ActionScript como las que defina el usuario. Véase %{Function}%. |
| Key | Esta clase proporciona métodos y propiedades para obtener información sobre el teclado y las teclas que se presionan. Véase %{Key}%. |
| LoadVars | La clase LoadVars le permite transferir variables entre un archivo SWF y un servidor en pares nombre-valor. Véase %{LoadVars}%. |
| LocalConnection | La clase LocalConnection le permite desarrollar archivos SWF capaces de enviarse instrucciones unos a otros sin utilizar el método `fscommand()` ni JavaScript. Véase %{LocalConnection}%. |
| Math | La clase Math sirve para acceder cómodamente a constantes y funciones matemáticas comunes. Todas las propiedades y métodos de la clase Math son estáticos y deben llamarse mediante la sintaxis `Math.`*`method`*`(`*`parameter`*`)` o `Math.`*`constant`*. Véase %{Math}%. |

| Clase | Descripción |
|---|---|
| Microphone | La clase Microphone proporciona acceso al micrófono del usuario, si es que hay uno instalado. Cuando se utiliza con Flash Communication Server, el archivo SWF puede difundir y registrar audio desde el micrófono de un usuario. Véase %{Microphone}%. |
| Mouse | La clase Mouse controla el ratón en un archivo SWF; por ejemplo, esta clase permite ocultar o mostrar el puntero del ratón. Véase %{Mouse}%. |
| MovieClip | Cada uno de los clips de película de un archivo SWF es una instancia de la clase MovieClip. Los métodos y propiedades de esta clase sirven para controlar los objetos de clip de película. Véase %{MovieClip}%. |
| MovieClipLoader | Esta clase le permite implementar funciones callback de detector que proporcionan información de estado mientras se están cargando archivos SWF, JPEG, GIF y PNG en instancias de clips de película. Véase %{MovieClipLoader}%. |
| NetConnection | La clase NetConnection establece una conexión de transmisión local para reproducir un archivo Flash Video (FLV) desde una dirección HTTP o desde un sistema de archivos local. Véase %{NetConnection}%. |
| NetStream | La clase NetStream controla la reproducción de archivos FLV de un sistema de archivos local o de una dirección HTTP. Véase %{NetStream}%. |
| Number | Es un envolvente para el tipo de datos primitivo de número. Véase %{Number}%. |
| Object | Esta clase se encuentra en la raíz de la jerarquía de clases de ActionScript; el resto de las clases heredan sus métodos y sus propiedades. Véase %{Object}%. |
| PrintJob | La clase PrintJob permite imprimir contenido de un archivo SWF, incluido contenido que se presenta dinámicamente, y documentos de varias páginas. Véase %{PrintJob}%. |
| Selection | La clase Selection le permite establecer y controlar el campo de texto en el que se encuentra el punto de inserción (el campo de texto seleccionado). Véase %{Selection}%. |
| SharedObject | La clase SharedObject proporciona almacenamiento de datos local persistente en el equipo cliente, de manera similar a las cookies. Esta clase permite compartir datos en tiempo real entre objetos en el equipo del cliente. Véase %{SharedObject}%. |
| Sound | La clase Sound controla los sonidos de un archivo SWF. Véase %{Sound}%. |

| Clase | Descripción |
|---|---|
| Stage | Esta clase proporciona información sobre las dimensiones, la alineación y el modo de escala de un archivo SWF. También notifica los eventos de cambio de tamaño del escenario. Véase %{Stage}%. |
| String | Es un envolvente para el tipo de datos primitivo de cadena, que permite utilizar los métodos y las propiedades del objeto String para manipular tipos de valores de cadena primitivos. Véase %{String}%. |
| System | La clase System proporciona información sobre Flash Player y el sistema en el que se ejecuta Flash Player (por ejemplo, la resolución de pantalla y el lenguaje actual del sistema). También permite mostrar u ocultar el panel de configuración de Flash Player y modificar la configuración de seguridad del archivo SWF. Véase %{System}%. |
| TextField | La clase TextField proporciona control sobre los campos dinámicos y de entrada de texto, como la recuperación de información de formato, la invocación de controladores de eventos y el cambio de propiedades del tipo alpha o background. Véase %{TextField}%. |
| TextFormat | La clase TextFormat permite aplicar estilos de formato a caracteres o parágrafos de un objeto TextField. Véase %{TextFormat}%. |
| TextSnapshot | El objeto TextSnapshot le permite acceder y diseñar texto estático dentro de un clip de película. Véase %{TextSnapshot}%. |
| Video | La clase Video le permite mostrar objetos de vídeo en un archivo SWF. Puede utilizar esta clase con Flash Communication Server para mostrar un flujo de vídeo en vivo en un archivo SWF, o dentro de Flash para mostrar un archivo Flash Video (FLV). Véase %{Video}%. |
| XML | Esta clase contiene métodos y propiedades para trabajar con objetos XML. Véase %{XML}%. |
| XMLNode | Esta clase representa un único nodo de un árbol de documentos XML. Es la superclase de la clase XML. Véase %{XMLNode}%. |
| XMLSocket | La clase XMLSocket permite crear una conexión de socket permanente entre un equipo servidor y un cliente que ejecute Flash Player. Los sockets de cliente permiten la transferencia de datos de baja latencia, como la que necesitan las aplicaciones de chat en tiempo real. Véase %{XMLSocket}%. |
| XMLUI | El objeto XMLUI permite la comunicación con archivos SWF que se utilizan como interfaces de usuario personalizadas para las funciones de extensibilidad de la herramienta de edición de Flash (como, por ejemplo, Comportamientos, Comandos, Efectos y Herramientas). Véase %{XMLUI}%. |

# El paquete flash.display

El paquete flash.display contiene la clase BitmapData, que puede utilizar para crear presentaciones visuales.

| Clase | Descripción |
|---|---|
| BitmapData | La clase BitmapData le permite crear en el documento mapas de bits o imágenes transparentes u opacos con tamaño arbitrario y manipularlos de diversas formas durante la ejecución. Véase %{BitmapData (flash.display.BitmapData)}%. |

# El paquete flash.external

El paquete flash.external le permite comunicarse con el contenedor de Flash Player empleando código ActionScript. Por ejemplo, si incorpora un archivo SWF en una página HTML, dicha página HTML es el contenedor. Podría comunicarse con la página HTML empleando la clase ExternalInterface y JavaScript. También denominada la API externa.

| Clase | Descripción |
|---|---|
| ExternalInterface | La clase ExternalInterface es la API externa, un subsistema que permite la comunicación entre el código ActionScript y el contenedor de Flash Player (como una página HTML que utiliza JavaScript) o una aplicación de escritorio que utiliza Flash Player. Véase %{ExternalInterface (flash.external.ExternalInterface)}%. |

# El paquete flash.filters

El paquete flash.filters contiene clases para los efectos de filtro de mapa de bits disponibles en Flash Player 8. Los filtros permiten aplicar efectos visuales muy diversos, como desenfoque, bisel, iluminación y sombreado, a instancias de Image y MovieClip. Para más información sobre cada clase, consulte las referencias cruzadas proporcionadas en la siguiente tabla.

| Clase | Descripción |
|---|---|
| BevelFilter | *La clase BevelFilter le permite añadir un efecto de bisel a una instancia de clip de película.* Véase %{BevelFilter (flash.filters.BevelFilter)}%. |
| BitmapFilter | La clase BitmapFilter es una clase básica para todos los efectos de filtro. Véase %{BitmapFilter (flash.filters.BitmapFilter)}%. |

| Clase | Descripción |
|---|---|
| BlurFilter | *La clase* BlurFilter *le permite aplicar un efecto de desenfoque a instancias de clips de película.* Véase %{BlurFilter (flash.filters.BlurFilter)}%. |
| ColorMatrixFilter | La clase ColorMatrixFilter le permite aplicar una transformación de matriz de 4 x 5 en los valores de color ARGB y alfa de cada píxel de la imagen de entrada. Tras aplicar la transformación, puede producir un resultado con un nuevo conjunto de valores de color ARGB y alfa. Véase %{ColorMatrixFilter (flash.filters.ColorMatrixFilter)}%. |
| ConvolutionFilter | La clase ConvolutionFilter le permite aplicar un efecto de filtro de convolución de matrices. Véase %{ConvolutionFilter (flash.filters.ConvolutionFilter)}%. |
| DisplacementMapFilter | La clase DisplacementMapFilter le permite utilizar valores de píxel de una imagen especificada (la imagen del mapa de desplazamiento) para desplazar espacialmente la instancia original (un clip de película) a la que se aplica el filtro. Véase %{DisplacementMapFilter (flash.filters.DisplacementMapFilter)}%. |
| DropShadowFilter | La clase DropShadowFilter le permite añadir sombreado a un clip de película. Véase %{DropShadowFilter (flash.filters.DropShadowFilter)}%. |
| GlowFilter | La clase GlowFilter le permite añadir un efecto de iluminación a un clip de película. Véase %{GlowFilter (flash.filters.GlowFilter)}%. |
| GradientBevelFilter | La clase GradientBevelFilter le permite aplicar un efecto de bisel degradado a un clip de película. Véase %{GradientBevelFilter (flash.filters.GradientBevelFilter)}%. |
| GradientGlowFilter | La clase GradientGlowFilter le permite aplicar un efecto de iluminación degradada a un clip de película. Véase %{GradientGlowFilter (flash.filters.GradientGlowFilter)}%. |

# El paquete flash.geom

El paquete flash.geom contiene clases geométricas, como puntos, rectángulos y matrices de transformación. Estas clases permiten utilizar la clase BitmapData y la función de caché de mapa de bits. Para más información sobre cada clase, consulte las referencias cruzadas proporcionadas en la siguiente tabla.

| Clase | Descripción |
|---|---|
| ColorTransform | La clase ColorTransform le permite establecer matemáticamente el valor de color RGB y la transformación de color de una instancia. Puede recuperar estos valores después de haberlos establecido. Véase %{ColorTransform (flash.geom.ColorTransform)}%. |
| Matrix | Representa una matriz de transformación que determina cómo asignar puntos de un espacio de coordenadas a otro. Véase %{Matrix (flash.geom.Matrix)}%. |
| Point | El objeto Point representa una ubicación en un sistema de coordenadas bidimensional, donde `x` representa el eje horizontal e `y` representa el eje vertical. Véase %{Point (flash.geom.Point)}%. |
| Rectangle | La clase Rectangle se utiliza para crear y modificar objetos Rectangle. Véase %{Rectangle (flash.geom.Rectangle)}%. |
| Transform | Recopila datos acerca de las transformaciones de color y manipulaciones de coordenadas que se aplican a una instancia de objeto. Véase %{Transform (flash.geom.Transform)}%. |

# El paquete flash.net

El paquete flash.net contiene clases que le permiten cargar y descargar uno o varios archivos entre el equipo de un usuario y el servidor. Para más información sobre cada clase, consulte las referencias cruzadas proporcionadas en la siguiente tabla.

| Clase | Descripción |
|---|---|
| FileReference | La clase FileReference le permite cargar y descargar uno o varios archivos entre el equipo de un usuario y un servidor. Véase %{FileReference (flash.net.FileReference)}%. |
| FileReferenceList | La clase FileReferenceList le permite cargar uno o varios archivos del equipo de un usuario en un servidor. Véase %{FileReferenceList (flash.net.FileReferenceList)}%. |

## El paquete flash.text

El paquete flash.text contiene la clase TextRenderer para trabajar con la función de suavizado avanzado disponible en Flash Player 8.

| Clase | Descripción |
|---|---|
| TextRenderer | Esta clase ofrece la funcionalidad necesaria para la capacidad de suavizado avanzado presente en Flash Player 8. Consulte %{TextRenderer (flash.text.TextRenderer)}%. |

## El paquete mx.lang

El paquete mx.lang incluye la clase Locale para trabajar con texto en varios idiomas.

| Clase | Descripción |
|---|---|
| Locale | Esta clase le permite controlar la forma en la que se muestra texto en varios idiomas en un archivo SWF. Véase %{Locale (mx.lang.Locale)}%. |

## Los paquetes System y TextField

El paquete System contiene las prestaciones, el IME y las clases de seguridad. Estas clases se ocupan de la configuración del cliente que podría afectar a la aplicación en Flash Player. Para más información sobre cada clase, consulte las referencias cruzadas proporcionadas en la siguiente tabla.

| Clase | Descripción |
|---|---|
| capabilities | La clase capabilities determina las capacidades del sistema y de la versión de Flash Player en que se aloja el archivo SWF. Esto le permite personalizar el contenido para diferentes formatos. Véase %{capabilities (System.capabilities)}%. |
| IME | La clase IME permite manipular directamente el editor de método de entrada (IME) del sistema operativo en la aplicación Flash Player que se ejecuta en un equipo cliente. Véase %{IME (System.IME)}%. |
| security | La clase security contiene métodos que especifican cómo pueden comunicarse entre sí archivos SWF de diferentes dominios. Véase %{security (System.security)}%. |

El paquete TextField contiene la clase StyleSheet que puede utilizar para aplicar estilos CSS al texto.

| Clase | Descripción |
|---|---|
| StyleSheet | La clase StyleSheet permite crear un objeto de hoja de estilos que contenga reglas de formato de texto, como tamaño de fuente, color y otros estilos de formato. Véase %{StyleSheet (TextField.StyleSheet)}%. |

# Utilización de clases incorporadas

En un entorno de programación orientada a objetos (OOP), una *clase* define una categoría de objetos. Una clase describe las propiedades (datos) y el comportamiento (métodos) de un objeto, del mismo modo que un plano arquitectónico describe las características de un edificio. Para obtener información sobre clases y otros conceptos de programación orientada a objetos, consulte las siguientes secciones:

- "Conceptos básicos sobre la programación orientada a objetos" en la página 201
- "Escritura de archivos de clases personalizadas" en la página 205

Flash 8 has dispone de un gran número de clases incorporadas que puede utilizar en el código (consulte "Clases de nivel superior y clases incorporadas" en la página 258) y que le ayudarán a añadir fácilmente interactividad a las aplicaciones. Para utilizar las propiedades y los métodos definidos por una clase incorporada, primero debe crear una *instancia* de dicha clase (salvo en el caso de clases que tengan miembros estáticos). La relación entre una instancia y su clase es similar a la relación entre una casa y sus planos arquitectónicos, como se trata en la sección "Clases de nivel superior y clases incorporadas" en la página 258.

Para más información sobre la utilización de clases que se han incorporado en Flash 8, consulte los siguientes temas:

- "Creación de nuevas instancias de clases incorporadas" en la página 270
- "Acceso a las propiedades de los objetos incorporados" en la página 270
- "Llamada a métodos de objeto incorporados" en la página 271
- "Miembros de clase (estáticos)" en la página 271
- "Precarga de archivos de clase" en la página 273
- "Exclusión de clases" en la página 272

## Creación de nuevas instancias de clases incorporadas

Para crear una instancia de una clase de ActionScript, utilice el operador `new` para invocar la función constructora de la clase. La función constructora tiene siempre el mismo nombre que la clase, y devuelve una instancia de la clase, que normalmente se asigna a una variable.

Por ejemplo, el código siguiente crea un nuevo objeto Sound:

```
var song_sound:Sound = new Sound();
```

En algunos casos, no es necesario crear una instancia de una clase para utilizar sus propiedades y métodos. Para más información, consulte "Miembros de clase (estáticos)" en la página 271.

## Acceso a las propiedades de los objetos incorporados

Utilice el operador de punto (`.`) para acceder al valor de una propiedad en un objeto. Ponga el nombre del objeto a la izquierda del punto y el nombre de la propiedad en el lado derecho. Por ejemplo, en la siguiente sentencia, `my_obj` es el objeto y `firstName`, la propiedad:

```
my_obj.firstName
```

El siguiente código crea un nuevo objeto Array y luego muestra su propiedad `length`:

```
var my_array:Array = new Array("apples", "oranges", "bananas");
trace(my_array.length); // 3
```

También puede utilizar el operador de acceso a una matriz (`[]`) para acceder a las propiedades de un objeto, por ejemplo, al utilizar el operador de acceso a matriz para fines de depuración. El siguiente ejemplo reproduce indefinidamente un objeto para mostrar cada una de sus propiedades.

### Para reproducir indefinidamente el contenido de un objeto:

1. Cree un nuevo documento de Flash y guárdelo como **forin.fla**.
2. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

   ```
   var results:Object = {firstName:"Tommy", lastName:"G", age:7, avg:0.336,
     b:"R", t:"L"};
   for (var i:String in results) {
       trace("the value of [" + i + "] is: " + results[i]);
   }
   ```

   El código anterior define un nuevo elemento Object denominado "results" y define los valores de `firstName`, `lastName`, `age`, `avg`, `b` y `t`. Un bucle `for..in` traza cada propiedad en el objeto "results" y traza su valor en el panel Salida.
3. Seleccione Control > Probar película para probar el documento de Flash.

Para más información sobre operadores, incluidos los de punto y acceso a una matriz, consulte "Operadores" en la página 142. Para más información sobre métodos y propiedades, consulte el Capítulo 5, "Funciones y métodos", en la página 169. Para ver ejemplos de la utilización de propiedades de la clase MovieClip incorporada, consulte el Capítulo 11, "Trabajo con clips de película", en la página 373. Para ver ejemplos de la utilización de propiedades de las clases TextField, String, TextRenderer y TextFormat, consulte el Capítulo 12, "Utilización de texto y cadenas", en la página 407.

## Llamada a métodos de objeto incorporados

Puede llamar al método de un objeto mediante el operador de punto (`.`) seguido por el método. Por ejemplo, el código siguiente crea un nuevo objeto Sound y llama a su método `setVolume()`:

```
var my_sound:Sound = new Sound(this);
my_sound.setVolume(50);
```

Para ver ejemplos sobre la utilización de métodos de la clase MovieClip incorporada, consulte el Capítulo 11, "Trabajo con clips de película", en la página 373. Para ver ejemplos sobre la utilización de métodos de las clases TextField, String, TextRenderer y TextFormat incorporadas, consulte el Capítulo 12, "Utilización de texto y cadenas", en la página 407.

## Miembros de clase (estáticos)

Algunas clases de ActionScript incorporadas tienen *miembros de clase* (o *miembros estáticos*). Se accede o se invoca a los miembros de clase (propiedades y métodos) desde el propio nombre de la clase, no desde una instancia de la misma. Por consiguiente, no se crea una instancia de la clase para utilizar estas propiedades y métodos.

Por ejemplo, todas las propiedades de la clase Math son estáticas. El código siguiente invoca el método `max()` de la clase Math para determinar entre dos números cuál es el mayor:

```
var largerNumber:Number = Math.max(10, 20);
trace(largerNumber); // 20
```

Para más información sobre los métodos estáticos de la clase Math, así como ejemplos de su uso, consulte %{Math}% en *Referencia del lenguaje ActionScript 2.0*.

# Exclusión de clases

Para reducir el tamaño de un archivo SWF, puede excluir clases de la compilación sin que ello le impida acceder a ellas y utilizarlas para la verificación de tipos. Por ejemplo, es posible que le interese hacer esto si está desarrollando una aplicación que utiliza varios archivos SWF o bibliotecas compartidas, particularmente en el caso de que acceden a muchas de las mismas clases. La exclusión de clases le ayudan a evitar la duplicación de clases en dichos archivos.

Para más información sobre la exclusión de clases, consulte los siguientes temas:

- "Precarga de archivos de clase" en la página 273

### Para excluir clases de la compilación:

1. Cree un nuevo archivo XML.
2. Asigne al archivo XML el nombre ***FLA_filename*_exclude.xml**, donde *FLA_filename* es el nombre de su archivo FLA sin la extensión.

   Por ejemplo, si su archivo FLA se llama sellStocks.fla, el nombre del archivo XML deberá ser sellStocks_exclude.xml.
3. Guarde el archivo en el mismo directorio que el archivo FLA.
4. Sitúe las siguientes etiquetas en el archivo XML:

   ```
   <excludeAssets>
     <asset name="className1" />
     <asset name="className2" />
   </excludeAssets>
   ```

   Los valores que especifique para los atributos name en las etiquetas `<asset>` serán nombres de clases que desee excluir del archivo SWF. Añada tantos como sean necesarios para su aplicación. Por ejemplo, el siguiente archivo XML excluye las clases mx.core.UIObject y mx.screens.Slide del archivo SWF:

   ```
   <excludeAssets>
     <asset name="mx.core.UIObject" />
     <asset name="mx.screens.Slide" />
   </excludeAssets>
   ```

Para obtener información sobre la precarga de clases, consulte "Precarga de archivos de clase" en la página 273.

# Precarga de archivos de clase

En esta sección se describen algunas de las metodologías para precargar y exportar clases en Flash 8 (incluidas las clases que utilizan los componentes de la versión 2 de la arquitectura de componentes de Macromedia). La *precarga* implica cargar algunos de los datos de un archivo SWF antes de que el usuario inicie la interacción con el archivo. Flash importa clases en el primer fotograma de un archivo SWF cuando se utilizan clases externas y estos datos son el primer elemento que se carga en un archivo SWF. El proceso es similar en las clases de componentes porque el marco de componentes también se carga en el primer fotograma de un archivo SWF. Cuando se crean aplicaciones grandes, el tiempo de carga puede ser largo si es necesario importar datos, de modo que hay que manejar los datos de forma inteligente, como se muestra en los siguientes procedimientos.

Como las clases son los primeros datos que se cargan, puede tener problemas para crear una barra de progreso o para cargar una animación si las clases se cargan antes de la barra de progreso, porque es probable que desee que la barra refleje el progreso de la carga de todos los datos, incluidas las clases. Por lo tanto, es preferible cargar las clases después de otras partes del archivo SWF pero antes de utilizar los componentes.

El siguiente procedimiento muestra cómo cargar el fotograma en el que las clases se cargan en un archivo SWF.

### Para seleccionar un fotograma distinto para cargar las clases en un archivo SWF:

1. Seleccione Archivo > Configuración de publicación.
2. Seleccione la ficha Flash y haga clic en el botón Configuración.
3. En el cuadro de texto Fotograma de exportación para clases, escriba el número de un nuevo fotograma para determinar cuándo deben cargarse las clases.
4. Haga clic en Aceptar.

No puede utilizar ninguna de las clases hasta que la cabeza lectora alcance el fotograma donde ha elegido que deben cargarse. Por ejemplo, los componentes de la versión 2 necesitan clases para poder funcionar, por lo tanto, debe cargar componentes después del Fotograma de exportación para clases de ActionScript 2.0. Si exporta para el fotograma 3, no podrá utilizar ninguna de las clases hasta que la cabeza lectora alcance el fotograma 3 y cargue los datos.

Si desea precargar un archivo que utiliza clases, como las clases de componentes de la versión 2, debe precargar los componentes en el archivo SWF. Para ello, debe establecer la exportación de componentes en un fotograma distinto en el archivo SWF. De forma predeterminada, los componentes de la interfaz de usuario se exportan en el fotograma 1 del archivo SWF, así que asegúrese de que anula la selección de Exportar en primer fotograma del cuadro de diálogo Vinculación del componente.

Si los componentes no se cargan en el primer fotograma, puede crear una barra de progreso personalizada para el primer fotograma del archivo SWF. No haga referencia a ningún componente en el código ActionScript ni incluya componentes en el escenario hasta que haya cargado las clases para el fotograma especificado en el cuadro de texto Fotograma de exportación para clases.

| ATENCIÓN | Debe exportar los componentes después de las clases de ActionScript que utilizan. |
|---|---|

CAPÍTULO 7

# Herencia

7

En el Capítulo 6, “Clases”, hemos aprendido a escribir archivos de clases y hemos visto cómo las clases ayudan a organizar el código en archivos externos. En dicho capítulo, también se ha descrito cómo organizar los archivos de clases en paquetes relacionados. Este capítulo tiene como objetivo mostrarle cómo escribir clases más avanzadas que amplíen la funcionalidad de una clase existente. Se trata de un asunto de gran utilidad, ya que la extensión de sus propias clases o de clases existentes le permitirá añadir nuevos métodos y propiedades.

Para más información sobre la herencia, consulte “Herencia” en la página 275. Para más información sobre métodos y propiedades, consulte el Capítulo 5, “Funciones y métodos”, en la página 169.

Para más información sobre la herencia, consulte los temas siguientes:



## Herencia

En el Capítulo 6, “Clases” hemos visto cómo podría crearse un archivo de clase para establecer sus propios tipos de datos personalizados. Al aprender a crear archivos de clases personalizados, se conoce la forma de trasladar código de la línea de tiempo a archivos externos. El traslado de código a archivos externos facilita la edición del código. Ahora que ya está familiarizado con los principios básicos de la creación de clases personalizadas, es el momento de aprender una técnica de programación orientada a objetos (OOP) conocida como *creación de subclases* o *ampliación de una clase* y que le permite crear nuevas clases a partir de una clase existente.

Una de las ventajas de la OOP es que puede crear *subclases* de una clase. La subclase hereda todas las propiedades y métodos de una *superclase*. Por ejemplo, si amplía (o *crea una subclase de* ) la clase MovieClip, estará creando una clase personalizada que amplía la clase MovieClip. La subclase hereda todas las propiedades y los métodos de la clase MovieClip. O bien puede crear un conjunto de clases que amplíen una superclase personalizada. Por ejemplo, la clase Lettuce (lechuga) podría ampliar la superclase Vegetable (verduras).

La subclase normalmente define métodos y propiedades adicionales que puede utilizar en la aplicación, de ahí que se conozca como *ampliar* la superclase. Las subclases también pueden suplantar a (proporcionar sus propias definiciones para) los métodos heredados de una superclase. Si una subclase sustituye un método heredado de su superclase, ya no podrá acceder a la definición de la superclase dentro de la subclase. La única excepción a la regla anterior es que, si está dentro de la función constructora de la subclase, llamará al constructor de la superclase mediante la sentencia `super`. Para más información sobre la sustitución, consulte "Sustitución de métodos y propiedades" en la página 281.

Por ejemplo, puede crear una clase Mammal (mamífero) que defina determinadas propiedades y comportamientos comunes a todos los mamíferos. Luego podría crear una subclase Cat (gato) que se ampliara a la clase Mammal. El uso de subclases permite reutilizar código, por lo que, en lugar de volver a crear todo el código común a ambas clases, podría simplemente ampliar una clase existente. A su vez, otra subclase, la clase Siamese (siamés), podría ampliar la clase Cat y, así, sucesivamente. En una aplicación compleja, determinar la estructura jerárquica de las clases constituye una gran parte del proceso de diseño.

La herencia y la creación de subclases resultan muy útiles en aplicaciones grandes, ya que le permiten crear una serie de clases relacionadas que comparten funcionalidad. Por ejemplo, puede crear una clase Employee (empleado) que defina los métodos y propiedades básicos del empleado típico de una empresa. Luego puede crear una nueva clase denominada Contractor (contratista) que amplíe la clase Employee y herede todos sus métodos y propiedades. La clase Contractor podría añadir sus propios métodos y propiedades específicos o sustituir métodos y propiedades definidos en la superclase Employee. Luego puede crear una nueva clase denominada Manager (jefe) que también amplíe la clase Employee y defina métodos y propiedades adicionales como `hire()` (contratar), `fire()` (despedir), `raise()` (subir el sueldo) y `promote()` (ascender). Puede incluso ampliar una subclase, por ejemplo, Manager, y crear una nueva clase denominada Director que, una vez más, añada nuevos métodos o sustituya los métodos existentes.

Cada vez que amplíe una clase existente, la nueva clase heredará todos los métodos y propiedades actuales de la subclase. Si no estuvieran relacionadas las clases, tendría que reescribir cada método y propiedad en cada archivo de clase independiente, aunque la funcionalidad fuese la misma que en clases similares. Tendría que invertir mucho tiempo no solamente en codificar, sino también en depurar la aplicación y mantener el proyecto si una lógica similar cambiara en varios archivos.

En ActionScript, debe utilizar la palabra clave `extends` para establecer la herencia entre una clase y su superclase, o bien ampliar una interfaz. Para más información sobre el uso de la palabra clave `extends`, consulte "Escritura de subclases en Flash" en la página 277 y "Escritura de una subclase" en la página 278. Para obtener información adicional sobre la palabra clave `extends`, consulte %{extends statement}% en *Referencia del lenguaje ActionScript 2.0*.

# Escritura de subclases en Flash

En la programación orientada a objetos, una subclase puede heredar las propiedades y los métodos de otra clase, denominada *superclase*. Puede ampliar sus propias clases, además de muchas clases principales y de ActionScript de Flash Player. No puede ampliar la clase TextField ni clases estáticas, como las clases Math, Key y Mouse.

Para crear este tipo de relaciones entre dos clases, debe utilizar la cláusula `extends` de la sentencia class. Para especificar una superclase, utilice la sintaxis siguiente:

```
class SubClass extends SuperClass {}
```

La clase que se especifica en SubClass hereda todas las propiedades y los métodos definidos en SuperClass.

Por ejemplo, podría crear una clase Mammal que defina propiedades y métodos comunes a todos los mamíferos. Para crear una variación de la clase Mammal, como una clase Marsupial, podría ampliar la clase Mammal, es decir, crear una subclase de la clase Mammal, de la siguiente forma:

```
class Marsupial extends Mammal {}
```

La subclase hereda todas las propiedades y los métodos de la superclase, incluidos las propiedades y los métodos que se han declarado como privados mediante la palabra clave `private`.

Para más información sobre la ampliación de clases, consulte los temas siguientes:

- "Escritura de una subclase" en la página 278
- "Sustitución de métodos y propiedades" en la página 281

Para más información sobre miembros privados, consulte "Métodos y propiedades (miembros) públicos, privados y estáticos" en la página 217. Para ver un ejemplo en el que se crea una subclase, consulte "Ejemplo: Ampliación de la clase Widget" en la página 279.

## Escritura de una subclase

En el siguiente código se define la clase personalizada JukeBox, que amplía la clase Sound. Define una matriz llamada `song_arr` y un método denominado `playSong()` que reproduce una canción e invoca el método `loadSound()`, que hereda de la clase Sound.

```
class JukeBox extends Sound {
  public var song_arr:Array = new Array("beethoven.mp3", "bach.mp3",
  "mozart.mp3");
  public function playSong(songID:Number):Void {
    super.loadSound(song_arr[songID], true);
  }
}
```

Si no efectúa una llamada a `super()` en la función constructora de una subclase, el compilador generará de manera automática una llamada al constructor de su superclase inmediata sin ningún parámetro como primera sentencia de la función. Si la superclase no tiene un constructor, el compilador crea una función constructora vacía y, a continuación, genera un llamada a esa función desde la subclase. No obstante, si la superclase toma parámetros en su definición, debe crear un constructor en la subclase y llamar a la superclase con los parámetros necesarios.

No se permite la herencia múltiple (heredar de más de una clase) en ActionScript 2.0. No obstante, las clases pueden de hecho heredar de múltiples clases si se utilizan sentencias `extends` por separado, como se muestra en el siguiente ejemplo:

```
// no permitido
class C extends A, B {} // **Error: Una clase no podrá ampliar más de una
  clase.

// permitido
class B extends A {}
class C extends B {}
```

También puede utilizar interfaces para implementar una forma de herencia múltiple limitada. Para más información sobre interfaces, consulte el Capítulo 8, “Interfaces”, en la página 289. Para ver un ejemplo en el que se crea una subclase, consulte “Ejemplo: Ampliación de la clase Widget” en la página 279. Para obtener información adicional sobre `super`, consulte %{super statement}% en *Referencia del lenguaje ActionScript 2.0*.

## Ejemplo: Ampliación de la clase Widget

Los miembros de clases se propagan en las subclases de la superclase que definen estos miembros. En el siguiente ejemplo se muestra cómo podría crear una clase Widget que se amplíe (crear una subclase) escribiendo una clase llamada SubWidget.

### Para crear la clase Widget y la subclase SubWidget:

1. Cree un nuevo archivo ActionScript y guárdelo como **Widget.as**.
2. Añada el siguiente código al nuevo documento:

```
class Widget {
  public static var widgetCount:Number = 0;
  public function Widget() {
    Widget.widgetCount++;
  }
}
```

3. Guarde los cambios en el archivo ActionScript.
4. Cree un nuevo archivo ActionScript y guárdelo como **SubWidget.as** en el mismo directorio que la clase Widget.
5. En SubWidget.as, escriba el siguiente código en la ventana Script:

```
class SubWidget extends Widget {
  public function SubWidget() {
    trace("Creating subwidget #" + Widget.widgetCount);
  }
}
```

6. Guarde los cambios en SubWidget.as.
7. Cree un nuevo archivo FLA y guárdelo como **subWidgetTest.fla** en el mismo directorio que los anteriores archivos de clases de ActionScript.
8. En el archivo subWidgetTest.fla, escriba el siguiente código en el fotograma 1 de la línea de tiempo principal:

```
var sw1:SubWidget = new SubWidget();
var sw2:SubWidget = new SubWidget();
trace("Widget.widgetCount = " + Widget.widgetCount);
trace("SubWidget.widgetCount = " + SubWidget.widgetCount);
```

   El código anterior crea dos instancias de la clase SubWidget: sw1 y sw2. Cada llamada al constructor `SubWidget` traza el valor actual de la propiedad estática `Widget.widgetCount`. Debido a que la clase SubWidget es una subclase de la clase Widget, puede acceder a la propiedad `widgetCount` a través de la clase SubWidget y el compilador reescribe la referencia (en el código de bytes, no en el archivo ActionScript) como `Widget.widgetCount`. Si intenta acceder a la propiedad estática `widgetCount` de las instancias de las clase Widget o SubWidget, como sw1 o sw2, el compilador devuelve un error.

9. Guarde los cambios en el documento.

10. Seleccione Control > Probar película para probar el documento de Flash.

    El panel Salida muestra el resultado siguiente:

```
Creating subwidget #1
Creating subwidget #2
Widget.widgetCount = 2
SubWidget.widgetCount = 2
```

    Este resultado aparece porque, a pesar de que en ningún momento se llama de forma explícita al constructor de la clase Widget, sí lo llama automáticamente el constructor de la clase SubWidget. Esto hace que el constructor de la clase Widget incremente la variable estática `widgetCount` de la clase Widget.

    El compilador de ActionScript 2.0 puede resolver referencias a miembros estáticos dentro de las definiciones de clases.

    Si no especifica el nombre de clase para la propiedad `Widget.widgetCount`, sino que sólo hace referencia a `widgetCount`, el compilador de ActionScript 2.0 resuelve la referencia en `Widget.widgetCount` y exporta correctamente dicha propiedad. De igual forma, si hace referencia a la propiedad como `SubWidget.widgetCount`, el compilador reescribe la referencia (en el código de bytes, no en el archivo ActionScript) como `Widget.widgetCount` porque SubWidget es una subclase de la clase Widget.

| ATENCIÓN | Si intenta acceder a la variable estática `widgetCount` de la clase Widget mediante las instancias sw1 o sw2, Flash genera un error en el que se indica que sólo se puede acceder directamente a los miembros estáticos a través de las clases. |
|---|---|

Para que el código alcance una legibilidad óptima, Macromedia recomienda utilizar siempre referencias explícitas a variables de miembros estáticos del código, como se muestra en el ejemplo anterior. El uso de referencias explícitas significa que puede identificar fácilmente el lugar en que se encuentra la definición de un miembro estático.

## Sustitución de métodos y propiedades

Cuando una subclase amplía una superclase, la subclase hereda todos los métodos y propiedades de la superclase. Una de las ventajas que ofrece el uso y la ampliación de clases es que permite, no sólo proporcionar una nueva funcionalidad para una clase existente, sino también modificar la funcionalidad existente. Por ejemplo, tomemos la clase Widget que ha creado en el "Ejemplo: Ampliación de la clase Widget" en la página 279. Podría crear un nuevo método en la superclase (Widget) y luego sustituir el método en la subclase (SubWidget) o simplemente utilizar el método heredado de la clase Widget. En el siguiente ejemplo se muestra cómo sustituir métodos existentes en las clases.

### Para sustituir métodos en una subclase:

1. Cree un nuevo documento ActionScript y guárdelo como **Widget.as**.
2. En Widget.as, escriba el siguiente código ActionScript en la ventana Script.

> **NOTA** Si ha creado la clase Widget en un ejemplo anterior, modifique el código existente añadiendo el método `doSomething()` de la siguiente forma:

```
class Widget {
  public static var widgetCount:Number = 0;
  public function Widget() {
    Widget.widgetCount++;
  }
  public function doSomething():Void {
    trace("Widget::doSomething()");
  }
}
```

3. Guarde los cambios en el documento ActionScript.

   La clase Widget ahora define un constructor y un método público denominado `doSomething()`.

4. Cree un nuevo archivo ActionScript llamado **SubWidget.as** y guárdelo en el mismo directorio que Widget.as.

> **NOTA** Si ha creado la clase SubWidget en "Ejemplo: Ampliación de la clase Widget" en la página 279, puede utilizar este archivo como alternativa.

5. En SubWidget.as, escriba el siguiente código ActionScript en la ventana Script:

```
class SubWidget extends Widget {
  public function SubWidget() {
    trace("Creating subwidget # " + Widget.widgetCount);
    doSomething();
  }
}
```

6. Guarde los cambios en SubWidget.as.

   Observe que el constructor de la clase SubWidget llama al método `doSomething()` definido en la superclase.

7. Cree un nuevo documento de Flash y guárdelo como **subWidgetTest.fla** en el mismo directorio que los documentos ActionScript.

8. En subWidgetTest.fla, escriba el siguiente código ActionScript en el fotograma 1 de la línea de tiempo principal:

```
var sw1:SubWidget = new SubWidget();
var sw2:SubWidget = new SubWidget();
```

9. Guarde los cambios en el documento de Flash.

10. Seleccione Control > Probar película para probar el documento de Flash. Se ve el resultado siguiente en el panel Salida:

```
Creating subwidget #1
Widget::doSomething()
Creating subwidget #2
Widget::doSomething()
```

    El resultado muestra que el constructor de la clase SubWidget llama al constructor de su superclase (Widget), que incrementa la propiedad estática `widgetCount`. El constructor de SubWidget rastrea la propiedad estática de la superclase y llama al método `doSomething()`, que hereda de la superclase.

11. Abra la clase SubWidget y añada un nuevo método denominado `doSomething()`. Modifique la clase para que coincida con el siguiente código (añada el código en negrita):

```
class SubWidget extends Widget {
  public function SubWidget() {
    trace("Creating subwidget # " + Widget.widgetCount);
    doSomething();
  }
  public function doSomething():Void {
    trace("SubWidget::doSomething()");
  }
}
```

12. Guarde los cambios en el archivo de clase y luego abra de nuevo subwidgetTest.fla.

13. Seleccione Control > Probar película para probar el archivo. Se ve el resultado siguiente en el panel Salida:

```
Creating subwidget #1
SubWidget::doSomething()
Creating subwidget #2
SubWidget::doSomething()
```

El resultado anterior muestra que el método `doSomething()` del constructor de la clase SubWidget llama al método `doSomething()` de la clase actual en lugar de llamar al de la superclase.

Abra la clase SubWidget de nuevo y modifique el constructor de la clase SubWidget para que llame al método `doSomething()` de la superclase (añada el código en negrita):

```
public function SubWidget() {
  trace("Creating subwidget # " + Widget.widgetCount);
  super.doSomething();
}
```

Como se ha demostrado, puede añadir la palabra clave `super` para llamar al método `doSomething()` de la superclase en lugar de al método `doSomething()` de la clase actual. Para obtener información adicional sobre `super`, consulte la entrada `super` en Referencia del lenguaje ActionScript 2.0.

14. Guarde el archivo de clase SubWidget con el constructor modificado y seleccione Control > Probar película para volver a publicar el documento de Flash.

El panel Salida muestra el contenido del método `doSomething()` de la clase Widget.

# Utilización de polimorfismo en una aplicación

La programación orientada a objetos le permite expresar diferencias entre clases individuales mediante la técnica conocida como polimorfismo, que permite que las clases sustituyan métodos de sus superclases y definan implementaciones especializadas de dichos métodos.

Por ejemplo, puede comenzar con una clase llamada Mammal que tenga los métodos `play()` y `sleep()`. Puede crear las subclases Cat, Monkey y Dog para ampliar la clase Mammal. Las subclases sustituyen el método `play()` de la clase Mammal para reflejar los hábitos de estos tipos de animales en particular. Monkey implementa el método `play()` para columpiarse de árbol en árbol; Cat implementa el método `play()` para abalanzarse sobre una madeja; Dog implementa el método `play()` para buscar una pelota. Dado que la funcionalidad `sleep()` es similar entre los animales, utilizaría la implementación de la superclase. El siguiente procedimiento demuestra este ejemplo en Flash.

### Para utilizar polimorfismo en una aplicación:

1. Cree un nuevo documento ActionScript y guárdelo como **Mammal.as**.

   Este documento es la clase base para diversas clases de animales diferentes que creará en pasos posteriores.

2. En Mammal.as, escriba el siguiente código ActionScript en la ventana Script:

```
class Mammal {
  private var _gender:String;
  private var _name:String = "Mammal";

  // constructor
  public function Mammal(gender:String) {
    this._gender = gender;
  }

  public function toString():String {
    return "[object " + speciesName + "]";
  }
  public function play():String {
    return "Chase another of my kind.";
  }
  public function sleep():String {
    return "Close eyes.";
  }

  public function get gender():String {
    return this._gender;
  }
  public function get speciesName():String {
    return this._name;
  }
  public function set speciesName(value:String):Void {
    this._name = value;
  }
}
```

   La clase anterior define dos variables privadas, `_gender` y `_name`, que se utilizan para almacenar el género del animal y el tipo de mamífero. A continuación, se define el constructor de `Mammal`. El constructor toma un único parámetro, `gender`, que utiliza para establecer la variable `_gender` definida anteriormente. También se especifican tres métodos públicos adicionales: `toString()`, `play()` y `sleep()`, cada uno de los cuales devuelve objetos de cadena. Los tres últimos métodos son el captador (getter) y los definidores (setters) de las propiedades `_gender` y `_name` del mamífero.

3. Guarde el documento ActionScript.

   Esta clase actúa como superclase de las clases Cat, Dog y Monkey, que crearemos en breve. Puede utilizar el método `toString()` de la clase Mammal para mostrar una representación de cadena de cualquier instancia de Mammal (o cualquier instancia que haya ampliado la clase Mammal).

4. Cree un nuevo archivo ActionScript y guárdelo como **Cat.as** en el mismo directorio que el archivo de clase Mammal.as creado en el paso 1.

5. En Cat.as, escriba el siguiente código ActionScript en la ventana Script:

```
class Cat extends Mammal {
  // constructor
  public function Cat(gender:String) {
    super(gender);
    speciesName = "Cat";
  }

  public function play():String {
    return "Pounce a ball of yarn.";
  }
}
```

   Observe que está sustituyendo el método `play()` de la superclase Mammal. La clase Cat sólo define dos métodos, un constructor y un método `play()`. Dado que la clase Cat amplía la clase Mammal, la clase Cat hereda los métodos y propiedades de la clase Mammal. Para más información sobre sustitución, consulte "Sustitución de métodos y propiedades" en la página 281.

6. Guarde los cambios en el documento ActionScript.

7. Cree un nuevo documento ActionScript y guárdelo como **Dog.as** en el mismo directorio que los dos archivos de clases anteriores.

8. En Dog.as, escriba el siguiente código ActionScript en la ventana Script:

```
class Dog extends Mammal {
  // constructor
  public function Dog(gender:String) {
    super(gender);
    speciesName = "Dog";
  }

  public function play():String {
    return "Fetch a stick.";
  }
}
```

Observe que la estructura de la clase Dog es muy similar a la de la clase Cat, con la diferencia de que han cambiado algunos valores. Una vez más, la clase Dog amplía la clase Mammal y hereda todos sus métodos y propiedades. El constructor Dog toma una sola propiedad, gender, que pasa a la clase principal de la clase Dog, es decir, Mammal. La variable `speciesName` también se sustituye y se establece con la cadena `Dog`. El método `play()` de la clase principal también se sustituye.

9. Guarde los cambios en el documento ActionScript.
10. Cree otro documento ActionScript en el mismo directorio que los otros archivos y guárdelo como **Monkey.as**.
11. En Monkey.as, escriba el siguiente código ActionScript en la ventana Script:

```
class Monkey extends Mammal {
  // constructor
  public function Monkey(gender:String) {
    super(gender);
    speciesName = "Monkey";
  }

  public function play():String {
    return "Swing from a tree.";
  }
}
```

Al igual que las dos clases anteriores, Cat y Dog, la clase Monkey amplía la clase Mammal. El constructor de la clase Monkey llama al constructor de la clase Mammal y pasa el género (gender) al constructor de Mammal, además de establecer `speciesName` con la cadena `Monkey`. La clase Monkey también sustituye el comportamiento del método `play()`.

12. Guarde los cambios en el documento ActionScript.
13. Ahora que ha creado tres subclases de la clase Mammal, cree un nuevo documento de Flash denominado **mammalTest.fla**.
14. En mammalTest.fla, escriba el siguiente código ActionScript en el fotograma 1 de la línea de tiempo principal:

```
var mammals_arr:Array = new Array();
this.createTextField("info_txt", 10, 10, 10, 450, 80);
info_txt.html = true;
info_txt.multiline = true;
info_txt.border = true;
info_txt.wordWrap = true;

createMammals()
createReport()
```

```
function createMammals():Void {
  mammals_arr.push(new Dog("Female"));
  mammals_arr.push(new Cat("Male"));
  mammals_arr.push(new Monkey("Female"));
  mammals_arr.push(new Mammal("Male"));
}

function createReport():Void {
  var i:Number;
  var len:Number = mammals_arr.length;
  // Mostrar información de Mammal en 4 columnas de texto HMTL
  utilizando tabulaciones.
  info_txt.htmlText = "<textformat tabstops='[110, 200, 300]'>";
  info_txt.htmlText += "<b>Mammal\tGender\tSleep\tPlay</b>";
  for (i = 0; i < len; i++) {
    info_txt.htmlText += "<p>" + mammals_arr[i].speciesName
        + "\t" + mammals_arr[i].gender
        + "\t" + mammals_arr[i].sleep()
        + "\t" + mammals_arr[i].play() + "</p>";
    // La sentencia trace llama al método Mammal.toString().
    trace(mammals_arr[i]);
  }
  info_txt.htmlText += "</textformat>";
}
```

El código mammalTest.fla es un poco más complejo que las clases anteriores. En primer lugar, importa las tres clases de animales.

15. Guarde el documento de Flash y luego seleccione Control > Probar película para probar el documento.

    Verá la información de Mammal en un campo de texto en el escenario y el siguiente texto en el panel Salida:

```
[object Dog]
[object Cat]
[object Monkey]
[object Mammal]
```

CAPÍTULO 8

# Interfaces

8

En programación orientada a objetos (OOP), una interfaz es un documento que le permite declarar (pero no definir) los métodos que deben aparecer dentro de una clase. Al trabajar en un equipo de desarrolladores o crear aplicaciones grandes en Flash, las interfaces pueden ser de gran ayuda durante el desarrollo. Las interfaces permiten a los desarrolladores identificar fácilmente los métodos básicos de las clases de ActionScript. Estos métodos deben implementarse cuando los desarrolladores utilizan cada interfaz.

En este capítulo tendrá la oportunidad de ver varias interfaces de muestra y, al finalizar el capítulo, será capaz de crear sus propios archivos de interfaz. Si no está familiarizado con la creación de clases, lea el Capítulo 6, "Clases", antes de realizar los tutoriales y ejemplos de este capítulo.

Para más información sobre el trabajo con interfaces, consulte los temas siguientes:



## Interfaces

En programación orientada a objetos, las interfaces son como clases cuyos métodos no se implementan (definen), es decir, no "hacen" nada. Por consiguiente, una interfaz consta de métodos "vacíos". Otra clase puede implementar los métodos declarados por la interfaz. En ActionScript, la distinción entre interfaz y objeto sólo se aplica en la comprobación de errores en tiempo de compilación y en el cumplimiento de las reglas del lenguaje.

Una interfaz no es una clase; sin embargo, esto no es del todo cierto en ActionScript en tiempo de ejecución, ya que la interfaz es abstracta. Las interfaces de ActionScript existen de hecho en tiempo de ejecución para permitir la conversión de tipos (cambiar un tipo de datos existente por otro distinto). El modelo de objetos de ActionScript 2.0 no admite la herencia múltiple. Por lo tanto, una clase sólo puede heredar de una única clase principal. Esta clase principal puede ser una clase central o de Flash Player o bien una clase definida por el usuario (personalizada). Puede utilizar interfaces para implementar una forma limitada de herencia múltiple, a través de la cual una clase hereda de más de una clase.

Por ejemplo, en C++, la clase Cat podría ampliar la clase Mammal, así como una clase Playful (juguetón), que tiene los métodos `chaseTail` (perseguirCola) y `eatCatNip` (comerFriskies). Al igual que Java, ActionScript 2.0 no permite que una clase se amplíe en múltiples clases directamente, pero sí permite que una clase amplíe una sola clase e implemente múltiples interfaces. Por consiguiente, podría crear una interfaz Playful que declarara los métodos `chaseTail()` y `eatCatNip()`. Una clase Cat, o cualquier otra clase, podría implementar esta interfaz y proporcionar definiciones para dichos métodos.

Una interfaz también puede considerarse como un "contrato de programación" que puede utilizarse para aplicar relaciones entre clases que de otro modo no estarían relacionadas. Por ejemplo, suponga que está trabajando con un equipo de programadores y que cada uno de ellos está trabajando en una clase distinta dentro de la misma aplicación. Al diseñar la aplicación, se acordaron una serie de métodos que utilizan las distintas clases para comunicarse. Puede crear una interfaz que declare estos métodos, sus parámetros y sus tipos de devolución. Todas las clases que implementen esta interfaz deben proporcionar definiciones de dichos métodos; de lo contrario, se produce un error en el compilador. La interfaz es como un protocolo de comunicación que deben cumplir todas las clases.

Una forma de llevar esto a cabo es crear una clase que defina todos estos métodos y, a continuación, hacer que cada clase amplíe o herede de esta superclase. Sin embargo, dado que la aplicación consta de clases que no están relacionadas entre sí, no tiene ningún sentido organizarlas en una jerarquía de clases común. Es mejor crear una interfaz que declare los métodos que estas clases utilizarán para comunicarse y después hacer que cada clase implemente (proporcione sus propias definiciones para) estos métodos.

En general se puede programar correctamente sin utilizar interfaces. Sin embargo, cuando se utilizan adecuadamente, las interfaces pueden hacer que el diseño de las aplicaciones sea más elegante, escalable y sostenible.

Las interfaces de ActionScript existen en tiempo de ejecución para permitir la conversión de tipos; consulte Capítulo 10, “Conversión de objetos”, en la página 370. Una interfaz no es un objeto ni una clase, pero el flujo de trabajo es similar al que se utiliza al trabajar con clases. Para más información sobre el flujo de trabajo de clases, consulte “Escritura de archivos de clases personalizadas” en la página 205. Para ver un tutorial sobre la creación de una aplicación con interfaces, consulte “Ejemplo: Utilización de interfaces” en la página 298.

Para más información sobre el uso de interfaces, consulte las secciones siguientes:



## La palabra clave interface

La palabra clave `interface` define una interfaz. Una interfaz es similar a una clase, pero presenta estas diferencias importantes:

- Las interfaces sólo contienen declaraciones de métodos, no su implementación. Es decir, cada clase que implementa una interfaz debe proporcionar una implementación para cada método declarado en la interfaz.
- Sólo se permiten miembros públicos en una definición de interfaz; no se permiten miembros estáticos ni de clases.
- En las definiciones de interfaces no se permiten las sentencias `get` ni `set`.
- Para utilizar la palabra clave `interface`, debe especificar ActionScript 2.0 y Flash Player 6 o posterior en la ficha Flash del cuadro de diálogo Configuración de publicación del archivo FLA.

La palabra clave `interface` sólo se admite cuando se utiliza en archivos de script externos y no en scripts escritos en el panel Acciones.

## Asignación de nombre a las interfaces

Los nombres de interfaces empiezan por una letra mayúscula, al igual que los nombres de clase. Los nombres de interfaces suelen ser adjetivos como, por ejemplo, `Printable`. El nombre de interfaz `IEmployeeRecords` empieza con mayúscula y contiene palabras concatenadas donde se mezclan mayúsculas y minúsculas:

```
interface IEmployeeRecords {}
```

| NOTA | Algunos desarrolladores empiezan los nombres de interfaces con una **"I"** mayúscula para distinguirlas de las clases. Se trata de un hábito recomendable, ya que le permite distinguir rápidamente entre interfaces y clases normales. |
|---|---|

Para más información sobre convenciones de asignación de nombre, consulte el Capítulo 19, "Recomendaciones y convenciones de codificación para ActionScript 2.0", en la página 775.

## Definición e implementación de interfaces

El proceso de creación de una interfaz es el mismo que para crear una clase. Como ocurre con las clases, sólo puede definir interfaces en archivos de ActionScript externos. Como mínimo, este flujo de trabajo de creación de una interfaz incluye los siguientes pasos:

- Definir una interfaz en un archivo de ActionScript externo
- Guardar el archivo de interfaz en un directorio de rutas de clases designado (una ubicación en donde Flash busque las clases) o en el mismo directorio que el archivo FLA de la aplicación.
- Crear una instancia de la clase en otro script, ya sea en un documento de Flash (FLA) o en un archivo de script externo, o subinterfaces basadas en la interfaz original
- Crear una clase que implemente la interfaz en un archivo de script externo

Debe declarar una interfaz con la palabra clave `interface`, seguida del nombre de la interfaz y, a continuación, llaves de apertura y cierre (`{}`) que definen el cuerpo de la interfaz, como se muestra en el siguiente ejemplo:

```
interface IEmployeeRecords {
  // declaraciones del método interface
}
```

Una interfaz sólo puede contener declaraciones de método (función), que incluyen parámetros, tipos de parámetros y tipos de devolución de función.

Para más información sobre convenciones de estructuración de clases e interfaces, consulte el Capítulo 19, "Recomendaciones y convenciones de codificación para ActionScript 2.0", en la página 775. Para ver un tutorial sobre la creación de una aplicación que utiliza interfaces, consulte "Ejemplo: Utilización de interfaces" en la página 298.

Por ejemplo, el siguiente código declara una interfaz denominada `IMyInterface` que contiene dos métodos, `method1()` y `method2()`. El primer método, `method1()`, no tiene parámetros y especifica el tipo de devolución Void (lo que significa que no devuelve ningún valor). El segundo método, `method2()`, tiene un solo parámetro de tipo String (cadena) y especifica el tipo de devolución Boolean.

### Para crear una interfaz sencilla:

1. Cree un nuevo archivo ActionScript y guárdelo como **IMyInterface.as**.
2. Escriba el siguiente código ActionScript en la ventana Script:

```
interface IMyInterface {
  public function method1():Void;
  public function method2(param:String):Boolean;
}
```

3. Guarde los cambios en el archivo ActionScript.

   Para utilizar la interfaz dentro de una aplicación, deberá crear primero una clase que implemente la nueva interfaz.
4. Cree un nuevo archivo ActionScript y guárdelo como **MyClass.as** en el mismo directorio que IMyInterface.as.
5. En el archivo de la clase MyClass, escriba el siguiente código ActionScript en la ventana Script:

```
class MyClass {
}
```

   Para indicar a la clase personalizada (MyClass) que utilice la interfaz (IMyInterface), deberá utilizar la palabra clave `implements`, que especifica que una clase debe definir todos los métodos declarados en la interfaz (o interfaces) que implemente.
6. Modifique el código ActionScript de MyClass.as (añada el código en negrita) para que coincida con el siguiente fragmento:

```
class MyClass implements IMyInterface {
}
```

   La palabra clave `implements` debe situarse después del nombre de la clase.
7. Haga clic en el botón Revisar sintaxis.

   Flash muestra un error en el panel Salida que indica que MyClass debe implementar el método *X* de la interfaz IMyInterface. Este mensaje de error aparece porque toda clase que amplíe una interfaz debe definir cada uno de los métodos enumerados en el documento de la interfaz.

8. Modifique de nuevo el documento MyClass (añada el código en negrita) y escriba el código ActionScript para los métodos `method1()` y `method2()`, como se muestra en el siguiente fragmento:

```
class MyClass implements IMyInterface {
  public function method1():Void {
    // ...
  };
  public function method2(param:String):Boolean {
    // ...
    return true;
  }
}
```

9. Guarde el documento MyClass.as y haga clic en Revisar sintaxis.

   El panel Salida ya no muestra ningún mensaje de error ni ninguna advertencia porque ha definido los dos métodos.

El archivo de clase que crea no está limitado a los métodos públicos que usted define en el archivo de interfaz. El archivo de interfaz sólo esboza los métodos mínimos que debe implementar, además de las propiedades de dichos métodos y de los tipos de devolución. Las clases que implementan una interfaz concreta casi siempre incluyen métodos, variables y métodos getter y setter adicionales.

Los archivos de interfaz no pueden contener asignaciones ni declaraciones de variables. Las funciones declaradas en una interfaz no pueden contener llaves. Por ejemplo, la siguiente interfaz no se compila:

```
interface IBadInterface {
  // Error de compilador, no se permite especificar declaraciones de
  variables en las interfaces.
  public var illegalVar:String;

  // Error de compilador, no se permite especificar cuerpos de función en
  las interfaces.
  public function illegalMethod():Void {
  }

  // Error de compilador, Las interfaces no admiten métodos privados.
  private function illegalPrivateMethod():Void;

  // Error de compilador, Las interfaces no admiten getters/setters.
  public function get illegalGetter():String;
}
```

Para ver un tutorial que muestra cómo crear una interfaz compleja, consulte "Ejemplo: Utilización de interfaces" en la página 298.

Las reglas para asignar nombres a interfaces y para almacenarlas en paquetes son las mismas que para las clases; consulte "Asignación de nombre a los archivos de clases" en la página 236.

# Creación de interfaces como tipos de datos

Al igual que las clases, una interfaz define un nuevo tipo de datos. Puede decirse que cualquier clase que implemente una interfaz es del tipo definido por la interfaz. Esto resulta útil para determinar si un objeto dado implementa una interfaz concreta. Tomemos, por ejemplo, la interfaz `IMovable`, que creará en el siguiente ejemplo.

### Para crear una interfaz como tipo de dato:

1. Cree un nuevo documento ActionScript y guárdelo en el disco duro como **IMovable.as**.
2. En IMovable.as, escriba el siguiente código ActionScript en la ventana Script:

```
interface IMovable {
  public function moveUp():Void;
  public function moveDown():Void;
}
```

3. Guarde los cambios en el archivo ActionScript.
4. Cree un nuevo documento ActionScript y guárdelo como **Box.as** en el mismo directorio que IMovable.as.

   En este documento, deberá crear una clase Box que implemente la interfaz IMovable creada en el paso anterior.
5. En Box.as, escriba el siguiente código ActionScript en la ventana Script:

```
class Box implements IMovable {
  public var xPos:Number;
  public var yPos:Number;

  public function Box() {
  }

  public function moveUp():Void {
    trace("moving up");
    // definición de método
  }
  public function moveDown():Void {
    trace("moving down");
    // definición de método
  }
}
```

6. Guarde los cambios en el documento ActionScript.
7. Cree un nuevo documento de Flash llamado **boxTest.fla** y guárdelo en el mismo directorio que los dos documentos ActionScript anteriores.

8. Seleccione el fotograma 1 de la línea de tiempo, abra el Editor de ActionScript y luego escriba el siguiente código ActionScript en el panel Acciones (o la ventana Script):

```
var newBox:Box = new Box();
```

Este código ActionScript crea una instancia de la clase box, que debe declarar como variable del tipo Box.

9. Guarde los cambios en el documento de Flash y luego seleccione Control > Probar película para probar el archivo SWF.

En Flash Player 7 y posteriores, puede convertir una expresión en un tipo de interfaz u otro tipo de datos durante la ejecución. A diferencia de las interfaces Java, las interfaces ActionScript existen en tiempo de ejecución, lo que permite la conversión de tipos. Si la expresión es un objeto que implementa la interfaz o tiene una superclase que implementa la interfaz, se devuelve el objeto. De lo contrario, se devuelve `null`. Esto es útil para asegurarse de que un objeto concreto implemente una interfaz determinada. Para más información sobre la conversión de tipos, consulte el Capítulo 10, “Conversión de objetos”, en la página 370.

10. Añada el siguiente código al final del código ActionScript de boxTest.fla:

```
if (IMovable(newBox) != null) {
  newBox.moveUp();
} else {
  trace("box instance is not movable");
}
```

Este código ActionScript comprueba si la instancia `newBox` implementa la interfaz IMovable antes de llamar al método `moveUp()` en el objeto.

11. Guarde el documento de Flash y luego seleccione Control > Probar película para probar el archivo SWF.

Dado que la instancia Box implementa la interfaz IMovable, se llama al método `Box.moveUp()` y aparece el texto “moving up” (desplazando hacia arriba) en el panel Salida.

Para más información sobre conversiones, consulte el Capítulo 10, “Conversión de objetos”, en la página 370.

# Herencia e interfaces

Puede utilizar la palabra clave `extends` para crear subclases de una interfaz. Esto puede ser de gran utilidad en proyectos grandes en los que desee ampliar (o *crear subclases de*) una interfaz existente y añadir métodos adicionales. Todas las clases que implementen esa interfaz deberán definir estos métodos.

Un aspecto que debe tener en cuenta al ampliar interfaces es que recibirá mensajes de error en Flash si varios archivos de interfaz declaran funciones con los mismos nombres pero tienen parámetros o tipos de devolución diferentes.

En el siguiente ejemplo se muestra cómo pueden crearse subclases de un archivo de interfaz empleando la palabra clave `extends`.

### Para ampliar una interfaz:

1. Cree un nuevo archivo ActionScript y guárdelo como **Ia.as**.
2. En Ia.as, escriba el siguiente código ActionScript en la ventana Script:

```
interface Ia {
  public function f1():Void;
  public function f2():Void;
}
```

3. Guarde los cambios en el archivo ActionScript.
4. Cree un nuevo archivo ActionScript y guárdelo como **Ib.as** en la misma carpeta que el archivo Ia.as creado en el paso 1.
5. En Ib.as, escriba el siguiente código ActionScript en la ventana Script:

```
interface Ib extends Ia {
  public function f8():Void;
  public function f9():Void;
}
```

6. Guarde los cambios en el archivo ActionScript.
7. Cree un nuevo archivo ActionScript y guárdelo como **ClassA.as** en el mismo directorio que los dos archivos anteriores.
8. En ClassA.as, escriba el siguiente código ActionScript en la ventana Script:

```
class ClassA implements Ib {
  // f1() y f2() se definen en la interfaz Ia.
  public function f1():Void {
  }
  public function f2():Void {
  }

  // f8() y f9() se definen en la interfaz Ib, que amplía Ia.
  public function f8():Void {
  }
  public function f9():Void {
  }
}
```

9. Guarde el archivo de clase y haga clic en el botón Revisar sintaxis situado encima de la ventana Script.

   Flash no genera mensajes de error en tanto en cuanto estén definidos los cuatro métodos y coincidan con las definiciones de sus respectivos archivos de interfaz.

| NOTA | Las clases sólo pueden ampliar una clase en ActionScript 2.0, pero puede utilizar clases para implementar tantas interfaces como se desee. |
|---|---|

Si desea que la clase ClassA implemente varias interfaces en el ejemplo anterior, simplemente tendrá que separar las interfaces con comas. O bien, si tuviera una clase que ampliara una superclase e implementara varias interfaces, tendría que utilizar un código similar al siguiente:

```
class ClassA extends ClassB implements Ib, Ic, Id {...}.
```

# Ejemplo: Utilización de interfaces

En este ejemplo, creará una interfaz sencilla que podrá reutilizar en muchas clases diferentes.

### Para crear una interfaz:

1. Cree un nuevo archivo ActionScript y guárdelo como **IDocumentation.as**.
2. En IDocumentation.as, escriba el siguiente código ActionScript en la ventana Script:

```
interface IDocumentation {
  public function downloadUpdates():Void;
  public function checkForUpdates():Boolean;
  public function searchHelp(keyword:String):Array;
}
```

3. Guarde los cambios realizados en el archivo de interfaz de ActionScript.
4. Cree un nuevo archivo ActionScript en el mismo directorio que el archivo IDocumentation.as y guarde este nuevo archivo como **FlashPaper.as.**
5. En FlashPaper.as, escriba el siguiente código ActionScript en la ventana Script:

```
class FlashPaper implements IDocumentation {
}
```

6. Guarde los cambios realizados en el archivo ActionScript.
7. Haga clic en el botón Revisar sintaxis para la clase de ActionScript.

   Se ve un error similar al del siguiente mensaje:

```
**Error** path\FlashPaper.as: Line 1: La clase debe implementar el
  método 'checkForUpdates' de la interfaz 'IDocumentation'.

     class FlashPaper implements IDocumentation {

Total ActionScript Errors: 1  Reported Errors: 1
```

Este error aparece porque la clase FlashPaper actual no define ninguno de los métodos públicos que ha definido en la interfaz IDocumentation.

8. Abra el archivo de clase FlashPaper.as de nuevo y modifique el código ActionScript existente para que coincida con el siguiente código:

```
class FlashPaper implements IDocumentation {
  private static var __version:String = "1,2,3,4";
  public function downloadUpdates():Void {
  };
  public function checkForUpdates():Boolean {
    return true;
  };
  public function searchHelp(keyword:String):Array {
    return []
  };
}
```

9. Guarde los cambios en el archivo ActionScript y luego haga clic de nuevo en Revisar sintaxis.

   En esta ocasión no se ven errores en el panel Salida.

> **NOTA** Puede añadir al archivo de clase FlashPaper tantas variables o métodos estáticos, públicos o privados como desee. El archivo de interfaz sólo define un conjunto mínimo de métodos que deben aparecer en cualquier clase que implemente dicha interfaz.

10. Abra de nuevo el documento de interfaz IDocumentation y añada la siguiente línea de código en negrita (debajo del método `searchHelp()`):

```
interface IDocumentation {
  public function downloadUpdates():Void;
  public function checkForUpdates():Boolean;
  public function searchHelp(keyword:String):Array;
  public function addComment(username:String, comment:String):Void;
}
```

11. Guarde los cambios en el archivo de interfaz y vuelva a abrir el documento FlashPaper.as.

12. Haga clic en el botón Revisar sintaxis y verá un nuevo mensaje de error en el panel Salida:

```
**Error** path\FlashPaper.as: Line 1: La clase debe implementar el
  método 'addComment' de la interfaz 'IDocumentation'.

    class FlashPaper implements IDocumentation {

Total ActionScript Errors: 1  Reported Errors: 1
```

   Se ve el error anterior porque el archivo de clase FlashPaper.as ya no define todas las clases esbozadas en el archivo de interfaz. Para resolver este mensaje de error, deberá añadir el método `addComment()` a la clase FlashPaper o quitar la definición del método del archivo de interfaz IDocumentation.

13. Añada el siguiente método a la clase FlashPaper:

```
public function addComment(username:String, comment:String):Void {
  /* Enviar parámetros a la página del servidor, que inserta el
  comentario en la base de datos. */
}
```

14. Guarde los cambios en FlashPaper.as y haga clic en el botón Revisar sintaxis, tras lo cual no debería aparecer ningún error.

En la sección anterior, hemos creado una clase basada en el archivo de interfaz IDocumentation. En esta sección, crearemos una nueva clase que también implementará la interfaz IDocumentation, aunque añadirá algunos métodos y propiedades adicionales.

En este tutorial se demuestra la utilidad de las interfaces, ya que, si desea crear otra clase que amplíe la interfaz IDocumentation, podrá identificar fácilmente los métodos que deberá contener la nueva clase.

# Ejemplo: Creación de una interfaz compleja

En el siguiente ejemplo se muestran diversas formas de definir e implementar interfaces. En este tutorial, aprenderá a crear un archivo de interfaz sencillo, a escribir una clase que implemente varias interfaces y a hacer que varias interfaces amplíen otras interfaces para crear estructuras de datos más complejas.

### Para crear una interfaz compleja:

1. Cree un nuevo documento ActionScript y guárdelo como **InterfaceA.as**.
2. Cree una nueva carpeta llamada **complexInterface** y guarde en ella InterfaceA.as.

   Debe guardar en este directorio todos los archivos que cree para este tutorial.
3. En Interface.as, escriba el siguiente código ActionScript en la ventana Script:

```
// nombre de archivo: InterfaceA.as
interface InterfaceA {
  public function k():Number;
  public function n(z:Number):Number;
}
```

4. Guarde el documento ActionScript, cree un nuevo documento ActionScript denominado **ClassB.as** y guárdelo en el directorio complexInterface.

   ClassB.as implementa la interfaz InterfaceA creada anteriormente.

5. En ClassB.as, escriba el siguiente código ActionScript en la ventana Script:

```
// nombre de archivo: ClassB.as
class ClassB implements InterfaceA {
  public function k():Number {
    return 25;
  }
  public function n(z:Number):Number {
    return (z + 5);
  }
}
```

6. Guarde los cambios en el documento ClassB.as, cree un nuevo documento de Flash y guárdelo como **classbTest.fla** en el directorio complexInterface.

   Este archivo de clase prueba la clase ClassB creada anteriormente.

7. In classbTest.fla, escriba el siguiente código ActionScript en el fotograma 1 de la línea de tiempo:

```
// nombre de archivo: classbTest.fla
import ClassB;
var myB:ClassB = new ClassB();
trace(myB.k()); // 25
trace(myB.n(7)); // 12
```

8. Guarde los cambios en el documento de Flash y luego seleccione Control > Probar película para probar el documento de Flash.

   El panel Salida muestra dos números, 25 y 12, que son los resultados de los métodos `k()` y `n()` de la clase ClassB.

9. Cree un nuevo archivo ActionScript y guárdelo como **ClassC.as** en el directorio complexInterface.

   Este archivo de clase implementa la interfaz InterfaceA creada en el paso 1.

10. En ClassC.as, escriba el siguiente código ActionScript en la ventana Script:

```
// nombre de archivo: ClassC.as
class ClassC implements InterfaceA {
  public function k():Number {
    return 25;
  }
  // **Error** La clase debe también implementar el método 'n' de la
  interfaz 'InterfaceA'.
}
```

    Si hace clic en el botón Revisar sintaxis para el archivo de clase ClassC, Flash muestra un mensaje de error en el panel Salida que indica que la clase actual debe implementar el método `n()` en la interfaz InterfaceA. Al crear clases que implementen una interfaz, es importante definir métodos para cada entrada de la interfaz.

11. Cree un nuevo documento ActionScript y guárdelo como **InterfaceB.as** en el directorio complexInterface.

12. En InterfaceB.as, escriba el siguiente código ActionScript en la ventana Script:

```
// nombre de archivo: InterfaceB.as
interface InterfaceB {
  public function o():Void;
}
```

13. Guarde los cambios en el documento InterfaceB.as, cree un nuevo documento ActionScript y guárdelo en el directorio complexInterface con el nombre **ClassD.as**.

    Esta clase implementa la interfaz InterfaceA y la interfaz InterfaceB creadas en pasos anteriores. La clase ClassD debe incluir implementaciones de método por cada uno de los métodos enumerados en cada uno de los archivos de interfaz.

14. En ClassD.as, escriba el siguiente código ActionScript en la ventana Script:

```
// nombre de archivo: ClassD.as
class ClassD implements InterfaceA, InterfaceB {
  public function k():Number {
    return 15;
  }
  public function n(z:Number):Number {
    return (z * z);
  }
  public function o():Void {
    trace("o");
  }
}
```

15. Guarde los cambios en el archivo ClassD.as, cree un nuevo documento de Flash y guárdelo como **classdTest.fla**.

    El documento de Flash probará la clase ClassD creada anteriormente.

16. In classdTest.fla, añada el siguiente código ActionScript en el fotograma 1 de la línea de tiempo:

```
// nombre de archivo: classdTest.fla
import ClassD;
var myD:ClassD = new ClassD();
trace(myD.k()); // 15
trace(myD.n(7)); // 49
myD.o(); // o
```

17. Guarde los cambios en el archivo classdTest.fla y luego seleccione Control > Probar película para probar el archivo.

    Deberán aparecer los valores 15, 49 y la letra en el panel Salida. Estos valores son el resultado del método `ClassD.k()`, los métodos `ClassD.n()` y `ClassD.o()`, respectivamente.

18. Cree un nuevo documento ActionScript y guárdelo como **InterfaceC.as**.

    Esta interface amplía la interfaz InterfaceA creada anteriormente y añade una nueva definición de método.

19. En InterfaceC.as, escriba el siguiente código ActionScript en la ventana Script:

```
// nombre de archivo: InterfaceC.as
interface InterfaceC extends InterfaceA {
  public function p():Void;
}
```

20. Guarde los cambios en el archivo ActionScript, cree un nuevo archivo ActionScript y guárdelo como **ClassE.as** en el directorio complexInterface.

    Esta clase implementa dos interfaces, InterfaceB e InterfaceC.

21. En ClassE.as, escriba el siguiente código ActionScript en la ventana Script:

```
// nombre de archivo: ClassE.as
class ClassE implements InterfaceB, InterfaceC {
  public function k():Number {
    return 15;
  }
  public function n(z:Number):Number {
    return (z + 5);
  }
  public function o():Void {
    trace("o");
  }
  public function p():Void {
    trace("p");
  }
}
```

22. Guarde los cambios en el documento ActionScript, cree un nuevo documento de Flash y guárdelo como **classeTest.fla** en el directorio complexInterface.

**23.** En classeTest.fla, escriba el siguiente código ActionScript en el fotograma 1 de la línea de tiempo:

```
// nombre de archivo: classeTest.fla
import ClassE;
var myE:ClassE = new ClassE();
trace(myE.k()); // 15
trace(myE.n(7)); // 12
myE.o(); // o
myE.p(); // p
```

**24.** Guarde el documento de Flash y luego seleccione Control > Probar película para probar el archivo SWF.

Aparecerán los valores `15`, `12`, `o` y `p` en el panel Salida. Estos son los valores devueltos por los métodos `ClassE.k()`, `ClassE.n()`, `ClassE.o()` y `ClassE.p()`. Dado que la clase ClassE implementa las interfaces InterfaceB y InterfaceC, deberá definirse cada método de los dos archivos de interfaz. Aunque las interfaces InterfaceB y InterfaceC sólo definen los métodos `o()` y `p()`, InterfaceC amplía InterfaceA. Esto significa que todos los métodos definidos que incluye, `k()` y `n()`, también deben implementarse.

CAPÍTULO 9

# Gestión de eventos

9

*Eventos*: acciones que tienen lugar durante la reproducción de un archivo SWF. Un evento como hacer clic con el ratón o presionar una tecla se denomina *evento de usuario*, puesto que es una consecuencia directa de una acción del usuario. Un evento generado automáticamente por Flash Player, como la aparición inicial de un clip de película en el escenario, se denomina *evento del sistema* porque no lo genera directamente el usuario.

A fin de que la aplicación reaccione ante los eventos, debe utilizar *controladores de eventos*, es decir, código ActionScript asociado con un objeto y un evento determinados. Por ejemplo, si un usuario hace clic en un botón del escenario, se podría avanzar la cabeza lectora hasta el siguiente fotograma. O bien, al finalizar la carga de un archivo XML por la red, el contenido de dicho archivo podría aparecer en un campo de texto.

Puede gestionar eventos en ActionScript de diversas formas:

- "Utilización de métodos de controlador de eventos" en la página 306
- "Utilización de detectores de eventos" en la página 308
- "Utilización de controladores de eventos de botones y de clips de película" en la página 313, específicamente, %{on handler}% y %{onClipEvent handler}%.
- "Difusión de eventos desde instancias de componentes" en la página 318

La utilización de controladores de eventos con %{loadMovie (método MovieClip.loadMovie)}% puede producir resultados impredecibles. Si asocia un controlador de eventos a un botón mediante `on()`, o si crea un controlador dinámico empleando un método de controlador de eventos como %{onPress (MovieClip.onPress handler)}% y posteriormente efectúa una llamada a `loadMovie()`, el controlador de eventos no estará disponible después de cargarse el nuevo contenido. No obstante, si utiliza %{onClipEvent handler}% o %{on handler}% para asociar un controlador de eventos a un clip de película y luego efectúa una llamada a `loadMovie()` en dicho clip de película, el controlador de eventos continuará estando disponible después de cargarse el nuevo contenido.

Para más información sobre la gestión de eventos, consulte las secciones siguientes:

# Utilización de métodos de controlador de eventos

Un método de controlador de eventos es un método de clase que se invoca cuando se produce un evento en una instancia de dicha clase. Por ejemplo, la clase MovieClip define un controlador de eventos `onPress` que se invoca cada vez que se presiona el ratón en un objeto de clip de película. A diferencia del resto de los métodos de una clase, un controlador de eventos no se invoca directamente; Flash Player lo invoca automáticamente cuando se produce el evento pertinente.

Las siguientes clases de ActionScript son ejemplos de clases que definen controladores de eventos: Button, ContextMenu, ContextMenuItem, Key, LoadVars, LocalConnection, Mouse, MovieClip, MovieClipLoader, Selection, SharedObject, Sound, Stage, TextField, XML y XMLSocket. Para más información sobre los controladores de eventos que proporcionan estas clases, consulte las entradas correspondientes a estas clases en *Referencia del lenguaje ActionScript 2.0*. La palabra *controlador* se añade al título de cada controlador de eventos.

De forma predeterminada, los métodos de controlador de eventos no están definidos: cuando se produce un evento determinado, el controlador de eventos correspondiente se invoca, pero la aplicación no responde de ninguna otra forma al evento. Para que la aplicación responda al evento, se debe definir una función con la sentencia function y después asignarla al controlador de eventos que corresponda. La función que se asigne al controlador de eventos se invocará automáticamente siempre que se produzca el evento.

Un controlador de eventos se compone de tres partes: el objeto al que se aplica el evento, el nombre del método del controlador de eventos del objeto y la función que se asigna al controlador. En el ejemplo siguiente se muestra la estructura básica de un controlador de eventos:

```
object.eventMethod = function () {
  // El código se escribe aquí, en respuesta al evento.
}
```

Por ejemplo, imagine que en el escenario hay un botón denominado next_btn. El código siguiente asigna una función al controlador de eventos `onPress` del botón; esta función hace que la cabeza lectora avance hasta el siguiente fotograma de la línea de tiempo actual:

```
next_btn.onPress = function () {
  nextFrame();
}
```

**Asignación de la referencia de una función** En el código anterior, la función `nextFrame()` se asigna a un controlador de eventos para `onPress`. También se puede asignar una referencia de función (nombre) a un método de controlador de eventos y definir la función posteriormente, como se muestra en el siguiente ejemplo:

```
// Asignar una referencia de función al controlador de eventos onPress del
  botón.
next_btn.onPress = goNextFrame;

// Definir la función goNextFrame().
function goNextFrame() {
  nextFrame();
}
```

Observe en el siguiente ejemplo que se asigna una referencia de función, no el valor devuelto por la función, al controlador de eventos `onPress`.

```
// Incorrecto
next_btn.onPress = goNextFrame();
// Correcto.
 next_btn.onPress = goNextFrame;
```

**Recepción de parámetros pasados** Algunos controladores de eventos reciben parámetros pasados que proporcionan información sobre el evento que se ha producido. Por ejemplo, el controlador de eventos `TextField.onSetFocus` se invoca cuando una instancia de campo de texto se selecciona con el teclado. Este controlador de eventos recibe una referencia al objeto de campo de texto que anteriormente estaba seleccionado con el teclado.

Por ejemplo, el código siguiente inserta texto en un campo de texto que ya no está seleccionado con el teclado:

```
this.createTextField("my_txt", 99, 10, 10, 200, 20);
my_txt.border = true;
my_txt.type = "input";
this.createTextField("myOther_txt", 100, 10, 50, 200, 20);
myOther_txt.border = true;
myOther_txt.type = "input";
myOther_txt.onSetFocus = function(my_txt:TextField) {
  my_txt.text = "I just lost keyboard focus";
};
```

**Controladores de eventos para los objetos de tiempo de ejecución** También se pueden asignar funciones a los controladores de eventos para los objetos que crea el usuario durante la ejecución. Por ejemplo, el código siguiente crea una nueva instancia de clip de película (`newclip_mc`) y después asigna una función al controlador de eventos `onPress` del clip:

```
this.attachMovie("symbolID", "newclip_mc", 10);
newclip_mc.onPress = function () {
  trace("You pressed me");
}
```

Para más información, consulte "Creación de clips de película en tiempo de ejecución" en la página 383.

**Sustitución de métodos de controlador de eventos** Mediante la creación de una clase que amplíe una clase de ActionScript, podrá sustituir métodos de controlador de eventos con las funciones que escriba. Puede definir un controlador de eventos en una nueva subclase que posteriormente podrá reutilizar para diversos objetos vinculando cualquier símbolo de la biblioteca de la clase ampliada a la nueva subclase. El siguiente código sustituye el controlador de eventos `onPress` de la clase MovieClip con una función que disminuye la transparencia del clip de película:

```
// Clase FadeAlpha -- establece la transparencia al hacer clic en el clip de
  película.
class FadeAlpha extends MovieClip {
  function onPress() {
    this._alpha -= 10;
  }
}
```

Para obtener instrucciones concretas sobre la ampliación de una clase de ActionScript y la vinculación con un símbolo en la biblioteca, consulte los ejemplos incluidos en "Asignación de una clase a símbolos en Flash" en la página 251. Para obtener información sobre la escritura y la utilización de clases personalizadas, consulte el Capítulo 6, "Clases".

# Utilización de detectores de eventos

Los detectores de eventos permiten que un objeto, denominado *objeto detector*, reciba eventos difundidos por otro objeto, denominado *objeto difusor*. El objeto difusor registra el objeto detector que va a recibir los eventos generados por el difusor. Por ejemplo, se puede registrar un clip de película para que reciba notificaciones `onResize` desde el escenario o que una instancia de botón reciba notificaciones `onChanged` de un objeto de campo de texto. Se pueden registrar varios objetos detectores para que reciban eventos de un único difusor y se puede registrar un único objeto detector para que reciba eventos de varios difusores.

El modelo detector-difusor para eventos, a diferencia de los métodos de controlador, permite que varios fragmentos de código detecten el mismo evento sin que se produzca ningún conflicto. Los modelos de evento que no utilizan el modelo detector/difusor, como `XML.onLoad()`, por ejemplo, pueden resultar problemáticos cuando varios fragmentos de código detectan el mismo evento; los diferentes fragmentos de código entran en conflicto por el control de dicha referencia única a la función callback `XML.onLoad`. Con el modelo de detector/difusor, pueden añadirse fácilmente detectores del mismo evento sin que ello provoque conflictos de código.

Las siguientes clases de ActionScript pueden difundir eventos: %{Key}%, %{Mouse}%, %{MovieClipLoader}%, %{Selection}%, %{Stage}% y %{TextField}%. Para comprobar los detectores que están disponibles para una clase, consulte la entrada correspondiente en *Referencia del lenguaje ActionScript 2.0*.

Para más información sobre detectores de eventos, consulte los siguientes temas:

- "Modelo de detector de eventos" en la página 309
- "Ejemplo de detector de eventos" en la página 310

La clase Escenario puede difundir eventos. El archivo de origen de muestra, stagesize.fla, se puede encontrar en la carpeta Samples del disco duro. En este ejemplo se muestra cómo la propiedad `Stage.scaleMode` afecta a los valores de `Stage.width` y `Stage.height` cuando se modifica el tamaño del navegador.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\StageSize.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/StageSize.

## Modelo de detector de eventos

El modelo de eventos de los detectores de eventos es muy parecido al modelo de los controladores de eventos (consulte "Utilización de métodos de controlador de eventos" en la página 306), aunque presenta dos diferencias fundamentales:

- Puede asignar el controlador de eventos al objeto detector, no al objeto que difunde el evento.
- Se llama a un método especial del objeto difusor, `addListener()`, que registra el objeto detector para que reciba sus eventos.

En el código siguiente se aprecia el modelo del detector de eventos:

```
var listenerObject:Object = new Object();
listenerObject.eventName = function(eventObj:Object) {
  // El código se escribe aquí
};
broadcasterObject.addListener(listenerObject);
```

El código comienza con un objeto, *listenerObject*, con una propiedad *eventName*. El objeto detector puede ser cualquier objeto, como un objeto ya existente, un clip de película o una instancia de botón del escenario, o bien una instancia de cualquiera de las clases de ActionScript. Por ejemplo, un clip de película personalizado puede implementar métodos de detector para detectores del escenario. Puede incluso tener un objeto que detecte diversos tipos de detectores.

La propiedad *eventName* es un evento que se produce en *broadcasterObject*, que a su vez difunde el evento a *listenerObject*. Se pueden registrar varios detectores de eventos para un único difusor de eventos.

Se asigna una función al detector de eventos que responde de alguna manera al evento.

Finalmente, se llama al método `addListener()` en el objeto difusor, al que se pasa el objeto detector al método addListener().

Para anular el registro de un objeto detector de forma que deje de recibir eventos, debe llamar al método `removeEventListener()` del objeto difusor y pasarle el nombre del evento que se va a eliminar y el objeto difusor.

```
broadcasterObject.removeListener(listenerObject);
```

## Ejemplo de detector de eventos

En el ejemplo siguiente se muestra cómo utilizar el detector de eventos `onSetFocus` en la clase Selection para crear un gestor de selección simple para un grupo de campos de introducción de texto. En este caso, el borde del campo de texto que se selecciona con el teclado se activa (aparece), mientras que el borde del texto que no está seleccionado se desactiva.

### Para crear un gestor de selección simple con detectores de eventos:

1. Con la herramienta Texto, cree un campo de texto en el escenario.
2. Seleccione el campo de texto y, en el inspector de propiedades, elija Introducción de texto en el menú emergente Tipo de texto y seleccione la opción Mostrar borde alrededor del texto.
3. Cree otro campo de introducción de texto debajo del primero.

   Compruebe que la opción Mostrar borde alrededor del texto no esté seleccionada para este campo de texto. Puede continuar con la creación de campos de entrada de texto.

4. Seleccione el fotograma 1 en la línea de tiempo y abra el panel Acciones (Ventana > Acciones).
5. Para crear un objeto que detecte la notificación de selección de la clase Selection, introduzca el código siguiente en el panel Acciones:

```
// Crea un objeto detector, focusListener.
var focusListener:Object = new Object();
// Define la función para el objeto detector.
focusListener.onSetFocus = function(oldFocus_txt:TextField,
  newFocus_txt:TextField) {
  oldFocus_txt.border = false;
  newFocus_txt.border = true;
}
```

   Este código crea un objeto llamado `focusListener` que define una propiedad `onSetFocus` y asigna una función a la propiedad. La función está formada por dos parámetros: una referencia al campo de texto que no está seleccionado y una referencia al campo de texto que está seleccionado. La función define la propiedad `border` del campo de texto que no está seleccionado con el valor `false` y define la propiedad `border` del campo que está seleccionado con el valor `true`.
6. Para registrar el objeto `focusListener` para que reciba eventos del objeto Selection, añada el código siguiente al panel Acciones:

```
// Registra focusListener con el difusor.
Selection.addListener(focusListener);
```

7. Pruebe la aplicación (Control > Probar película), haga clic en el primer campo de texto y presione la tecla Tabulador para pasar la selección de un campo a otro.

# Utilización de detectores de eventos con componentes

Al utilizar componentes, la sintaxis de detectores de eventos cambia ligeramente. Los componentes generan eventos, que se deben detectar específicamente utilizando un objeto detector o una función personalizada.

En el siguiente ejemplo se muestra cómo puede utilizar detectores de eventos para controlar el progreso de la descarga de una imagen cargada dinámicamente.

### Para detectar eventos del componente Loader:

1. Arrastre una instancia del componente Loader del panel Componentes al escenario.
2. Seleccione el cargador y escriba **my_ldr** en el cuadro de texto Nombre de instancia del inspector de propiedades.

3. Añada el código siguiente al fotograma 1 de la línea de tiempo principal;

```
System.security.allowDomain("http://www.helpexamples.com");

var loaderListener:Object = new Object();
loaderListener.progress = function(evt_obj:Object):Void {
  trace(evt_obj.type); // progress
  trace("\t" + evt_obj.target.bytesLoaded + " of " +
  evt_obj.target.bytesTotal + " bytes loaded");
}
loaderListener.complete = function(evt_obj:Object):Void {
  trace(evt_obj.type); // complete
}

my_ldr.addEventListener("progress", loaderListener);
my_ldr.addEventListener("complete", loaderListener);
my_ldr.load("http://www.helpexamples.com/flash/images/image1.jpg");
```

Este código ActionScript define un objeto detector denominado `loaderListener` que detecta dos eventos: `progress` y `complete`. Al distribuirse cada uno de estos eventos, se ejecuta su código y se muestra texto de depuración en el panel Salida (si prueba el archivo SWF en la herramienta de edición).

A continuación, deberá indicar a la instancia `my_ldr` que detecte cada uno de los eventos especificados (`progress` y `complete`) y especificar el objeto detector o la función que debe ejecutarse al distribuirse el evento. Finalmente, se llamará al método `Loader.load()`, que desencadena el inicio de la descarga de la imagen.

4. Seleccione Control > Probar película para probar el archivo SWF.

La imagen se descargará en la instancia Loader del escenario y se mostrarán varios mensajes en el panel Salida. Dependiendo del tamaño de la imagen que descargue y de si se le ha asignado una caché en el sistema local del usuario, el evento `progress` podría distribuirse varias veces, mientras que el evento complete sólo se distribuirá después de que la imagen haya terminado de descargarse.

Al trabajar con componentes y eventos de distribución, la sintaxis varía ligeramente con respecto a la de los detectores de eventos de los ejemplos anteriores. En especial, deberá utilizar el método `addEventListener()` en lugar de llamar a `addListener()`. En segundo lugar, deberá especificar el evento específico que desea detectar, además del objeto o función de detector de eventos.

En lugar de utilizar un objeto detector, como en el primer procedimiento de "Utilización de detectores de eventos con componentes" en la página 311, puede utilizar una función personalizada. El código del ejemplo anterior podría reescribirse de la siguiente forma:

```
System.security.allowDomain("http://www.helpexamples.com");

my_ldr.addEventListener("progress", progressListener);
my_ldr.addEventListener("complete", completeListener);
my_ldr.load("http://www.helpexamples.com/flash/images/image1.png");

function progressListener(evt_obj:Object):Void {
  trace(evt_obj.type); // progress
  trace("\t" + evt_obj.target.bytesLoaded + " of " +
  evt_obj.target.bytesTotal + " bytes loaded");
}
function completeListener(evt_obj:Object):Void {
  trace(evt_obj.type); // complete
}
```

**NOTA**

En los ejemplos anteriores, los detectores de eventos siempre se añaden antes de que se llame al método `Loader.load()`. Si llama al método `Loader.load()` antes de especificar los detectores de eventos, la carga podría concluir antes de que se hayan definido completamente los detectores de eventos. Esto significa que el contenido podría mostrarse sin que se hubiera detectado el evento `complete`.

# Utilización de controladores de eventos de botones y de clips de película

Puede asociar controladores de eventos directamente a una instancia de botón o de clip de película del escenario mediante los controladores de eventos `onClipEvent()` y `on()`. El controlador de eventos `onClipEvent()` difunde los eventos de clip de película y el controlador de eventos `on()` gestiona los eventos de botón.

Para asociar un controlador de eventos a una instancia de botón o clip de película, haga clic en la instancia de botón o clip de película en el escenario para que se seleccione y luego introduzca el código en el panel Acciones. El título del panel Acciones indica si el código se va a asociar al botón o al clip de película: Panel Acciones - Botón o Panel Acciones - Clip de película. Para obtener directrices relativas a la utilización de código asociado a instancias de botones o clips de película, consulte "Asociación de código a los objetos" en la página 791.

**NOTA**

No confunda los controladores de eventos de botón o clip de película con los eventos de componente, como `SimpleButton.click`, `UIObject.hide` y `UIObject.reveal`, que deben asociarse a instancias de componentes y se describen en Utilización de componentes.

Sólo puede asociar `onClipEvent()` y `on()` a instancias de clip de película que se hayan colocado en el escenario durante la edición. No pueden asociar `onClipEvent()` y `on()` a las instancias de clip de película creadas en tiempo de ejecución (con el método `attachMovie()`, por ejemplo). Para asociar controladores de eventos a objetos creados durante la ejecución, utilice los métodos de controlador de eventos o los detectores de eventos. (Consulte "Utilización de métodos de controlador de eventos" en la página 306 y "Utilización de detectores de eventos" en la página 308.)

**NOTA**

Asociar controladores `onClipEvent()` y `on()` no es una práctica recomendada. En su lugar, debería poner el código en scripts de fotograma en un archivo de clase, tal y como se muestra en este manual. Para más información, consulte **"Utilización de métodos de controlador de eventos" en la página 306** y **"Asociación de código a los objetos" en la página 791**.

Para más información sobre controladores de eventos de botones y clips de película, consulte los siguientes temas:

- "Utilización de onClipEvent con métodos de controlador de eventos" en la página 314
- "Especificación de eventos para métodos on o onClipEvent" en la página 316
- "Asociación o asignación de varios controladores a un objeto" en la página 317

## Utilización de onClipEvent con métodos de controlador de eventos

En algunos casos, es posible utilizar diferentes técnicas para gestionar eventos sin que se produzcan conflictos. La utilización de los métodos `on()` y `onClipEvent()` no interfiere con la utilización de métodos de controlador de eventos definidos por el usuario.

Por ejemplo, imagine que hay un botón en un archivo SWF. El botón puede tener un controlador `on(press)`, que indica al archivo SWF que debe empezar a reproducirse, y el mismo botón puede tener un método `onPress()`, para el que se define una función que indica a un objeto del escenario que gire. Cuando se hace clic en el botón, el archivo SWF empieza a reproducirse y el objeto gira. En función del momento o los tipos de eventos que desee invocar, podrá utilizar los métodos `on()` y `onClipEvent()`, métodos de controlador de eventos o ambas técnicas de gestión de eventos.

No obstante, el ámbito de las variables y objetos en los controladores `on()` y `onClipEvent()` es diferente al de los controladores de eventos y los detectores de eventos. Consulte "Ámbito del controlador de eventos" en la página 320.

También puede utilizar `on()` con clips de película para crear clips de película que reciban eventos de botón. Para más información, consulte "Creación de clips de película con estados de botón" en la página 319. Para obtener información sobre la especificación de eventos para `on()` y `onClipEvent()`, consulte "Especificación de eventos para métodos on o onClipEvent" en la página 316.

### Para utilizar un controlador on y un controlador de eventos onPress:

1. Cree un nuevo documento de Flash y guárdelo como **handlers.fla**.
2. Seleccione la herramienta Rectángulo para dibujar un gran cuadrado en el escenario.
3. Seleccione la herramienta Selección, haga doble clic en el cuadrado del escenario y presione F8 para iniciar el cuadro de diálogo Convertir en símbolo.
4. Introduzca un nombre de símbolo para el cuadro, establezca el tipo en Clip de película y haga clic en Aceptar.
5. Asigne al clip de película del escenario el nombre de instancia **box_mc**.
6. Añada el siguiente código ActionScript directamente en el símbolo de clip de película en el escenario:

```
on (press) {
  trace("on (press) {...}");
}
```

7. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
box_mc.onPress = function() {
  trace("box_mc.onPress = function() {...};");
};
```

8. Seleccione Control > Probar película para probar el documento de Flash.

   Al hacer clic en el símbolo de clip de película del escenario, se envía la siguiente salida al panel Salida:

   on (press) {...}

   box_mc.onPress = function() {...};

**NOTA**

Asociar controladores `onClipEvent()` y `on()` no es una práctica recomendada. En su lugar, debería poner el código en scripts de fotograma en un archivo de clase, tal y como se muestra en este manual. Para más información, consulte "Utilización de métodos de controlador de eventos" en la página 306 y "Asociación de código a los objetos" en la página 791.

# Especificación de eventos para métodos on o onClipEvent

Para utilizar un controlador `on()` o `onClipEvent()`, asócielo directamente a una instancia de botón o de clip de película del escenario y especifique el evento que desea gestionar para dicha instancia. Para obtener una lista completa de eventos admitidos por los controladores de eventos `on()` y `onClipEvent()`, consulte `%{on handler}%` y `%{onClipEvent handler}%` en *Referencia del lenguaje ActionScript 2.0*.

Por ejemplo, el controlador de eventos `on()` siguiente se ejecuta siempre que el usuario hace clic en el botón al que se ha asociado el controlador:

```
on (press) {
  trace("Thanks for pressing me.");
}
```

Se pueden especificar dos o más eventos para cada controlador `on()`, separados por comas. El ActionScript de un controlador se ejecuta cuando se produce uno de los eventos especificados por el controlador. Por ejemplo, el controlador `on()` siguiente asociado a un botón se ejecuta cada vez que el ratón se desplaza sobre el botón y luego se sitúa fuera del botón.

```
on (rollOver, rollOut) {
  trace("You rolled over, or rolled out");
}
```

También se pueden añadir eventos de pulsaciones de teclas mediante los controladores `on()`. Por ejemplo, el siguiente código traza una cadena cuando se presiona el número 3 del teclado. Seleccione una instancia de clip de película o botón y añada el siguiente código al panel Acciones:

```
on (keyPress "3") {
  trace("You pressed 3")
}
```

O bien, si desea que se realice el trazo cuando el usuario presione la tecla Intro, podría utilizar el siguiente formato de código. Seleccione una instancia de clip de película o botón y añada el siguiente código al panel Acciones:

```
on (keyPress "<Enter>") {
  trace("Enter Pressed");
}
```

Seleccione Control > Probar película y presione la tecla Intro para vez el trazo de cadena en el panel Salida. Si no se traza nada, seleccione Control > Deshabilitar métodos abreviados de teclado y vuelva a intentarlo. Para más información sobre la adición de la interactividad de la pulsación de teclas a las aplicaciones, consulte `%{Key}%`.

**NOTA**

Asociar controladores `onClipEvent()` y `on()` no es una práctica recomendada. En su lugar, debería poner el código en scripts de fotograma en un archivo de clase, tal y como se muestra en este manual. Para más información, consulte "Utilización de métodos de controlador de eventos" en la página 306 y "Asociación de código a los objetos" en la página 791.

## Asociación o asignación de varios controladores a un objeto

También puede asociar más de un controlador a un objeto si quiere que se ejecuten scripts diferentes cuando se produzcan eventos diferentes. Por ejemplo, puede asociar los controladores `onClipEvent()` siguientes a la misma instancia de clip de película. El primero se ejecuta cuando el clip de película se carga por primera vez (o aparece en el escenario); el segundo se ejecuta cuando el clip de película se descarga del escenario.

```
on (press) {
  this.unloadMovie()
}
onClipEvent (load) {
  trace("I've loaded");
}
onClipEvent (unload) {
  trace("I've unloaded");
}
```

**NOTA**

Asociar controladores `onClipEvent()` y `on()` no es una práctica recomendada. En su lugar, debería poner el código en scripts de fotograma en un archivo de clase, tal y como se muestra en este manual. Para más información, consulte "Utilización de métodos de controlador de eventos" en la página 306 y "Asociación de código a los objetos" en la página 791.

Para asociar varios controladores a un objeto mediante el código que se ha colocado en la línea de tiempo, consulte el siguiente ejemplo. El código asocia los controladores `onPress` y `onRelease` a una instancia de clip de película.

**Para asignar varios controladores a un objeto:**

1. Cree un nuevo documento de Flash y asígnele el nombre **assignMulti.fla**.
2. Seleccione el fotograma 1 de la línea de tiempo y añada el siguiente código en el panel Acciones:

```
this.createEmptyMovieClip("img_mc", 10);
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
     target_mc.onPress = function() {
          target_mc.startDrag();
     };
     target_mc.onRelease = function() {
          target_mc.stopDrag();
     };
}
mclListener.onLoadError = function(target_mc:MovieClip) {
     trace("error downloading image");
}
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mclListener);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img_mc);
```

3. Seleccione Control > Probar película para probar el documento.

   La imagen se carga en la instancia `img_mc` y los controladores de eventos `onPress()` y `onRelease()` permiten arrastrar la imagen alrededor del escenario.

# Difusión de eventos desde instancias de componentes

Puede especificar el modo en que debe gestionarse un evento para cualquier instancia de componente. Los eventos de componentes se gestionan de forma distinta a los eventos difundidos desde objetos nativos de ActionScript.

Para más información, consulte “Gestión de eventos de componentes” en *Utilización de componentes*.

# Creación de clips de película con estados de botón

Cuando se asocia un controlador `on()` a un clip de película o se asigna una función a uno de los controladores de eventos de ratón MovieClip para una instancia de clip de película, el clip de película responde a los eventos de ratón del mismo modo que un botón. También se pueden crear estados de botón automáticos (Arriba, Sobre y Abajo) en un clip de película añadiendo las etiquetas de fotograma `_up`, `_over` y `_down` a la línea de tiempo del clip de película.

Cuando el usuario desplaza el ratón por el clip de película o hace clic en él, la cabeza lectora se desplaza al fotograma que lleva la etiqueta de fotograma pertinente. Para designar el área activa que utiliza un clip de película, use la propiedad %{hitArea (propiedad MovieClip.hitArea)}%.

### Para crear estados de botón en un clip de película:

1. Cree un nuevo documento de Flash y guárdelo como **mcbutton.fla**.
2. Con la herramienta Rectángulo, arrastre un rectángulo pequeño (aproximadamente 100 píxeles de ancho por 20 píxeles de alto) al escenario.
3. Haga doble clic en la forma con la herramienta Selección y presione F8 para iniciar el cuadro de diálogo Convertir en símbolo.
4. Introduzca el nombre de símbolo **mcbutton**, establezca el tipo de símbolo en clip de película y haga clic en Aceptar.
5. Haga doble clic en el símbolo de clip de película en el escenario para introducir el módulo de edición de símbolos.
6. Cree una nueva capa en la línea de tiempo del clip de película y cambie el nombre de la nueva capa a **labels**.
7. Introduzca la etiqueta de fotograma `_up` en el inspector de propiedades.
8. Cree una nueva capa sobre la capa predeterminada y la capa labels.
9. Cambie la nueva capa **actions** y añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo del clip de película:

   ```
   stop();
   ```

10. Seleccione el fotograma 10 de las tres capas y, a continuación, seleccione Insertar > Línea de tiempo > Fotograma clave.
11. Añada una acción `stop()` en el fotograma 10 de la capa actions y una etiqueta de fotograma **_over** en el fotograma 10 de la capa layer.

12. Seleccione el rectángulo en el fotograma 10 y utilice el inspector de propiedades para seleccionar otro color de relleno.
13. Cree nuevos fotogramas clave en el fotograma 20 para cada una de las tres capas y añada una etiqueta de fotograma **_down** en el inspector de propiedades.
14. Modifique el color del rectángulo en el fotograma 20 de forma que cada uno de los tres estados de botón tenga un color diferente.
15. Vuelva a la línea de tiempo principal.
16. Para que el clip de película responda a eventos de ratón, realice una de las acciones siguientes:
    - Asocie un controlador de eventos `on()` a la instancia de clip de película, como se indica en "Utilización de controladores de eventos de botones y de clips de película" en la página 313.
    - Asigne una función a uno de los controladores de eventos de ratón del objeto del clip de película (`onPress`, `onRelease`, etc.), tal y como se indica en "Utilización de métodos de controlador de eventos" en la página 306.
17. Seleccione Control > Probar película para probar el documento de Flash.

    Mueva el puntero del ratón sobre la instancia de clip de película en el escenario para que el clip de película vaya a su estado _over. Haga clic en la instancia de clip de película y la cabeza lectora irá al estado _down del clip de película.

# Ámbito del controlador de eventos

El ámbito o *contexto*, de las variables y los comandos que se declaran y se ejecutan con un controlador de eventos depende del tipo de controlador de eventos que se esté utilizando: controladores de eventos o detectores de eventos, o bien los controladores `on()` y `onClipEvent()`. Si está definiendo un controlador de eventos en una nueva clase de ActionScript, el ámbito también dependerá de cómo defina el controlador de eventos. En esta sección se incluyen ejemplos para ActionScript 1.0 y para ActionScript 2.0.

**Ejemplos para ActionScript 1.0** Las funciones asignadas a los métodos de controlador de eventos (como todas las funciones ActionScript que se crean) definen un ámbito de variable local, pero los controladores `on()` y `onClipEvent()` no lo hacen.

Por ejemplo, veamos los dos controladores de eventos siguientes. El primero es un controlador de eventos `onPress` asociado a un clip de película denominado `clip_mc`. El segundo es un controlador `on()` asociado a la misma instancia de clip de película.

```
// Asociado a la línea de tiempo del clip principal de clip_mc:
clip_mc.onPress = function () {
  var shoeColor; // variable de función local
  shoeColor = "blue";
}
// controlador on() asociado a clip_mc:
on (press) {
  var shoeColor; // sin ámbito de variable local
  shoeColor = "blue";
}
```

Aunque los dos controladores de evento tienen el mismo código, los resultados pueden ser diferentes. En el primer caso, la variable `color` es local para la función definida para `onPress`. En el segundo caso, dado que el controlador `on()` no define un ámbito de variable local, la variable se define en el ámbito de la línea de tiempo del clip de película `clip_mc`.

Para los controladores de eventos `on()` asociados a los botones, en lugar de a los clips de película, las variables (así como las llamadas de función y método) se invocan en el ámbito de la línea de tiempo que contiene la instancia de botón.

Por ejemplo, el controlador de eventos `on()` siguiente produce un resultado diferente dependiendo de si se ha asociado a un objeto de clip de película o de botón. En el primer caso, la llamada a la función `play()` pone en marcha la cabeza lectora en la línea de tiempo que contiene el botón; en el segundo, la llamada a la función `play()` inicia la línea de tiempo del clip de película al cual está asociado el controlador.

```
// Asociado al botón.
on (press) {
  play(); // Reproduce la línea de tiempo principal.
}
// Asociado al clip de película.
on (press) {
  play(); // Reproduce la línea de tiempo del clip de película.
}
```

Es decir, cuando se asocia a un objeto de botón, la función `play()` se aplica a la línea de tiempo que contiene el botón, es decir, la línea de tiempo principal del botón. Pero si el controlador `on(press)` está asociado a un objeto de clip de película, la llamada a la función `play()` se aplica al clip de película al que se ha asignado el controlador. Si asocia el siguiente código a un clip de película, se reproduce la línea de tiempo principal:

```
// Asociado al clip de película.
on (press) {
  _parent.play(); // Reproduce la línea de tiempo principal.
}
```

Dentro de una definición de controlador de eventos o de detector de eventos, se aplicaría la misma función `play()` a la línea de tiempo que contiene la definición de función. Por ejemplo, supongamos que declara el siguiente método de controlador de eventos `my_mc.onPress` en la línea de tiempo que contiene la instancia de clip de película `my_mc`:

```
// Función definida en una línea de tiempo:
my_mc.onPress = function () {
  play(); // reproduce la línea de tiempo en la que se ha definido.
};
```

Para reproducir el clip de película que define el controlador de eventos `onPress`, haga referencia explícita a ese clip mediante la palabra clave `this`, del modo siguiente:

```
// Función definida en la línea de tiempo raíz
my_mc.onPress = function () {
  this.play(); // reproduce la línea de tiempo del clip my_mc.
};
```

Sin embargo, el mismo código situado en la línea de tiempo raíz de una instancia de botón reproduciría la línea de tiempo raíz:

```
my_btn.onPress = function () {
  this.play(); // reproduce la línea de tiempo raíz
};
```

Para más información sobre el ámbito de la palabra clave `this` en los controladores de eventos, consulte "Ámbito de la palabra clave this" en la página 324.

**Ejemplo de ActionScript 2.0** La clase TextLoader siguiente se utiliza para cargar un archivo de texto y muestra texto después de cargar correctamente el archivo.

```
// TextLoader.as
class TextLoader {
  private var params_lv:LoadVars;
  public function TextLoader() {
    params_lv = new LoadVars();
    params_lv.onLoad = onLoadVarsDone;
    params_lv.load("http://www.helpexamples.com/flash/params.txt");
  }
  private function onLoadVarsDone(success:Boolean):Void {
    _level0.createTextField("my_txt", 999, 0, 0, 100, 20);
    _level0.my_txt.autoSize = "left";
    _level0.my_txt.text = params_lv.monthNames; // undefined
  }
}
```

Este código no puede funcionar correctamente porque hay un problema de ámbito de los controladores de eventos; existe confusión en cuanto a qué hace referencia `this` entre el controlador de eventos `onLoad` y la clase. El comportamiento que podría esperar de este ejemplo es que se invoque el método `onLoadVarsDone()` en el ámbito del objeto TextLoader; sin embargo, se invoca en el ámbito del objeto LoadVars porque el método se extrajo del objeto TextLoader y se ha llevado al objeto LoadVars. El objeto LoadVars invoca el controlador de eventos `this.onLoad` cuando el archivo de texto se ha cargado correctamente y la función `onLoadVarsDone()` se invoca con `this` configurado como LoadVars, no como TextLoader. El objeto `params_lv` reside en el ámbito `this` cuando se invoca, aunque la función `onLoadVarsDone()` dependa del objeto `params_lv` por referencia. Por consiguiente, la función `onLoadVarsDone()` está esperando una instancia de `params_lv.params_lv` que no existe.

Para invocar correctamente el método `onLoadVarsDone()` en el ámbito del objeto TextLoader, puede utilizar la siguiente estrategia: utilice la expresión literal de una función para crear una función anónima que llame a la función deseada. El objeto `owner` continúa estando visible en el ámbito de la función anónima, de manera que puede utilizarse para localizar el objeto TextLoader que realiza la llamada.

```
// TextLoader.as
class TextLoader {
  private var params_lv:LoadVars;
  public function TextLoader() {
    params_lv = new LoadVars();
    var owner:TextLoader = this;
    params_lv.onLoad = function (success:Boolean):Void {
      owner.onLoadVarsDone(success);
    }
    params_lv.load("http://www.helpexamples.com/flash/params.txt");
  }
  private function onLoadVarsDone(success:Boolean):Void {
    _level0.createTextField("my_txt", 999, 0, 0, 100, 20);
    _level0.my_txt.autoSize = "left";
    _level0.my_txt.text = params_lv.monthNames; //
  January,February,March,...
  }
}
```

# Ámbito de la palabra clave this

La palabra clave `this` hace referencia al objeto del ámbito de ejecución actual. Según el tipo de técnica de controlador de eventos que utilice, `this` puede hacer referencia a distintos objetos.

**Dentro de una función de controlador de eventos o de detector de eventos,** `this` hace referencia al objeto que define el método de controlador o detector de eventos. Por ejemplo, en el código siguiente, `this` hace referencia a `my_mc`:

```
// controlador de eventos onPress() asociado a la línea de tiempo principal:
my_mc.onPress = function () {
  trace(this); // _level0.my_mc
}
```

**Dentro de un controlador on() asociado a un clip de película**, `this` hace referencia al clip de película al que está asociado el controlador `on()`, como se muestra en el siguiente código:

```
// Asociado al clip de película denominado my_mc en la línea de tiempo
  principal
on (press) {
  trace(this); // _level0.my_mc
}
```

**Dentro de un controlador on() asociado a un botón**, `this` hace referencia a la línea de tiempo que contiene el botón, como se muestra en el siguiente código:

```
// Asociado al botón en la línea de tiempo principal
on (press) {
  trace(this); // _level0
}
```

# Utilización de la clase Delegate

La clase Delegate permite ejecutar una función en un ámbito específico. Esta clase se proporciona para que se pueda distribuir el mismo evento en dos funciones diferentes (consulte "Delegación de eventos a funciones" en *Utilización de componentes*) y para que se pueda llamar a las funciones dentro del ámbito de la clase que las contiene.

Cuando se pasa una función como un parámetro a `EventDispatcher.addEventListener()`, la función se invoca en el ámbito de la instancia del componente difusor, no del objeto en el que está declarada (consulte "Delegación del ámbito de una función" en *Utilización de componentes*). Se puede utilizar `Delegate.create()` para llamar a la función del ámbito del objeto que declara.

El siguiente ejemplo muestra los tres métodos de detección para eventos para una instancia de componente Button. Las formas de añadir detectores de eventos a una instancia de componente Button tienen como resultado que el evento se distribuya en un ámbito diferente.

### Para utilizar la clase Delegate para detectar eventos:

1. Cree un nuevo documento de Flash y guárdelo como **delegate.fla**.
2. Arrastre un componente Button desde la carpeta User Interface del panel Componentes a la biblioteca.

   Añadirá y colocará la instancia de botón en el escenario mediante el código ActionScript en un paso posterior.
3. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
import mx.controls.Button;
import mx.utils.Delegate;

function clickHandler(eventObj:Object):Void {
  trace("[" + eventObj.type + "] event on " + eventObj.target + "
  instance.");
  trace("\t this -> " + this);
}

var buttonListener:Object = new Object();
buttonListener.click = function(eventObj:Object):Void {
  trace("[" + eventObj.type + "] event on " + eventObj.target + "
  instance.");
  trace("\t this -> " + this);
};

this.createClassObject(Button, "one_button", 10, {label:"One"});
one_button.move(10, 10);
one_button.addEventListener("click", clickHandler);

this.createClassObject(Button, "two_button", 20, {label:"Two"});
two_button.move(120, 10);
two_button.addEventListener("click", buttonListener);

this.createClassObject(Button, "three_button", 30, {label:"Three"});
three_button.move(230, 10);
three_button.addEventListener("click", Delegate.create(this,
  clickHandler));
```

   El código anterior se divide en seis secciones (cada sección se separa por una línea en blanco). La primera sección importa la clase Button (para el componente Button), así como la clase Delegate. La segunda sección del código define una función a la que se llamará cuando el usuario haga clic en algunos de los botones. La tercera sección del código crea un objeto que se utiliza como un detector de eventos y el objeto detecta un único evento, `click`.

Cada una de las tres secciones que faltan crea una nueva instancia de componente Button en el escenario, vuelve a colocar la instancia y añade un detector de eventos para el evento `click`. El primer botón añade un detector de eventos para el evento `click` y pasa directamente una referencia a una función de controlador `click`. El segundo evento añade un detector de eventos para el evento `click` y pasa una referencia a un objeto de detector, que contiene un controlador para el evento `click`. Finalmente, la tercera función añade un detector de eventos para el evento `click`, utiliza la clase Delegate para distribuir el evento click en el ámbito `this` (donde `this` es igual a `_level0`) y pasa una referencia a la función de controlador `click`.

4. Seleccione Control > Probar película para probar el documento de Flash.
5. Haga clic en cada instancia de botón en el escenario para ver el ámbito en el que se controla el evento.
   a. Haga clic en el primer botón del escenario para rastrear el siguiente texto en el panel Salida:

      ```
      [click] event on _level0.one_button instance.
         this -> _level0.one_button
      ```

      Al hacer clic en la instancia `one_button`, el ámbito `this` se refiere a la propia instancia de botón.

   b. Haga clic en el segundo botón del escenario para rastrear el siguiente texto en el panel Salida:

      ```
      [click] event on _level0.two_button instance.
         this -> [object Object]
      ```

      Al hacer clic en la instancia `two_button`, el ámbito `this` se refiere al objeto `buttonListener`.

   c. Haga clic en el tercer botón del escenario para rastrear el siguiente texto en el panel Salida:

      ```
      [click] event on _level0.three_button instance.
         this -> _level0
      ```

      Al hacer clic en la instancia `three_button`, el ámbito `this` se refiere al ámbito que especifica en la llamada de método `Delegate.create()` o, en este caso, `_level0`.

CAPÍTULO 10

# Datos y tipos de datos

10

Este es el primero de una serie de capítulos dedicados a describir y demostrar algunos conceptos fundamentales de ActionScript. Tendrá la oportunidad de practicar algunas técnicas de codificación básicas que le permitirán aprender a crear aplicaciones complejas. En este capítulo, también aprenderá a utilizar datos en un archivo FLA y conocerá los tipos de datos con los que puede trabajar. En el siguiente capítulo, Capítulo 4, "Principios básicos de la sintaxis y el lenguaje" aprenderá a utilizar la sintaxis de ActionScript y a formar sentencias. Seguidamente, en el Capítulo 5, "Funciones y métodos" se demuestra cómo utilizar funciones y métodos en el lenguaje ActionScript.

Para más información sobre los datos y tipos de datos, consulte las secciones siguientes:



## Datos

Los *datos* son números, cadenas y otra información que puede manipular en Flash. La utilización de datos suele ser esencial a la hora de crear aplicaciones y sitios Web. También puede utilizar datos al crear gráficos avanzados y animaciones generadas mediante script, y es posible que tenga que manipular los valores empleados para controlar los efectos.

Puede definir datos en *variables* dentro de Flash o cargar datos de archivos o sitios externos mediante XML, servicios Web, clases ActionScript incorporadas, etc. Puede almacenar datos en una base de datos y luego representar dicha información de diversas formas en un archivo SWF. Entre ellas, puede mostrar la información en campos de texto o componentes, o bien mostrar imágenes en instancias de clip de película.

Entre las formas más comunes de datos se encuentran las cadenas (secuencias de caracteres, como pueden ser nombres o un texto), números, objetos (como, por ejemplo, clips de película), valores booleanos (`true` y `false`), etc. En este capítulo, también se describen los tipos de datos de Flash y cómo utilizarlos.

Para obtener información sobre los tipos de datos, consulte "Tipos de datos" en la página 328. Para obtener información sobre las variables, consulte "Variables" en la página 343.

# Tipos de datos

Un *tipo de datos* describe un dato y los tipos de operaciones que pueden realizarse con él. Los datos se almacenan en variables. Puede utilizar tipos de datos al crear variables, instancias de objetos y definiciones de funciones para asignar el tipo de datos que está manipulando. Al escribir código ActionScript, se emplean muchos tipos de datos diferentes.

ActionScript 2.0 define varios tipos de datos que se utilizan con frecuencia. Los tipos de datos describen el tipo de valor que puede contener una variable o un elemento de ActionScript. Una variable asignada a un tipo de datos sólo puede contener un valor incluido entre el conjunto de valores de dicho tipo de datos. Para obtener información sobre las variables, consulte "Variables" en la página 343.

ActionScript dispone de un gran número de tipos básicos de datos que probablemente utilizará con frecuencia en sus aplicaciones. Consulte la tabla en "Tipos de datos simples y complejos" en la página 329 para más información.

ActionScript también dispone de clases principales, como Array y Date, que se consideran tipos de datos complejos o de referencia. Para más información sobre tipos de datos complejos y de referencia, consulte "Tipos de datos simples y complejos" en la página 329. Asimismo, todos los tipos de datos y clases se definen íntegramente en *Referencia del lenguaje ActionScript 2.0*.

También puede crear clases personalizadas para las aplicaciones. Todas las clases que defina utilizando la declaración class también se considerarán tipos de datos. Para más información sobre clases principales y otros tipos de clases incorporadas, consulte "Clases de nivel superior y clases incorporadas" en la página 258. Para más información sobre la creación de clases personalizadas, consulte el Capítulo 6, "Clases", en la página 195.

En ActionScript 2.0, se puede asignar tipos de datos a variables cuando se declaran. Los tipos de datos que asigne podrán ser de cualquiera de los tipos principales o podrán ser clases personalizadas creadas por usted. Para más información, consulte "Asignación de tipos de datos y "strict data typing"" en la página 337.

Cuando se depuran scripts, a veces es necesario determinar el tipo de datos de una expresión o variable para entender por qué se comporta de cierta manera. Esto se puede hacer con los operadores `instanceof` y `typeof` (consulte "Determinación del tipo de datos" en la página 342).

Puede convertir un tipo de datos en otro durante la ejecución mediante una de las funciones de conversión siguientes: `Array()`, `Boolean()`, `Number()`, `Object()`, `String()`.

Puede encontrar un archivo de origen de ejemplo, datatypes.fla, en la carpeta Samples del disco duro, que muestra cómo utilizar los tipos de datos en la aplicación.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\DataTypes.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/DataTypes.

## Tipos de datos simples y complejos

Todos los valores de tipos de datos pueden agruparse en dos categorías principales: *simples* o *complejos*.

Un *valor simple* (o tipo de dato simple) es un valor que ActionScript almacena en el nivel más bajo de abstracción, lo que significa que las operaciones con tipos de datos simples son generalmente más rápidas y eficientes que las operaciones realizadas con tipos de datos complejos. Los siguientes tipos de datos definen un conjunto de uno o varios valores simples: Boolean (booleano), null (nulo), Number (número), String (cadena) y undefined (no definido).

Un *valor complejo* (o tipo de datos complejo) es un valor que no es simple y que hace referencia a los valores simples. Estos se denominan con frecuencia tipos de datos *de referencia*. Los valores complejos pertenecen al tipo de datos Object (objeto) o a un tipo de datos basado en el tipo de datos Object. Entre los tipos de datos que definen conjuntos de valores complejos se encuentran Array (matriz), Date (fecha), Error, Function (función) y XML. Para más información sobre estos tipos de datos complejos, consulte sus correspondientes entradas en *Referencia del lenguaje ActionScript 2.0*.

Las variables que contienen datos de tipo simple se comportan en ciertas situaciones de modo diferente a las que contienen tipos complejos. Para más información, consulte “Utilización de variables en un proyecto” en la página 364.

ActionScript dispone de los siguientes tipos básicos de datos que puede utilizar en sus aplicaciones:

| Tipo de datos | Descripción |
|---|---|
| Boolean | Simple. El tipo de datos Boolean (booleano) consta de dos valores: `true` y `false`. Ningún otro valor es válido para variables de este tipo. El valor predeterminado de una variable booleana declarada pero no inicializada es `false`. Para más información, consulte "Tipo de datos Boolean (booleano)" en la página 331. |
| MovieClip | Complejo. El tipo de datos MovieClip permite controlar los símbolos de clips de película mediante los métodos de la clase MovieClip. Para más información, consulte "Tipo de datos MovieClip (clip de película)" en la página 332. |
| null | Simple. El tipo de datos null (nulo) contiene el valor `null`. Este valor significa "ningún valor", es decir, no hay datos. Puede asignar el valor `null` en distintas situaciones para indicar que una propiedad o variable no tiene ningún valor asignado. El tipo de datos null es el tipo de datos predeterminado para todas las clases que definen tipos de datos complejos. Una excepción a esta regla es la clase Object, que adopta de manera predeterminada el valor `undefined`. Para más información, consulte "Tipo de datos null (nulo)" en la página 334. |
| Number | Simple. Este tipo de datos representa enteros, enteros sin signo y números de coma flotante. Para almacenar un número de coma flotante, debe incluir una coma decimal en el número. Sin la coma decimal, el número se almacena como un entero. El tipo de datos Number puede almacenar valores entre `Number.MAX_VALUE` (muy alto) y `Number.MIN_VALUE` (muy bajo). Para más información, consulte *Referencia del lenguaje ActionScript 2.0* y "Tipo de datos Number (número)" en la página 334 |
| Object | Complejo. El tipo de datos Object (objeto) se define mediante la clase Object. La clase Object sirve de base para todas las definiciones de clases en ActionScript y le permite organizar unos objetos dentro de otros (objetos anidados). Para más información, consulte "Tipo de datos Object (objeto)" en la página 335. |
| String | Simple. El tipo de datos String (cadena) representa una secuencia de caracteres de 16 bits que puede incluir letras, números y signos de puntuación. Las cadenas se almacenan como caracteres Unicode empleando el formato UTF-16. Una operación sobre un valor de cadena (String) devuelve una nueva instancia de la cadena. Para más información, consulte "Tipo de datos String (cadena)" en la página 336. |

| Tipo de datos | Descripción |
|---|---|
| undefined | Simple. El tipo de datos undefined (no definido) contiene un valor: `undefined`. Este es el valor predeterminado de las instancias de la clase Object. Sólo puede asignar el valor `undefined` a variables que pertenezcan a la clase Object. Para más información, consulte "Tipo de datos undefined (no definido)" en la página 337. |
| Void | Complejo. El tipo de datos Void (vacío) sólo contiene un valor: `void`. Este tipo de datos se usa para designar funciones que no devuelven un valor. Void es un tipo de datos complejo que hace referencia al tipo de datos Void simple. Para más información, consulte "Tipo de datos Void (vacío)" en la página 337. |

Puede encontrar un archivo de origen de ejemplo, datatypes.fla, en la carpeta Samples del disco duro, que muestra cómo utilizar los tipos de datos en una aplicación.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\DataTypes.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/DataTypes.

## Tipo de datos Boolean (booleano)

Un valor booleano puede ser `true` o `false`. ActionScript también convierte los valores `true` y `false` en `1` y `0` según sea adecuado. Los valores booleanos se usan con mayor frecuencia con los operadores lógicos en sentencias de ActionScript que realizan comparaciones para controlar el flujo de un script.

El siguiente ejemplo carga un archivo de texto en un archivo SWF y muestra un mensaje en el panel Salida si el archivo de texto no se carga correctamente, o bien los parámetros si la carga es correcta. Consulte los comentarios en el ejemplo de código para obtener más detalles.

```
var my_lv:LoadVars = new LoadVars();
//"success" (correcto) es un valor booleano
my_lv.onLoad = function(success:Boolean) {
  //si "success" es true, trazar monthNames
  if (success){
    trace(my_lv.monthNames);
  //si "success" es false, trazar un mensaje
  } else {
    trace("no se puede cargar el archivo de texto");
  }
};
my_lv.load("http://www.helpexamples.com/flash/params.txt");
```

En el siguiente ejemplo se comprueba que los usuarios introducen valores en dos instancias del componente TextInput. Se crean dos variables booleanas, `userNameEntered` y `isPasswordCorrect`, y si ambas variables dan como resultado `true` (verdadero), se asigna un mensaje de bienvenida a la variable de cadena (String) `titleMessage`.

```
// Añadir dos componentes TextInput, un componente Label y un componente
  Button al escenario.
// Utilizar "strict data type" para las tres instancias de componentes
var userName_ti:mx.controls.TextInput;
var password_ti:mx.controls.TextInput;
var submit_button:mx.controls.Button;
var welcome_lbl:mx.controls.Label;

//Ocultar la etiqueta
welcome_lbl.visible = false;

// Crear un objeto detector, que se utiliza con el componente Button.
// Al hacerse clic en el botón, se comprueba un nombre de usuario y
  contraseña.
var btnListener:Object = new Object();
btnListener.click = function(evt:Object) {
  // Comprueba que el usuario ha introducido al menos un carácter en las
  instancias TextInput
  //y devuelve un valor booleano true/false.
  var userNameEntered:Boolean = (userName_ti.text.length > 0);
  var isPasswordCorrect:Boolean = (password_ti.text == "vertigo");
  if (userNameEntered && isPasswordCorrect) {
    var titleMessage:String = "¡Bienvenido " + userName_ti.text + "!";
    welcome_lbl.text = titleMessage;
    //muestra la etiqueta
    welcome_lbl.visible = true;
  }
};
submit_button.addEventListener("click", btnListener);
```

Para más información, consulte "Utilización de funciones en Flash" en la página 183 y "Operadores lógicos" en la página 162.

## Tipo de datos MovieClip (clip de película)

Los clips de película son símbolos que pueden reproducir animaciones en una aplicación Flash. Son el único tipo de datos que hace referencia a elementos gráficos. El tipo de datos MovieClip permite controlar los símbolos de clips de película mediante los métodos de la clase MovieClip.

No es necesario que utilice un constructor para llamar a los métodos de la clase MovieClip. Puede crear una instancia de clip de película en el escenario o crear una instancia dinámicamente. Posteriormente sólo tendrá que llamar a los métodos de la clase MovieClip utilizando el operador de punto (`.`).

**Utilización de clips de película en el escenario** El siguiente ejemplo llama a los métodos `startDrag()` y `getURL()` para las distintas instancias de clip de película que se encuentran en el escenario:

```
my_mc.startDrag(true);
parent_mc.getURL("http://www.macromedia.com/support/" + product);
```

El segundo ejemplo devuelve la anchura de un clip de película denominado `my_mc` existente en el escenario. La instancia de destino debe ser un clip de película y el valor devuelto debe ser numérico.

```
function getMCWidth(target_mc:MovieClip):Number {
  return target_mc._width;
}
trace(getMCWidth(my_mc));
```

**Creación de clips de película dinámicamente** La utilización de ActionScript para crear clips de película de forma dinámica resulta útil cuando desea evitar la creación manual de clips de película en el escenario o tener que asociarlos desde la biblioteca. Por ejemplo, puede crear una galería de imágenes con gran cantidad de imágenes en miniatura que desea organizar en el escenario. `MovieClip.createEmptyMovieClip()` le permite crear una aplicación completa mediante ActionScript.

Para crear dinámicamente un clip de película, utilice `MovieClip.createEmptyMovieClip()`, como se muestra en el siguiente ejemplo:

```
// Crea un clip de película para incluir el contenedor.
this.createEmptyMovieClip("image_mc", 9);
// Carga una imagen en image_mc.
image_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
```

El segundo ejemplo crea un clip de película denominado `square_mc` que utiliza la interfaz API de dibujo para dibujar un rectángulo. Los controladores de eventos y los métodos `startDrag()` y `stopDrag()` de la clase MovieClip se añaden para permitir que el rectángulo pueda arrastrarse.

```
this.createEmptyMovieClip("square_mc", 1);
square_mc.lineStyle(1, 0x000000, 100);
square_mc.beginFill(0xFF0000, 100);
square_mc.moveTo(100, 100);
square_mc.lineTo(200, 100);
square_mc.lineTo(200, 200);
square_mc.lineTo(100, 200);
square_mc.lineTo(100, 100);
square_mc.endFill();
square_mc.onPress = function() {
  this.startDrag();
};
square_mc.onRelease = function() {
  this.stopDrag();
};
```

Para más información, consulte el Capítulo 11, “Trabajo con clips de película”, en la página 373 y la entrada %{MovieClip}% de *Referencia del lenguaje ActionScript 2.0*.

## Tipo de datos null (nulo)

El tipo de datos null (nulo) tiene únicamente un valor: `null`. Este valor significa *ningún valor*, es decir, no hay datos. Puede asignar el valor `null` en distintas situaciones para indicar que una propiedad o variable no tiene ningún valor asignado todavía. Por ejemplo, puede asignar el valor `null` en las siguientes situaciones:

- Para indicar que una variable existe pero todavía no ha recibido ningún valor
- Para indicar que una variable existe pero ya no contiene ningún valor
- Como valor de retorno de una función, para indicar que la función no ha encontrado ningún valor disponible para devolverlo
- Como parámetro de una función, para indicar que un parámetro se ha omitido

Diversos métodos y funciones devuelven `null` si no se ha establecido ningún valor. En el siguiente ejemplo se demuestra cómo utilizar `null` para comprobar si los campos de un formulario tienen la selección actual del formulario:

```
if (Selection.getFocus() == null) {
  trace("no selection");
}
```

## Tipo de datos Number (número)

El tipo de datos Number (número) es un número de coma flotante de doble precisión. El valor mínimo de un objeto de número es aproximadamente 5e-324. El máximo es aproximadamente 1.79E+308.

Puede manipular los números mediante los operadores aritméticos de suma (+), resta (-), multiplicación (*), división (/), módulo (%), incremento (++) y decremento (--). Para más información, consulte "Utilización de operadores numéricos" en la página 156.

También puede utilizar métodos de las clases incorporadas Math y Number para manipular los números. Para obtener información sobre los métodos y las propiedades de estas clases, consulte las entradas %{Math}% y %{Number}% en *Referencia del lenguaje ActionScript 2.0*.

En el ejemplo siguiente se utiliza el método `sqrt()` (raíz cuadrada) de la clase Math para devolver la raíz cuadrada del número 100:

```
Math.sqrt(100);
```

En el siguiente ejemplo se averigua un entero aleatorio entre 10 y 17 (inclusive):

```
var bottles:Number = 0;
bottles = 10 + Math.floor(Math.random() * 7);
trace("There are " + bottles + " bottles");
```

En el ejemplo siguiente, se averigua el porcentaje cargado del clip de película `intro_mc` y se representa en forma de entero:

```
var percentLoaded:Number = Math.round((intro_mc.getBytesLoaded() /
  intro_mc.getBytesTotal()) * 100);
```

## Tipo de datos Object (objeto)

Un objeto es un conjunto de propiedades. Una *propiedad* es un atributo que describe el objeto. Por ejemplo, la transparencia de un objeto (como, por ejemplo, un clip de película) es un atributo que describe su apariencia. Por consiguiente, (la transparencia) `_alpha` es una propiedad. Cada propiedad tiene un nombre y un valor. El valor de una propiedad puede ser cualquier tipo de datos de Flash, incluso el tipo de datos Object. Esto permite organizar unos objetos dentro de otros o *anidarlos*.

Para especificar objetos y sus propiedades, debe utilizar el operador punto (`.`). Por ejemplo, en el siguiente código, `hoursWorked` es una propiedad de `weeklyStats`, que a su vez es una propiedad de `employee`:

```
employee.weeklyStats.hoursWorked
```

El objeto MovieClip de ActionScript tiene métodos que permiten controlar las instancias del símbolo de clip de película en el escenario. En este ejemplo se utilizan los métodos `play()` y `nextFrame()`:

```
mcInstanceName.play();
mc2InstanceName.nextFrame();
```

También puede crear objetos personalizados para organizar la información en la aplicación Flash. Para añadir interactividad a una aplicación con ActionScript, necesita numerosos tipos de información: por ejemplo, puede necesitar un nombre de un usuario, su edad y su número de teléfono, la velocidad de una pelota, los nombres de los artículos de un carrito de la compra, el número de fotogramas cargados, o la última tecla que presionó el usuario. La creación de objetos personalizados permite organizar esta información en grupos, simplificar la creación de scripts y reutilizarlos.

El siguiente código ActionScript muestra un ejemplo de utilización de objetos para organizar la información. Crea un nuevo objeto denominado `user` y crea tres propiedades, `name`, `age` y `phone`, que son tipos de datos String (cadena) y Number (número).

```
var user:Object = new Object();
user.name = "Irving";
user.age = 32;
user.phone = "555-1234";
```

Para más información, consulte “Ejemplo: Escritura de clases personalizadas” en la página 233.

## Tipo de datos String (cadena)

Una cadena es una secuencia de caracteres tales como letras, números y signos de puntuación. Las cadenas se introducen en una sentencia de ActionScript entre comillas simples (') o dobles (").

Una forma habitual de utilizar el tipo de datos de cadena consiste en asignar una cadena a una variable. Por ejemplo, en la siguiente sentencia, `"L7"` es una cadena asignada a la variable `favoriteBand_str`:

```
var favoriteBand_str:String = "L7";
```

Puede utilizar el operador de suma (+) para *concatenar* o unir dos cadenas. ActionScript trata los espacios del comienzo o del final de una cadena como parte literal de la cadena. La siguiente expresión incluye un espacio después de la coma:

```
var greeting_str:String = "Welcome, " + firstName;
```

Para incluir comillas en una cadena, coloque delante el carácter de barra inversa (\). A esto se le llama *utilizar una secuencia de escape* en un carácter. Existen otros caracteres que no pueden representarse en ActionScript, a menos que se utilice una secuencia de escape especial. En la siguiente tabla se enumeran todos los caracteres de escape de ActionScript:

| Secuencia de escape | Carácter |
|---|---|
| `\b` | Carácter de retroceso (ASCII 8) |
| `\f` | Carácter de salto de página (ASCII 12) |
| `\n` | Carácter de avance de línea (ASCII 10) |
| `\r` | Carácter de retorno de carro (ASCII 13) |
| `\t` | Carácter de tabulación (ASCII 9) |
| `\"` | Comillas dobles |
| `\'` | Comillas simples |
| `\\` | Barra inversa |
| `\000 - \377` | Byte especificado en octal |
| `\x00 - \xFF` | Byte especificado en hexadecimal |
| `\u0000 - \uFFFF` | Carácter Unicode de 16 bits especificado en hexadecimal |

Las cadenas en ActionScript son inmutables, al igual que en Java. Toda operación que modifique una cadena devuelve una nueva cadena.

La clase String es una clase incorporada en ActionScript. Para obtener información sobre los métodos y las propiedades de la clase String, consulte la entrada %{String}% en *Referencia del lenguaje ActionScript 2.0*.

## Tipo de datos undefined (no definido)

El tipo de datos undefined (no definido) tiene un valor, `undefined`, que se asigna automáticamente a una variable cuando aún no se le ha asignado ningún valor, ya sea mediante el código o por la interacción del usuario.

El valor `undefined` se asigna automáticamente; a diferencia de `null`, no tiene que asignar `undefined` a una variable o propiedad. El tipo de datos undefined permite comprobar si se ha establecido o definido una variable. Este tipo de datos le permite escribir código que se ejecuta solamente cuando la aplicación se está ejecutando, como se muestra en el siguiente ejemplo:

```
if (init == undefined) {
  trace("initializing app");
  init = true;
}
```

Si la aplicación tiene varios fotogramas, el código no se ejecuta una segunda vez, ya que la variable `init` ha dejado de ser undefined (es decir, ya está definida).

## Tipo de datos Void (vacío)

El tipo de datos Void (vacío) tiene un valor, `void`, y se utiliza en la definición de una función para indicar que la función no devuelve un valor, como se muestra en el siguiente ejemplo:

```
//Crea una función con el tipo de datos devueltos Void
function displayFromURL(url:String):Void {}
```

# Asignación de tipos de datos y “strict data typing”

Las variables se utilizan en Flash para incluir valores en el código. Puede declarar de forma explícita el tipo de objeto de una variable al crearla, lo que recibe el nombre de *strict data typing*.

Si no define explícitamente un elemento como contenedor de un número, una cadena u otro tipo de datos, Flash Player intentará, en tiempo de ejecución, determinar el tipo de datos de un elemento al asignarlo. Si asigna un valor a una variable, como se muestra en el ejemplo siguiente, Flash Player evalúa en tiempo de ejecución el elemento del lado derecho del operador y determina que su tipo de datos es Number:

```
var x = 3;
```

Dado que `x` no se declaró mediante “strict data typing”, el compilador no podrá determinar el tipo; para el compilador, la variable *x* puede tener un valor de cualquier tipo. (Consulte “Asignación de un tipo de datos” en la página 339.) Una asignación posterior puede cambiar el tipo de `x`; por ejemplo, la sentencia `x = "hello"` cambia el tipo de `x` a String (cadena).

ActionScript convierte siempre automáticamente los tipos de datos simples (como Boolean, Number, String, null o undefined) cuando una expresión requiere la conversión y las variables no se han introducido con "strict data typing".

El método "strict data typing" ofrece diversas ventajas a la hora de compilar. La declaración de tipos de datos (strict data typing) puede ayudar a evitar o diagnosticar errores existentes en el código al compilar. Para declarar una variable mediante "strict data typing", utilice el siguiente formato:

```
var variableName:datatype;
```

| NOTA | En ocasiones, "strict data typing" se conoce como la *comprobación de tipos al compilar* una variable. |
|---|---|

Puesto que las discordancias entre tipos de datos activan errores del compilador, "strict data typing" le ayuda a encontrar errores en el código al compilar e impide que se asigne un tipo de datos incorrecto a una variable existente. Durante la edición, "strict data typing" activa sugerencias para el código en el editor de ActionScript (pero deberá seguir utilizando sufijos de nombres de instancias para los elementos visuales).

"Strict data typing" garantiza que no se asigne accidentalmente un tipo de valor incorrecto a una variable. Durante la compilación, Flash comprueba errores de discordancia entre los tipos y muestra un mensaje de error si ha utilizado un tipo de valor erróneo. Por consiguiente, la utilización de "strict typing" también contribuye a garantizar que no intente acceder a propiedades o métodos que no forman parte del tipo de un objeto. El uso de "strict data typing" implica que el editor de ActionScript muestre automáticamente sugerencias para el código de los objetos.

Para más información sobre la creación de variables, consulte "Variables" en la página 343. Para más información sobre la asignación de nombres a variables, consulte "Asignación de nombre a variables" en la página 348. Para más información sobre la asignación de tipos de datos y los tipos que puede asignar, consulte "Asignación de un tipo de datos" en la página 339.

Puede encontrar un archivo de origen de ejemplo, datatypes.fla, en la carpeta Samples del disco duro, que muestra cómo utilizar los tipos de datos en una aplicación.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\DataTypes.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/DataTypes.

## Asignación de un tipo de datos

Debe asignar tipos de datos siempre que defina una variable con independencia de si declara una variable mediante la palabra clave `var`, crea un argumento de función, establece el tipo de devolución de función o define una variable para utilizarla dentro de un bucle `for` o `for..in`. Para asignar un tipo de datos, utilice la *sintaxis de signo de dos puntos posterior*, lo que significa que, tras el nombre de la variable, se incluyen dos puntos y, seguidamente, el tipo de datos:

```
var my_mc:MovieClip;
```

Existen numerosas posibilidades para los tipos de datos, desde tipos de datos nativos, como Number, String, Boolean, hasta clases incorporadas incluidas en Flash Player 8, como BitmapData o FileReference, o incluso clases personalizadas escritas por usted u otros desarrolladores. Los tipos de datos que es posible que tenga que especificar con más frecuencia son los tipos de datos incorporados, como Number, String, Boolean, Array u Object, que se muestran en los siguientes ejemplos de código.

Para asignar un tipo de datos específico a un elemento, especifique el tipo con la palabra clave `var` y la sintaxis de signo de dos puntos posterior, como se muestra en el siguiente ejemplo:

```
// "Strict typing" de variable u objeto
var myNum:Number = 7;
var birthday:Date = new Date();

// "Strict typing" de los parámetros
function welcome(firstName:String, age:Number) {
}

// "Strict typing" de los parámetros y del valor devuelto
function square(myNum:Number):Number {
  var squared:Number = myNum * myNum;
  return squared;
}
```

Puede declarar el tipo de datos de objetos basados en clases incorporadas (Button, Date, etc.) así como las clases e interfaces que cree. En el siguiente ejemplo, si tiene un archivo llamado Student.as en el que define la clase Student, puede especificar que los objetos que cree sean del tipo Student:

```
var myStudent:Student = new Student();
```

En este ejemplo, suponga que escribe el código siguiente:

```
// en el archivo de clase Student.as
class Student {
  public var status:Boolean; // propiedad de objetos Student
}

// en el archivo FLA
var studentMaryLago:Student = new Student();
studentMaryLago.status = "enrolled"; /* El tipo de la declaración de
  asignación no coincide: se encontró String donde era necesario Boolean.
  */
```

Cuando Flash compila este script, se genera un error de coincidencia de tipos porque el archivo SWF espera un valor booleano.

Si escribe una función que carece de tipo de devolución, puede especificar el tipo de devolución Void para dicha función. O bien, si crea un acceso directo a una función, puede asignar el tipo de datos Function a la nueva variable. Para especificar que los objetos son de tipo Function o Void, consulte el siguiente ejemplo:

```
function sayHello(name_str:String):Void {
  trace("Hello, " + name_str);
}
sayHello("world"); // Hello, world (Hola, mundo)
var greeting:Function = sayHello;
greeting("Augustus"); // Hello, Augustus (Hola, Augusto)
```

Otra de las ventajas de “strict data typing” es que Flash muestra de forma automática sugerencias para el código para los objetos incorporados si se introducen con strict typing. Para más información, consulte “Asignación de tipos de datos y “strict data typing”” en la página 337.

Los archivos publicados mediante ActionScript 1.0 no respetan las asignaciones de “strict data typing” al compilar, por lo que la asignación de un tipo de valor erróneo a una variable que ha introducido con strict typing no genera un error del compilador.

```
var myNum:String = "abc";
myNum = 12;
/* No hay error en ActionScript 1.0, pero hay un error de coincidencia en
  ActionScript 2.0 */
```

El motivo de este comportamiento es que al publicar un archivo en ActionScript 1.0, Flash interpreta una sentencia como, por ejemplo, `var myNum:String = "abc"`, como sintaxis con barras, en lugar de “strict typing”. (ActionScript 2.0 no admite la sintaxis con barras.) Este comportamiento puede dar como resultado que se asigne un objeto a una variable de tipo incorrecto y que el compilador permita llamadas de método no válidas y que se pasen referencias de propiedades no definidas sin notificarlas.

Los archivos publicados con ActionScript 2.0 pueden usar “strict data typing” opcionalmente. Por consiguiente, si implementa “strict data typing” en el código, asegúrese de que establece la configuración de publicación para ActionScript 2.0. Puede especificar la configuración de publicación y definir la versión de ActionScript que desea para publicar los archivos modificando la configuración de publicación desde el menú principal (Archivo > Configuración de publicación) o haciendo clic en el botón Configuración del inspector de propiedades (asegúrese de que no hay instancias seleccionadas). Para utilizar una versión concreta de ActionScript o Flash Player, seleccione la ficha Flash del cuadro de diálogo Configuración de publicación y seleccione una opción del menú emergente Versión de ActionScript.

Para obtener información sobre la verificación de tipos, consulte “Verificación de tipos” en la página 341.

## Verificación de tipos

La *verificación de tipos* se refiere a la comprobación de que el tipo de una variable y una expresión sean compatibles. Por consiguiente, Flash comprueba que el tipo especificado para una variable coincida con el valor o valores que les asigne. Para más información sobre “strict data typing” y la asignación de tipos de datos, consulte “Asignación de tipos de datos y “strict data typing”” en la página 337 y “Asignación de un tipo de datos” en la página 339.

La verificación de tipos puede tener lugar en la compilación o en la ejecución. Si utiliza “strict data typing”, la verificación de tipos tiene lugar durante la compilación. Dado que ActionScript es un lenguaje con tipos dinámicos, ActionScript puede también verificar los tipos durante la ejecución.

Por ejemplo, el siguiente código no especifica el tipo de datos del parámetro `xParam`. Durante la ejecución, se utiliza el parámetro para incluir un valor de tipo Number (número) y luego un valor de tipo `String` (cadena). La función `dynamicTest()` utiliza posteriormente el operador `typeof` para comprobar si el parámetro es de tipo String o Number.

```
function dynamicTest(xParam) {
  if (typeof(xParam) == "string") {
    var myStr:String = xParam;
    trace("String: " + myStr);
  } else if (typeof(xParam) == "number") {
    var myNum:Number = xParam;
    trace("Number: " + myNum);
  }
}
dynamicTest(100);
dynamicTest("one hundred");
```

No es necesario que añada explícitamente información de tipos de datos en el código ActionScript. El compilador de ActionScript le permite utilizar propiedades e invocar métodos que no existen durante la compilación. Esto le permite crear propiedades o asignar métodos dinámicamente durante la ejecución.

Un ejemplo de la flexibilidad que ofrece la verificación dinámica de tipos es el uso de propiedades y métodos desconocidos durante la compilación. Dado que el código es menos restrictivo, esto puede aportar ventajas en algunas situaciones de codificación. Por ejemplo, el siguiente código crea una función denominada `runtimeTest()` que invoca un método y devuelve una propiedad, pero el compilador desconoce todos estos elementos. El código no generará errores al compilarse, pero si la propiedad o el método no están accesibles durante la ejecución, se producirá un error durante la ejecución.

```
function runtimeTest(myParam) {
  myParam.someMethod();
  return myParam.someProperty;
}
```

## Determinación del tipo de datos

Durante la prueba y la depuración de los programas, es posible que detecte problemas que parezcan estar relacionados con los tipos de datos de diferentes elementos. O bien, si utiliza variables no asociadas explícitamente a ningún tipo de datos, puede que le resulte útil conocer el tipo de datos de una variable determinada. Mediante ActionScript, puede determinar el tipo de datos de un elemento. Puede utilizar el operador `typeof` para que se devuelva información sobre los datos.

Utilice el operador `typeof` para obtener los tipos de datos, pero recuerde que `typeof` no devuelve información sobre la clase a la que pertenece una instancia.

En el siguiente ejemplo se muestra cómo utilizar el operador `typeof` para que se devuelva el tipo del objeto del que está realizando el seguimiento:

```
// Crear una nueva instancia de clase LoadVars.
var my_lv:LoadVars = new LoadVars();

/* El operador typeof no especifica la clase, sólo especifica que my_lv es
  un objeto */
var typeResult:String = typeof(my_lv);
trace(typeResult); // objeto
```

En este ejemplo, se crea una nueva variable de cadena (String) denominada `myName` y luego se convierte al tipo de datos Number (número):

```
var myName:String = new String("17");
trace(myName instanceof String); // true
var myNumber:Number = new Number(myName);
trace(myNumber instanceof Number); // true
```

Para más información sobre estos operadores, consulte `%{typeof operator}%` y `%{instanceof operator}%` en *Referencia del lenguaje ActionScript 2.0*. Para más información sobre los procesos de prueba y depuración, consulte el Capítulo 18, “Depuración de aplicaciones”, en la página 753. Para más información sobre herencia e interfaces, consulte el Capítulo 7, “Herencia”, en la página 275. Para más información sobre clases, consulte el Capítulo 6, “Clases”, en la página 195.

# Variables

Una *variable* es un contenedor que almacena información. En el siguiente código ActionScript se muestra el aspecto de una variable en ActionScript:

```
var myVariable:Number = 10;
```

Esta variable contiene un valor numérico. El uso de `:Number` en el código anterior asigna el tipo de valor que contiene dicha variable, denominado *data typing* (asignación de tipo a los datos). Para más información sobre la escritura de datos, consulte “Asignación de tipos de datos y “strict data typing”” en la página 337 y “Asignación de un tipo de datos” en la página 339.

El contenedor (representado por el nombre de la variable) es siempre el mismo en todo el código ActionScript, pero el contenido (el *valor*) puede cambiar. Puede cambiar el valor de una variable en un script tantas veces como desee. Al cambiar el valor de una variable a medida que se reproduce el archivo SWF, puede registrar y guardar información sobre las acciones del usuario, registrar valores que cambian conforme se reproduce el archivo SWF o comprobar si una determinada condición es `true` o `false`. Puede que sea necesario que la variable se actualice continuamente mientras se reproduce el archivo SWF, como cuando cambia la puntuación de un jugador en un juego de Flash. Las variables son esenciales cuando se crea y manipula la interacción del usuario en un archivo SWF.

Cuando se declara una variable por primera vez, se recomienda asignarle un valor. La asignación de un valor inicial se conoce como *inicializar* la variable, acción que normalmente se realiza en el fotograma 1 de la línea de tiempo o desde dentro de una clase que se carga cuando el archivo SWF comienza a reproducirse. Existen diferentes tipos de variables, que se ven afectadas por el ámbito. Para más información sobre los diferentes tipos de variables y el ámbito, consulte "Variables y ámbito" en la página 354.

| SUGERENCIA | Inicializar las variables permite realizar un seguimiento y comparar el valor de la variable a medida que se reproduce el archivo SWF. |
|---|---|

| NOTA | Flash Player 7 y versiones posteriores evalúan las variables no inicializadas de forma distinta a Flash Player 6 y versiones anteriores. Si ha escrito scripts para Flash Player 6 y tiene previsto escribir o transferir scripts para Flash Player 7 o versiones posteriores, deberá conocer estas diferencias para evitar un comportamiento inesperado. |
|---|---|

Las variables pueden contener diferentes tipos de datos; para más información, consulte "Tipos de datos" en la página 328. El tipo de datos que contiene una variable afecta al modo en el que cambia el valor de la variable al asignar dicho valor en un script.

El tipo de información que habitualmente se guarda en una variable es una URL (tipo String -cadena-), un nombre de usuario (tipo String), el resultado de una operación matemática (tipo Number -número-), el número de veces que ocurre un evento (tipo Number) o si un usuario ha hecho clic en un botón determinado (tipo Boolean -booleano-). Cada archivo SWF e instancia de objeto (como, por ejemplo, un clip de película) tiene un conjunto de variables, cada una de ellas con su propio valor independiente del de otras variables definidas en otros archivos SWF o clips de película.

Para ver el valor de una variable, utilice la sentencia `trace()` que enviará el valor al panel Salida. Seguidamente, el valor aparece en el panel Salida al comprobar el archivo SWF en el entorno de prueba. Por ejemplo, `trace(hoursWorked)` envía el valor de la variable `hoursWorked` al panel Salida en el entorno de prueba. También puede comprobar y establecer los valores de las variables en el Depurador en el entorno de prueba.

Para más información sobre variables, consulte los temas siguientes:

- "Declaración de variables" en la página 345
- "Asignación de valores" en la página 345
- "Asignación de nombre a variables" en la página 348
- "Utilización de variables en una aplicación" en la página 349
- "Variables y ámbito" en la página 354

## Declaración de variables

Puede declarar variables en un fotograma de la línea de tiempo, directamente en un objeto o en un archivo de clase externo.

Defina variables mediante la palabra clave `var` y siga las convenciones de denominación de variables. Puede declarar una variable denominada `firstName` como se muestra en el siguiente ejemplo:

```
var firstName:String;
```

Al declarar una variable, se asigna un tipo de datos a la variable. En este caso, se asigna el tipo de datos String (cadena) a la variable `firstName`. Para más información sobre la asignación de tipos de datos, consulte “Asignación de tipos de datos y “strict data typing”” en la página 337.

## Valores predeterminados

Un *valor predeterminado* es el valor que contiene una variable antes de establecer su valor. Una variable se *inicializa* al establecer su valor por primera vez. Si declara una variable pero no establece su valor, dicha variable estará *sin inicializar*. Una variable no inicializada adopta de manera predeterminada el valor `undefined` (no definido). Para más información sobre la creación y utilización de variables, consulte “Variables” en la página 343.

## Asignación de valores

Puede definir un *valor* como el contenido actual de una variable. El valor puede ser una cadena, número, matriz, objeto, XML, fecha o incluso una clase personalizada creada por usted. Recuerde que la declaración de una variable en Flash se realiza mediante la palabra clave `var`. Al declarar la variable, se asigna también un tipo de datos a la variable. También puede asignar un valor a una variable, siempre y cuando el valor coincida con el tipo de datos que asigne a la variable.

En el siguiente ejemplo se muestra cómo podría crearse una variable denominada `catName`:

```
var catName:String;
```

Después de declarar la variable, puede asignarle un valor. Tras la línea de código ActionScript anterior, podría incluir la siguiente:

```
catName = "Pirate Eye";
```

| NOTA | Dado que `Pirate Eye` es una cadena, el valor tiene que estar incluido entre comillas rectas (comillas). |
|---|---|

En este ejemplo, se asigna el valor `Pirate Eye` a la variable `catName`. Al declarar la variable, también puede asignarle un valor en lugar de asignárselo posteriormente (como en los ejemplos anteriores). Puede establecer la variable `catName` al declararla, como se muestra en el siguiente ejemplo:

```
var catName:String = "Pirate Eye";
```

Si desea mostrar el valor de la variable `catName` en el entorno de prueba, puede utilizar la sentencia `trace()`. Esta sentencia envía el valor al panel Salida. Puede realizar un seguimiento del valor de la variable `catName` y comprobar que el valor real no incluye las comillas empleando el siguiente código ActionScript:

```
var catName:String = "Pirate Eye";
trace(catName); // Pirate Eye
```

Recuerde que el valor que asigne debe coincidir con el tipo de datos que le haya asignado (en este caso, String). Si posteriormente intenta asignar un número a la variable `catName` como, por ejemplo, `catName = 10`, verá el siguiente error en el panel Salida cuando compruebe el archivo SWF:

```
Type mismatch in assignment statement: found Number where String is
  required.
```

Este error indica que ha intentado establecer un tipo de datos erróneo para una variable especificada.

Al asignar un valor numérico a una variable, las comillas no son necesarias, como se muestra en el siguiente código:

```
var numWrinkles:Number = 55;
```

Si desea cambiar el valor de `numWrinkles` posteriormente en el código, puede asignarle un nuevo valor mediante el siguiente código ActionScript:

```
numWrinkles = 60;
```

Al cambiar la asignación del valor de una variable existente, no es necesario utilizar la palabra clave `var` ni definir el tipo de datos de la variable (`:Number` en este caso).

Si el valor es numérico o booleano (`true` o `false`), el valor no requiere comillas rectas (comillas). En el siguiente fragmento se muestran ejemplos de valores numéricos y booleanos:

```
var age:Number = 38;
var married:Boolean = true;
var hasChildren:Boolean = false;
```

En el ejemplo anterior, la variable `age` contiene un valor entero (no decimal), aunque también puede utilizar un valor decimal o de coma flotante como, por ejemplo, 38,4. Las variables booleanas (como `married` -casado- o `hasChildren` -con hijos-) sólo tienen dos valores posibles, `true` (verdadero) o `false` (falso).

Si desea crear una matriz y asignarle valores, el formato es ligeramente distinto, como se muestra en el siguiente código:

```
var childrenArr:Array = new Array("Pylon", "Smithers", "Gil");
```

Existe una sintaxis alternativa (abreviada) para la creación de una matriz mediante operadores de acceso a una matriz, que utilizan corchetes (`[]`). Puede reescribir el ejemplo anterior de la siguiente forma:

```
var childrenArr:Array = ["Pylon", "Smithers", "Gil"];
```

Para más información sobre la creación de matrices y los operadores de acceso a una matriz, consulte "Matrices" en la página 129 y "Utilización de la sintaxis con punto para referirse a una instancia" en la página 81.

De igual forma, puede crear un objeto nuevo denominado `myObj`. Puede crear un objeto nuevo de cualquiera de las siguientes formas. La primera forma (más larga) de codificar una matriz es la siguiente:

```
var myObj:Object = new Object();
myObj.firstName = "Steve";
myObj.age = 50;
myObj.childrenArr = new Array("Mike", "Robbie", "Chip");
```

La segunda forma (abreviada) de codificar la matriz `myObj` es la siguiente:

```
var myObj:Object = {firstName:"Steve", age:50, childrenArr:["Mike",
  "Robbie", "Chip"]};
```

Como puede comprobar en este ejemplo, la utilización del método abreviado puede ahorrarle tiempo y le evitará tener que introducir datos, particularmente al definir instancias de objetos. Es importante familiarizarse con esta sintaxis alternativa, ya que se encontrará con ella si trabaja en equipo o si utiliza código ActionScript de terceros que haya encontrado en Internet o en manuales.

**NOTA**

No todas las variables requieren su definición explícita. Flash crea algunas variables automáticamente. Por ejemplo, para averiguar las dimensiones del escenario, podría utilizar los valores de las siguientes variables predefinidas: `Stage.width` y `Stage.height`.

## Operadores y variables

Puede que se pregunte por el uso de símbolos matemáticos en el código. Los símbolos se denominan *operadores* en ActionScript. Los operadores calculan un valor nuevo a partir de uno o varios valores y un operador le permite asignar un valor a una variable del código. El operador de igualdad (=) permite asignar un valor a una variable:

```
var username:String = "Gus";
```

Otro ejemplo es el operador de suma (+), que se utiliza para sumar dos o más valores numéricos y generar un nuevo valor. Si utiliza el operador + en dos o más valores de cadena, las cadenas se concatenarán. Los valores que manipulan los operadores se denominan *operandos*.

Al asignar un valor, se utiliza un operador para definir el valor de una variable. Por ejemplo, en el siguiente script se utiliza el operador de asignación para asignar el valor 7 a la variable `numChildren`:

```
var numChildren:Number = 7;
```

Si desea cambiar el valor de la variable `numChildren`, utilice el siguiente código:

```
numChildren = 8;
```

| NOTA | No es necesario que utilice `var`, ya que la variable se ha definido anteriormente. |
|---|---|

Para más información sobre el uso de operadores en el código ActionScript, consulte "Operadores" en la página 142.

## Asignación de nombre a variables

Tenga cuidado al comenzar a asignar nombres a las variables, ya que, aunque pueden tener prácticamente cualquier nombre, deben cumplirse algunas reglas. El nombre de una variable debe seguir estas reglas:

- Una variable debe ser un identificador.

| NOTA | Un *identificador* es el nombre de una variable, una propiedad, un objeto, una función o un método. El primer carácter de un identificador debe ser una letra, un carácter de subrayado (_) o un símbolo de dólar ($). Los caracteres posteriores pueden ser números, letras, caracteres de subrayado o símbolos de dólar. |
|---|---|

- Una variable no puede ser una palabra clave ni un literal de ActionScript, como `true`, `false`, `null` o `undefined`. Para más información sobre el uso de literales, consulte "Literales" en la página 94.

- Una variable debe ser exclusiva en su ámbito (consulte "Variables y ámbito" en la página 354).
- Una variable no debe ser ningún elemento del lenguaje ActionScript, como, por ejemplo, un nombre de clase.

Si no sigue estas reglas al asignar nombre a una variable, puede que se produzcan errores de sintaxis o resultados impredecibles. En el siguiente ejemplo, si asigna a una variable el nombre `new` y luego comprueba el documento, Flash generará un error de compilador:

```
// Este código funciona de la forma esperada.
var helloStr:String = new String();
trace(helloStr.length); // 0
// Pero si asigna a una variable el nombre de una clase incorporada...
var new:String = "hello"; //error: Identificador esperado
var helloStr:String = new String();
trace(helloStr.length); // no definido
```

El editor de ActionScript admite las sugerencias de código para las clases incorporadas y para las variables basadas en dichas clases. Si desea que Flash proporcione sugerencias de código para un tipo de objeto determinado que usted asigna a una variable, puede introducir una variable con "strict typing". Las sugerencias para el código ofrecen una sintaxis al estilo de la información sobre herramientas y un menú emergente que le ayuda a escribir el código rápidamente.

Por ejemplo, escriba el código siguiente:

```
var members:Array = new Array();
members.
```

En cuanto escriba el punto (`.`) en el panel Acciones, Flash mostrará una lista de métodos y propiedades disponibles para los objetos Array.

Para conocer las convenciones de codificación recomendadas en la asignación de nombre a las variables, consulte "Asignación de nombre a variables" en la página 780

## Utilización de variables en una aplicación

En esta sección utilizará variables en fragmentos breves de código ActionScript. Debe declarar e inicializar una variable en un script antes de poder utilizarla en una expresión. Las expresiones son combinaciones de operandos y operadores que representan un valor. Por ejemplo, en la expresión `i+2`, `i` y `2` son operandos y + es un operador.

Si no se inicializa una variable antes de utilizarla en una expresión, la variable se queda como no definida y puede generar resultados inesperados. Para más información sobre la escritura de expresiones, consulte el Capítulo 4, "Principios básicos de la sintaxis y el lenguaje", en la página 75.

Si se utiliza una variable no definida, como en el ejemplo siguiente, el valor de la variable en Flash Player 7 y versiones posteriores será `NaN` y el script podría producir resultados imprevistos:

```
var squared:Number = myNum * myNum;
trace(squared); // NaN
var myNum:Number = 6;
```

En el ejemplo siguiente, la sentencia que declara e inicializa la variable `myNum` debe ir al principio, de modo que `squared` pueda sustituirse por un valor:

```
var myNum:Number = 6;
var squared:Number = myNum * myNum;
trace(squared); // 36
```

Ocurre lo mismo al pasar una variable no definida a un método o a una función, como se muestra a continuación.

### Para comparar el paso de variables no definidas y definidas a una función:

1. Arrastre un componente Button desde el panel Componentes al escenario.
2. Abra el inspector de propiedades y escriba **bad_button** en el cuadro de texto Nombre de instancia.
3. Escriba el código siguiente en el fotograma 1 de la línea de tiempo.

```
// No funciona
function badClickListener(evt:Object):Void {
  getURL(targetUrl);
  var targetUrl:String = "http://www.macromedia.com";
}
bad_button.addEventListener("click", badClickListener);
```

4. Seleccione Control > Probar película y observe que el botón no funciona (no abre la página Web).
5. Arrastre otro componente Button al escenario. Seleccione el botón.
6. Abra el inspector de propiedades y escriba **good_button** en el cuadro de texto Nombre de instancia.
7. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo (detrás del código ActionScript que ha añadido anteriormente):

```
// Funciona
function goodClickListener(evt:Object):Void {
  var targetUrl:String = "http://www.macromedia.com";
  getURL(targetUrl);
}
good_button.addEventListener("click", goodClickListener);
```

8. Seleccione Control > Probar película y haga clic en el segundo botón que ha añadido al escenario.

   Este botón abre correctamente la página Web.

El tipo de datos que contiene la variable afecta a cómo y cuándo cambia el valor de la variable. Las variables que contienen datos de tipo simple, como son las cadenas y números, se *pasan por valor*, lo que significa que se utiliza el valor actual de la variable en lugar de una referencia a dicho valor. Ejemplos de tipos de datos complejos son Array (matriz) y Object (objeto).

En el siguiente ejemplo, se establece `myNum` con el valor `15` y se copia el valor en `otherNum`. Al cambiar `myNum` a 30 (en la línea 3 del código), el valor de `otherNum` continúa siendo `15` porque `otherNum` no busca su valor en `myNum`. La variable `otherNum` contiene el valor de `myNum` que recibe (en la línea 2 del código).

### Para utilizar variables en el código ActionScript:

1. Cree un nuevo documento Flash y guárdelo como **var_example.fla**.
2. Seleccione el fotograma 1 de la línea de tiempo y, en el panel Acciones, introduzca el siguiente código:

```
var myNum:Number = 15;
var otherNum:Number = myNum;
myNum = 30;
trace(myNum); // 30
trace(otherNum); // 15
```

   Al cambiar `myNum` a 30 (en la línea 3 del código), el valor de `otherNum` continúa siendo `15` porque `otherNum` no busca su valor en `myNum`. La variable `otherNum` contiene el valor de `myNum` que recibe (en la línea 2 del código).

3. Seleccione Control > Probar película para ver cómo se muestran los valores en el panel Salida.
4. Ahora añada el siguiente código ActionScript tras el código añadido en el paso 2:

```
function sqr(myNum:Number):Number {
  myNum *= myNum;
  return myNum;
}
var inValue:Number = 3;
var outValue:Number = sqr(inValue);
trace(inValue); // 3
trace(outValue); // 9
```

   En este código, la variable `inValue` contiene un valor simple, `3`, de modo que el valor se pasa a la función `sqrt()` y el valor que devuelve es `9`. El valor de la variable `inValue` no cambia, aunque sí cambia el valor de `myNum` en la función.

5. Seleccione Control > Probar película para ver cómo se muestran los valores en el panel Salida.

Los tipos de datos Object pueden incluir una gran cantidad de información tan compleja que una variable con este tipo de dato no contiene el valor real, sino una referencia a un valor. Esta referencia es como un alias que apunta al contenido de la variable. Cuando la variable necesita conocer su valor, la referencia solicita el contenido y devuelve la respuesta sin transferir el valor a la variable.

Para obtener información sobre el paso de una variable por referencia, consulte "Paso de una variable por referencia" en la página 352.

## Paso de una variable por referencia

Dado que los tipos de datos Array y Object contienen una referencia a un valor en lugar de contener el valor en sí, deberá tener cuidado cuando utilice matrices y objetos.

En el siguiente ejemplo se muestra cómo pasar un objeto por referencia. Al crear una copia de la matriz, en realidad sólo se crea una copia de la referencia (o *alias*) al contenido de la matriz. Cuando edite el contenido de la segunda matriz, modificará el contenido tanto de la primera como de la segunda matriz, ya que ambas señalan al mismo valor.

### Para pasar un objeto por referencia:

1. Seleccione Archivo > Nuevo, elija Documento de Flash para crear un nuevo documento FLA y guárdelo como **copybyref.fla**.
2. Seleccione el fotograma 1 de la línea de tiempo y, en el panel Acciones, introduzca el siguiente código:

```
var myArray:Array = new Array("tom", "josie");
var newArray:Array = myArray;
myArray[1] = "jack";
trace(myArray); // tom,jack
trace(newArray); // tom,jack
```

3. Seleccione Control > Probar película para probar el código ActionScript.

   Este código ActionScript crea un objeto Array denominado `myArray` que tiene dos elementos. Debe crear la variable `newArray` y pasar una referencia a `myArray`. Cuando cambie el segundo elemento de `myArray` a `jack`, afectará a cualquier variable que haga referencia al mismo. La sentencia `trace()` envía `tom,jack` al panel Salida.

| NOTA | Flash utiliza un índice basado en cero, lo que significa que 0 es el primer elemento de la matriz, 1 es el segundo y, así, sucesivamente. |
|---|---|

En el siguiente ejemplo, `myArray` contiene un objeto Array, de modo que se pasa la matriz a la función `zeroArray()` por referencia. La función `zeroArray()` acepta un objeto Array como parámetro y define el valor 0 para todos los elementos de dicha matriz. Puede modificar la matriz porque se pasa por referencia.

### Para pasar una matriz por referencia:

1. Seleccione Archivo > Nuevo, elija Documento de Flash para crear un nuevo archivo FLA y guárdelo como **arraybyref.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
function zeroArray (theArr:Array):Void {
  var i:Number;
  for (i = 0; i < theArr.length; i++) {
    theArr[i] = 0;
  }
}

var myArr:Array = new Array();
myArr[0] = 1;
myArr[1] = 2;
myArr[2] = 3;
trace(myArr); // 1,2,3
zeroArray(myArr);
trace(myArr); // 0,0,0
```

3. Seleccione Control > Probar película para probar el código ActionScript.

   La primera sentencia `trace()` de este código ActionScript muestra el contenido original de la matriz `myArray` (`1,2,3`). Tras llamar a la función `zeroArray()` y pasar una referencia a la matriz `myArray`, todos los valores de matriz se sobrescriben y se establecen en cero. La siguiente sentencia `trace()` muestra el nuevo contenido de la matriz `myArray` (`0,0,0`). Dado que se pasa la matriz por referencia y no el valor en sí, no es necesario devolver el contenido actualizado de la matriz desde dentro de la función `zeroArray()`.

Para más información sobre matrices, consulte "Matrices" en la página 129.

## Variables y ámbito

El ámbito de una variable se refiere al área en la que se conoce la variable (área en la que está *definida*) y se puede hacer referencia a ella. El área en la que se conoce la variable puede estar dentro de una determinada línea de tiempo o dentro de una función, o podría conocerse globalmente en toda la aplicación. Para más información sobre el ámbito, consulte "Ámbito y referencias" en la página 86.

Comprender en qué consiste el ámbito de las variables es importante a la hora de desarrollar aplicaciones Flash con ActionScript. El ámbito no sólo indica cuándo y dónde puede hacer referencia a las variables, sino también el tiempo durante el cual una determinada variable existe en una aplicación. Al definir variables en el cuerpo de una función, éstas dejan de existir al finalizar la función especificada. Si intenta hacer referencia a objetos de un ámbito erróneo o a variables que han caducado, aparecerán errores en los documentos de Flash que provocarán un comportamiento inesperado o que no esté disponible una determinada funcionalidad.

ActionScript contiene tres tipos de ámbitos de variable:

- Las funciones y las Variables globales son visibles para todas las líneas de tiempo y todos los ámbitos del documento. Por consiguiente, una variable global se define en todas las áreas del código.
- Las Variables de línea de tiempo están disponibles para cualquier script de dicha línea de tiempo.
- Las Variables locales están disponibles en el cuerpo de la función en la que se han declarado (incluidas entre llaves). Por consiguiente, las variables locales sólo se definen en una parte del código.

Para obtener directrices de utilización de ámbitos y variables, consulte el Capítulo 4, "Ámbito y referencias", en la página 86.

NOTA

Las clases de ActionScript 2.0 que crea el usuario son compatibles con los ámbitos de variable público, privado y estático. Para más información, consulte "Miembros de clase" en la página 221 y "Control del acceso de miembros en las clases" en la página 244.

No puede utilizar "strict data typing" en las variables globales. Para más información, consulte "Variables globales" en la página 355.

## Variables globales

Las funciones y las variables globales son visibles para todas las líneas de tiempo y todos los ámbitos del documento. Para declarar (o *crear*) una variable de ámbito global, escriba el identificador `_global` delante del nombre de la variable y no utilice la sintaxis `var =`. Por ejemplo, el código siguiente crea la variable global `myName`:

```
var _global.myName = "George"; // Sintaxis incorrecta para la variable
  global
_global.myName = "George"; // Sintaxis correcta para la variable global
```

Sin embargo, si inicializa una variable local con el mismo nombre que la variable global, no tendrá acceso a la variable global mientras se encuentre en el ámbito de la variable local, como se muestra en el siguiente ejemplo:

```
_global.counter = 100; // Declara la variable global
trace(counter); // Accede a la variable global y muestra 100
function count():Void {
  for (var counter:Number = 0; counter <= 2; counter++) { // Variable local
    trace(counter); // Accede a la variable local y muestra de 0 a 2
  }
}
count();
trace(counter); // Accede a la variable global y muestra 100
```

Este ejemplo muestra simplemente que no se accede a la variable global en el ámbito de la función `count()`. No obstante, puede acceder a la variable de ámbito global si incluye el prefijo `_global`. Por ejemplo, puede acceder a ella si antepone al contador el prefijo `_global`, como se muestra en el siguiente código:

```
trace(_global.counter);
```

No se puede asignar "strict data typing" a variables que crea en el ámbito _global, ya que tiene que utilizar la palabra clave var al asignar un tipo de datos. Por ejemplo, no podría hacer:

```
_global.foo:String = "foo";  //error de sintaxis
var _global.foo:String = "foo"; //error de sintaxis
```

El entorno limitado de seguridad de Flash Player versión 7 y posteriores aplica restricciones cuando se accede a variables globales desde archivos SWF cargados desde otros dominios de seguridad. Para más información, consulte el Capítulo 17, "Aspectos básicos de la seguridad", en la página 715.

## Variables de línea de tiempo

Las variables de línea de tiempo están disponibles para cualquier script de dicha línea de tiempo. Para declarar variables de línea de tiempo, utilice la sentencia `var` e inicialícelas en cualquier fotograma de la línea de tiempo. La variable está disponible para dicho fotograma y todos los fotogramas posteriores, como se muestra en el ejemplo siguiente.

### Para utilizar variables de línea de tiempo en un documento:

1. Cree un nuevo documento de Flash y asígnele el nombre **timelinevar.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
var myNum:Number = 15; /* inicializado en el fotograma 1, por lo que está
  disponible para todos los fotogramas */
```

3. Seleccione el fotograma 20 de la línea de tiempo.
4. Seleccione Insertar > Línea de tiempo > Fotograma clave vacío.
5. Con el nuevo fotograma clave seleccionado, escriba el siguiente código ActionScript en el panel Acciones:

```
trace(myNum);
```

6. Seleccione Control > Probar película para probar el nuevo documento.

   El valor 15 aparece en el panel Salida después de un segundo aproximadamente. Dado que los documentos de Flash se reproducen indefinidamente de manera predeterminada, el valor 15 aparece continuamente en el panel Salida cada vez que la cabeza lectora llega al fotograma 20 de la línea de tiempo. Para detener la acción de bucle, añada `stop();` tras la sentencia `trace()`.

Debe asegurarse de que declara una variable de línea de tiempo antes de intentar acceder a ella en un script. Por ejemplo, si coloca el código `var myNum:Number = 15;` en el fotograma 20, todos los scripts asociados a un fotograma antes del fotograma 20 no podrán acceder a `myNum` y permanecen sin definir en lugar de contener el valor `15`.

## Variables locales

Al utilizar la sentencia `var` dentro de un bloque de función, se declaran las *variables locales*. Al declarar una variable local dentro de un bloque de función (también conocido como *definición de función*), se define dentro del ámbito del bloque de función y caduca al final de la función de bloque. Por consiguiente, la variable local sólo existe dentro de dicha función.

Por ejemplo, si declara una variable denominada `myStr` dentro de una función denominada `localScope`, dicha variable no estará disponible fuera de la función.

```
function localScope():Void {
  var myStr:String = "local";
}
localScope();
trace(myStr); // No definida, ya que myStr no está definida globalmente
```

Si el nombre de la variable que utiliza para su variable local ya está declarada como variable de línea de tiempo, la definición local tendrá prioridad sobre la definición de línea de tiempo mientras la variable local se encuentre en el ámbito. La variable de línea de tiempo continuará existiendo fuera de la función. Por ejemplo, el siguiente código crea una variable de cadena de línea de tiempo denominada `str1` y luego crea una variable local con el mismo nombre dentro de la función `scopeTest()`. La sentencia `trace` situada dentro de la función genera la definición local de la variable, pero la sentencia `trace` situada fuera de la función genera la definición de línea de tiempo de la variable.

```
var str1:String = "Timeline";
function scopeTest():Void {
  var str1:String = "Local";
  trace(str1); // Local
}
scopeTest();
trace(str1); // Línea de tiempo
```

En el siguiente ejemplo se observa cómo determinadas variables sólo existen mientras existe una función determinada y pueden provocar errores si intenta hacer referencia a la variable fuera del ámbito de dicha función.

### Para utilizar variables locales en una aplicación:

1. Cree un nuevo documento de Flash.
2. Abra el panel Acciones (Ventana > Acciones) y añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo:

   ```
   function sayHello(nameStr:String):Void {
     var greetingStr:String = "Hello, " + nameStr;
     trace(greetingStr);
   }
   sayHello("world"); // Hello, world (Hola, mundo)
   trace(nameStr); // undefined (no definido)
   trace(greetingStr); // undefined (no definido)
   ```

3. Seleccione Control > Probar película para probar el documento.

   Flash muestra la cadena “Hello, world” en el panel Salida y muestra `undefined` para los valores de `nameStr` y `greetingStr` porque las variables ya no están disponibles en el ámbito actual. Sólo puede hacer referencia a `nameStr` y `greetingStr` en la ejecución de la función `sayHello`. Al terminar la función, las variables dejan de existir.

Las variables `i` y `j` se utilizan con frecuencia como contadores de bucles. En el siguiente ejemplo, `i` se utiliza como variable local; solamente existe dentro de la función `initArray()`:

```
var myArr:Array = new Array();
function initArray(arrayLength:Number):Void {
  var i:Number;
  for(i = 0; i < arrayLength; i++) {
    myArr[i] = i + 1;
  }
}
trace(myArr); // <blank>
initArray(3);
trace(myArr); // 1,2,3
trace(i); // undefined (no definido)
```

**NOTA** También es habitual ver la siguiente sintaxis para un bucle `for`: `for (var i:Number = 0; i < arrayLength; i++) {...}`.

Este ejemplo muestra `undefined` en el entorno de prueba de Flash porque la variable `i` no está definida en la línea de tiempo principal. Sólo existe en la función `initArray()`.

Puede utilizar las variables locales para evitar conflictos de nombres, que pueden originar errores inesperados en la aplicación. Por ejemplo, si utiliza `age` como variable local, podría utilizarla para almacenar la edad de una persona en un contexto y la edad del hijo de una persona en otro contexto. No existe conflicto en esta situación porque está utilizando estas variables en ámbitos independientes.

Es usual y recomendable utilizar variables locales en el cuerpo de una función de modo que ésta pueda actuar como un segmento de código independiente. Sólo puede cambiar una variable local dentro de su propio bloque de código. Si la expresión contenida en una función utiliza una variable global, código o eventos externos a la función podría modificar su valor, lo cual cambiaría la función.

Puede asignar un tipo de datos a una variable local al definirla, lo cual impide asignar el tipo de datos incorrecto a una variable existente. Para más información, consulte “Asignación de tipos de datos y “strict data typing”” en la página 337.

## Carga de variables

En las siguientes secciones, cargará variables del servidor de diversas formas o en un documento de una cadena de URL o FlashVars (puede utilizar FlashVars para pasar variables a Flash) en el código HTML. Estas prácticas demuestran que existen diversas formas de utilizar variables fuera de un archivo SWF.

Encontrará más información sobre la carga de variables (como, por ejemplo, pares nombre/valor) en el Capítulo 16, “Trabajo con datos externos”, en la página 671.

Puede utilizar variables de diversas formas en un archivo SWF, dependiendo de para qué necesite dichas variables. Para más información, consulte los siguientes temas:


- "Utilización de variables desde la URL" en la página 359
- "Utilización de FlashVars en una aplicación" en la página 362
- "Carga de variables desde un servidor" en la página 363


## Utilización de variables desde la URL

Al desarrollar una aplicación o un ejemplo sencillo en Flash, es posible que desee pasar valores de una página HTML al documento de Flash. Los valores transferidos se conocen a veces como la *cadena de consulta* o *variables con codificación URL*. Las variables de URL resultan útiles, por ejemplo, si desea crear un menú en Flash. Puede inicializar el menú para mostrar la navegación correcta de manera predeterminada. O bien puede crear un visor de imágenes en Flash y definir la imagen predeterminada que debe mostrarse en el sitio Web.

### Para utilizar variables de URL en un documento:

1. Cree un documento de Flash y asígnele el nombre **urlvariables.fla**.
2. Seleccione Archivo > Guardar como y guarde el documento en el escritorio.
3. Seleccione el fotograma 1 de la línea de tiempo y añada el siguiente código en el panel Acciones:

```
this.createTextField("myTxt", 100, 0, 0, 100, 20);
myTxt.autoSize = "left";
myTxt.text = _level0.myURL;
```

4. Seleccione Control > Probar película para probar el archivo SWF en Flash Player.

   El campo de texto muestra `undefined`. Si desea asegurarse de que las variables están bien definidas antes de continuar, deberá comprobar la existencia de las variables en Flash. Para ello, puede comprobar si están sin definir (undefined).
5. Para comprobar si la variable está definida, modifique el código ActionScript que ha añadido al panel Acciones en el paso 3 para que coincida con el siguiente código. Añada el código que aparece en **negrita**:

```
this.createTextField("myTxt", 100, 0, 0, 100, 20);
myTxt.autoSize = "left";
if (_level0.myURL == undefined) {
  myTxt.text = "myURL is not defined";
} else {
  myTxt.text = _level0.myURL;
}
```

Al publicar el documento de Flash, se crea un documento HTML de manera predeterminada en el mismo directorio que el archivo SWF. Si no se crea un archivo HTML, seleccione Archivo > Configuración de publicación y asegúrese de que selecciona HTML en la ficha Formatos. Seguidamente, publique de nuevo el documento.

El siguiente código muestra el código HTML del documento responsable de la incorporación de un documento de Flash a una página HTML. Debe examinar este código HTML para comprender cómo funcionan las variables de URL en el siguiente paso (donde añade código adicional para las variables de URL).

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
  codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/
  swflash.cab#version=8,0,0,0" width="550" height="400"
  id="urlvariables" align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="urlvariables.swf" />
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<embed src="urlvariables.swf" quality="high" bgcolor="#ffffff"
  width="550" height="400" name="urlvariables" align="middle"
  allowScriptAccess="sameDomain" type="application/x-shockwave-flash"
  pluginspage="http://www.macromedia.com/go/getflashplayer" />
</object>
```

6. Para pasar variables del documento HTML generado al documento de Flash, puede pasar variables tras la ruta y el nombre de archivo (urlvariables.swf). Añada el **texto en negrita** al archivo HTML que se ha generado en el escritorio.

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
  codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/
  swflash.cab#version=8,0,0,0" width="550" height="400"
  id="urlvariables" align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="urlvariables.swf?myURL=http://
  weblogs.macromedia.com" />
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<embed src="urlvariables.swf?myURL=http://weblogs.macromedia.com"
  quality="high" bgcolor="#ffffff" width="550" height="400"
  name="urlvariables" align="middle" allowScriptAccess="sameDomain"
  type="application/x-shockwave-flash" pluginspage="http://
  www.macromedia.com/go/getflashplayer" />
</object>
```

7. Si desea pasar varias variables a Flash, deberá separar los pares nombre/valor mediante un carácter ampersand (&). Localice el siguiente código del paso 6:

```
?myURL=http://weblogs.macromedia.com
```

Sustitúyalo por el siguiente texto:

```
?myURL=http://weblogs.macromedia.com&myTitle=Macromedia+News+Aggregator
```

Recuerde que debe realizar los mismos cambios tanto en la etiqueta `object` como en la etiqueta `embed` para mantener la coherencia entre todos los navegadores. Puede que observe que las palabras están separadas por signos +. Las palabras están separadas de esta forma porque los valores tienen codificación URL y el signo + representa un espacio en blanco sencillo.

| NOTA | Para obtener una lista de caracteres especiales con codificación de URL comunes, consulte la Nota técnica de Flash, URL Encoding: Reading special characters from a text file. |
|---|---|

Dado que el ampersand (`&`) actúa a modo de delimitador entre los diferentes pares nombre/valor, si los valores que desea pasar contienen ampersands, podrían producirse resultados inesperados. Dada la naturaleza de los pares nombre/valor y del análisis, si pasara los siguientes valores a Flash:

```
my.swf?name=Ben+&+Jerry&flavor=Half+Baked
```

Flash crearía las siguientes variables (y valores) en el ámbito raíz:

```
'name': 'Ben ' (note space at end of value)
' Jerry': '' (note space at beginning of variable name and an empty
  value)
'flavor': 'Half Baked'
```

Para evitar esto, deberá anular (*escape*) el carácter ampersand (`&`) del par nombre/valor con su equivalente con codificación URL (`%26`).

8. Abra el documento urlvariables.html y localice el siguiente código:

```
?myURL=http://weblogs.macromedia.com&myTitle=Macromedia+News+Aggregator
```

Sustitúyalo por el siguiente código:

```
?myURL=Ben+%26+Jerry&flavor=Half+Baked
```

9. Guarde el HTML revisado y compruebe de nuevo el documento de Flash.

Observará que Flash ha creado los siguientes pares nombre/valor.

```
'name': 'Ben & Jerry'
'flavor': 'Half Baked'
```

| NOTA | Todos los navegadores admiten cadenas de hasta 64 K (65535 bytes) de longitud. FlashVars debe asignarse tanto en la etiqueta `object` como en la etiqueta `embed` para que funcione en todos navegadores. |
|---|---|

## Utilización de FlashVars en una aplicación

La utilización de FlashVars para pasar variables a Flash es parecido a pasar variables en la URL en el código HTML. Con FlashVars, en lugar de pasar variables después del nombre de archivo, éstas se pasan en una etiqueta `param` independiente, así como en la etiqueta `embed`.

### Para utilizar FlashVars en un documento:

1. Cree un nuevo documento de Flash y asígnele el nombre **myflashvars.fla**.
2. Seleccione Archivo > Configuración de publicación, asegúrese de que está activado HTML y, a continuación, haga clic en Aceptar para cerrar el cuadro de diálogo.
3. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
this.createTextField("myTxt", 100, 0, 0, 100, 20);
myTxt.autoSize = "left";
if (_level0.myURL == undefined) {
  myTxt.text = "myURL is not defined";
} else {
  myTxt.text = _level0.myURL;
}
```

> **NOTA** De manera predeterminada, el código HTML se publica en la misma ubicación que myflashvars.fla.

4. Seleccione Archivo > Publicar para publicar los archivos SWF y HTML.
5. Abra el directorio que contiene los archivos publicados (el lugar del disco duro en el que guardó myflashvars.fla) y abra el documento HTML (de manera predeterminada, myflashvars.html) en un editor HTML como Dreamweaver o el Bloc de notas.
6. Añada el código que aparece debajo en **negrita** para que el documento HTML coincida con lo siguiente:

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
  codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/
  swflash.cab#version=8,0,0,0" width="550" height="400" id="myflashvars"
  align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="myflashvars.swf" />
<param name="FlashVars" value="myURL=http://weblogs.macromedia.com/">
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<embed src="myflashvars.swf" FlashVars="myURL=http://
  weblogs.macromedia.com/" quality="high" bgcolor="#ffffff" width="550"
  height="400" name="myflashvars" align="middle"
  allowScriptAccess="sameDomain" type="application/x-shockwave-flash"
  pluginspage="http://www.macromedia.com/go/getflashplayer" />
</object>
```

Este código pasa una variable sencilla denominada `myURL` que contiene la cadena `http://weblogs.macromedia.com`. Cuando se carga el archivo SWF, se crea una propiedad llamada `myURL` en el ámbito _level0. Una de las ventajas de utilizar FlashVars o pasar variables en la URL es que las variables están disponibles de manera inmediata en Flash cuando se carga el archivo SWF. Esto significa que no tiene que escribir ninguna función para comprobar si las variables han terminado de cargarse, lo que sí tendría que hacer si cargara las variables mediante LoadVars o XML.

7. Guarde los cambios en el documento HTML y luego ciérrelo.
8. Haga doble clic en myflashvars.html para comprobar la aplicación.

   El texto `http://weblogs.macromedia.com`, una variable en el archivo HTML, aparece en el archivo SWF.

| NOTA | Todos los navegadores admiten cadenas de hasta 64 K (65.535 bytes) de longitud. FlashVars debe asignarse tanto en la etiqueta `object` como en la etiqueta `embed` para que funcione en todos navegadores. |
|---|---|

## Carga de variables desde un servidor

Existen varias formas de cargar variables en Flash desde fuentes externas (como, por ejemplo, archivos de texto, documentos XML, etc.). Encontrará más información sobre la carga de variables, incluidos los pares nombre/valor, en el Capítulo 16, “Trabajo con datos externos”, en la página 671.

En Flash, puede cargar variables fácilmente utilizando la clase LoadVars, como se muestra en el siguiente ejemplo.

### Para cargar variables desde un servidor:

1. Cree un nuevo documento de Flash.
2. Seleccione el fotograma 1 de la línea de tiempo y añada el siguiente código ActionScript en el panel Acciones:

```
var my_lv:LoadVars = new LoadVars();
my_lv.onLoad = function(success:Boolean):Void {
  if (success) {
    trace(this.dayNames); // Sunday,Monday,Tuesday,...
  } else {
    trace("Error");
  }
}
my_lv.load("http://www.helpexamples.com/flash/params.txt");
```

Este código carga un archivo de texto de un servidor remoto y analiza los pares nombre/valor.

| SUGERENCIA | Descargue o vea el archivo de texto (http://www.helpexamples.com/flash/params.txt) en un navegador si desea saber cómo se formatean las variables. |
|---|---|

3. Seleccione Control > Probar película para probar el documento.

   Si el archivo se carga correctamente, se llama al evento `complete` y el panel Salida muestra el valor de `dayNames`. Si el archivo de texto no puede descargarse, el argumento `success` se establece con el valor `false` y el panel Salida muestra el texto `Error`.

## Utilización de variables en un proyecto

Al crear animaciones o aplicaciones con Flash, existen muy pocas situaciones en las que no sea necesario utilizar ninguna clase de variable en el proyecto. Por ejemplo, si crea un sistema de inicio de sesión, es posible que necesite variables para determinar si el nombre del usuario y la contraseña son válidos o si se han facilitado dichos datos.

Encontrará más información sobre la carga de variables (como, por ejemplo, pares nombre/valor) en el Capítulo 16, “Trabajo con datos externos”, en la página 671.

En el siguiente ejemplo, las variables se utilizan para almacenar la ruta de una imagen que va a cargar con la clase Loader, una variable para la instancia de la clase Loader y varias funciones a las que se llama en función de si el archivo se carga correctamente o no.

### Para utilizar variables en un proyecto:

1. Cree un nuevo documento de Flash y guárdelo como **imgloader.fla**.
2. Seleccione el fotograma 1 de la línea de tiempo y añada el siguiente código ActionScript en el panel Acciones:

```
/* Especificar la imagen predeterminada por si no se pasa ningún valor
  mediante FlashVars. */
var imgUrl:String = "http://www.helpexamples.com/flash/images/
  image1.jpg";
if (_level0.imgURL != undefined) {
  // Si se ha especificado la imagen, se sobrescribe el valor
  predeterminado.
  imgUrl = _level0.imgURL;
}
```

```
this.createEmptyMovieClip("img_mc", 10);
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip):Void {
  target_mc._x = (Stage.width - target_mc._width) / 2;
  target_mc._y = (Stage.height - target_mc._height) / 2;
}
mclListener.onLoadError = function(target_mc:MovieClip):Void {
  target_mc.createTextField("error_txt", 1, 0, 0, 100, 20);
  target_mc.error_txt.autoSize = "left";
  target_mc.error_txt.text = "Error downloading specified image;\n\t" +
  target_mc._url;
}
var myMCL:MovieClipLoader = new MovieClipLoader();
myMCL.addListener(mclListener);
myMCL.loadClip(imgUrl, img_mc);
```

La primera línea de código especifica la imagen que desea cargar dinámicamente en el documento de Flash. Seguidamente, deberá comprobar si se ha especificado un nuevo valor para `imgURL` mediante FlashVars o variables con codificación URL. Si se ha especificado un nuevo valor, éste nuevo valor sobrescribirá la URL de la imagen predeterminada. Para más información sobre la utilización de variables de URL, consulte "Utilización de variables desde la URL" en la página 359. Para obtener información sobre FlashVars, consulte "Utilización de FlashVars en una aplicación" en la página 362.

Las siguientes líneas del código definen la instancia de MovieClip y un objeto detector para la futura instancia de MovieClipLoader. El objeto detector de MovieClipLoader define dos controladores de eventos, `onLoadInit` y `onLoadError`. Los controladores se invocan cuando la imagen se carga correctamente y se inicializa en el escenario o si se produce un error al cargar la imagen. Seguidamente, se crea una instancia de MovieClipLoader y se utiliza el método `addListener()` para añadir el objeto detector previamente definido a MovieClipLoader. Finalmente, la imagen se descarga y se activa al llamar al método `MovieClipLoader.loadClip()`, que especifica el archivo de imagen que debe cargarse y el clip de película de destino en el que debe cargarse la imagen.

3. Seleccione Control > Probar película para probar el documento.

   Dado que está comprobando el documento de Flash en la herramienta de edición, no se pasará ningún valor para `imgUrl` mediante FlashVars ni en la URL, por lo que se mostrará la imagen predeterminada.

4. Guarde el documento de Flash y seleccione Archivo > Publicar para publicar el archivo como documento SWF y HTML.

| NOTA | Asegúrese de que Flash y HTML están seleccionados en el cuadro de diálogo Configuración de publicación. Seleccione Archivo > Configuración de publicación y elija la ficha Formatos. A continuación, seleccione ambas opciones. |
|---|---|

5. Si comprueba el documento en la herramienta Flash (seleccione Control > Probar película) o en un navegador local (Archivo > Vista previa de publicación > HTML), observará que la imagen se centra vertical y horizontalmente en el escenario.
6. Edite el documento HTML generado en un editor (como Dreamweaver o el Bloc de notas) y modifique el código HTML para que coincida con el siguiente texto:

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
  codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/
  swflash.cab#version=8,0,0,0" width="550" height="400"
  id="urlvariables" align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="urlvariables.swf" />
<param name="FlashVars" value="imgURL=http://www.helpexamples.com/flash/
  images/image2.jpg">
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<embed src="urlvariables.swf" quality="high" FlashVars="imgURL=http://
  www.helpexamples.com/flash/images/image2.jpg" bgcolor="#ffffff"
  width="550" height="400" name="urlvariables" align="middle"
  allowScriptAccess="sameDomain" type="application/x-shockwave-flash"
  pluginspage="http://www.macromedia.com/go/getflashplayer" />
</object>
```

7. Pruebe el documento HTML para ver los cambios. La imagen que especifique en el código HTML aparecerá en el archivo SWF.

   Para modificar este ejemplo y utilizar sus propias imágenes, modificaría el valor FlashVars (la cadena dentro de las comillas dobles).

# Organización de datos en objetos

Es posible que ya esté acostumbrado a utilizar los objetos que coloca en el escenario. Por ejemplo, es posible que tenga un objeto MovieClip en el escenario y que este objeto contenga otros clips de película dentro de él. Los campos de texto, los clips de película y los botones suelen denominarse objetos cuando se colocan en el escenario.

En ActionScript, los objetos son conjuntos de propiedades y métodos. Cada objeto tiene su propio nombre y es una instancia de una clase concreta. Los objetos incorporados pertenecen a clases que están predefinidas en ActionScript. Por ejemplo, la clase incorporada Date ofrece información procedente del reloj del sistema del equipo del usuario. Puede utilizar la clase incorporada LoadVars para cargar variables en el archivo SWF.

También puede crear objetos y clases mediante ActionScript. Puede crear un objeto que contenga un conjunto de datos, como, por ejemplo, el nombre, la dirección y el número de teléfono de una persona. Puede crear un objeto que contenga información de color para una imagen. La organización de datos en objetos puede ayudar a mantener los documentos de Flash mejor organizados. Para obtener información general sobre la creación de una clase personalizada que contenga un conjunto de métodos y propiedades, consulte "Escritura de archivos de clases personalizadas" en la página 205. Para obtener información detallada sobre clases incorporadas y personalizadas, consulte el Capítulo 6, "Clases", en la página 195

Existen varias formas de crear un objeto en ActionScript. En el siguiente ejemplo se crean objetos sencillos de dos formas diferentes y luego se reproduce el contenido de dichos objetos de forma indefinida.

### Para crear objetos sencillos en Flash:

1. Cree un nuevo documento de Flash y guárdelo como **simpleObjects.fla**.
2. Seleccione el fotograma 1 de la línea de tiempo y escriba el siguiente código ActionScript en el panel Acciones:

```
// La primera forma
var firstObj:Object = new Object();
firstObj.firstVar = "hello world";
firstObj.secondVar = 28;
firstObj.thirdVar = new Date(1980, 0, 1); // January 1, 1980
```

   Este código, que es una forma de crear un objeto sencillo, crea una nueva instancia de objeto y define una serie de propiedades dentro del objeto.

3. Ahora introduzca el siguiente código ActionScript tras el código introducido en el paso 2.

```
// La segunda forma
var secondObj:Object = {firstVar:"hello world", secondVar:28,
  thirdVar:new Date(1980, 0, 1)};
```

   Esta es otra forma de crear un objeto. Ambos objetos son equivalentes. Este código crea un nuevo objeto e inicializa algunas propiedades utilizando la notación abreviada de objeto.

4. Para reproducir de forma indefinida cada uno de los objetos anteriores y mostrar el contenido de los objetos, añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo (detrás del código que ya ha introducido):

```
var i:String;
for (i in firstObj) {
  trace(i + ": " + firstObj[i]);
}
```

5. Seleccione Control > Probar película para que aparezca el siguiente texto en el panel Salida.

```
firstVar: hello world
secondVar: 28
thirdVar: Tue Jan 1 00:00:00 GMT-0800 1980
```

También puede utilizar matrices para crear objetos. En lugar de utilizar una serie de variables, como `firstname1`, `firstname2` y `firstname3`, para representar un conjunto de variables, puede crear una matriz de objetos para representar los mismos datos. A continuación se ofrece una demostración de esta técnica.

### Para crear un objeto mediante una matriz:

1. Cree un nuevo documento de Flash y guárdelo como **arrayObject.fla**.
2. Seleccione el fotograma 1 de la línea de tiempo y escriba el siguiente código ActionScript en el panel Acciones:

```
var usersArr:Array = new Array();
usersArr.push({firstname:"George"});
usersArr.push({firstname:"John"});
usersArr.push({firstname:"Thomas"});
```

   La ventaja de organizar variables en matrices y objetos es que resulta mucho más fácil reproducir indefinidamente las variables y ver los valores, como se muestra en el paso siguiente.

3. Escriba el siguiente código detrás del código ActionScript añadido en el paso 2.

```
var i:Number;
for (i = 0; i < usersArr.length; i++) {
  trace(usersArr[i].firstname); // George, John, Thomas
}
```

4. Seleccione Control > Probar película para que aparezca el siguiente texto en el panel Salida:

```
George
John
Thomas
```

En el siguiente ejemplo se presenta otra forma de reproducir objetos indefinidamente. En este ejemplo, se crea un objeto y se reproduce indefinidamente mediante un bucle `for..in` y cada propiedad aparece en el panel Salida:

```
var myObj:Object = {var1:"One", var2:"Two", var3:18, var4:1987};
var i:String;
for (i in myObj) {
  trace(i + ": " + myObj[i]);
}
//genera el siguiente resultado:
/*
    var1: One
    var2: Two
    var3: 18
    var4: 1987
*/
```

Para obtener información sobre la creación de bucles, consulte el Capítulo 4, “Utilización de bucles for”, en la página 123. Para obtener información sobre bucles for...in, consulte “Utilización de bucles for..in” en la página 124. Para más información sobre objetos, consulte el Capítulo 6, “Clases”, en la página 195.

# Conversión

ActionScript 2.0 permite convertir un tipo de datos en otro. Convertir un objeto a un tipo diferente consiste en convertir el valor que contiene el objeto o la variable a un tipo diferente.

Los resultados de una conversión de tipo dependen de los tipos de datos de que se trate. Para convertir un objeto a otro tipo, deberá incluir el nombre del objeto entre paréntesis (`()`) y anteponerle el nombre del nuevo tipo. Por ejemplo, el siguiente código toma un valor booleano y lo convierte en un entero.

```
var myBoolean:Boolean = true;
var myNumber:Number = Number(myBoolean);
```

Para más información sobre conversiones, consulte los temas siguientes:

- “Conversión de objetos” en la página 370

## Conversión de objetos

La sintaxis de conversión es `type(item)`, donde se pretende que el compilador se comporte como si el tipo de datos de item fuese `type`. La conversión es básicamente una llamada de función que devuelve `null` si la conversión falla en tiempo de ejecución (lo que ocurre en archivos publicados para Flash Player 7 o posterior; los archivos publicados para Flash Player 6 no contemplan errores de conversión en tiempo de ejecución). Si la conversión se lleva a cabo correctamente, la llamada de función devuelve el objeto original. No obstante, el compilador no puede determinar si una conversión fallará en tiempo de ejecución, por lo que no genera errores de tiempo de compilación en estos casos.

El siguiente código muestra un ejemplo:

```
// Both the Cat and Dog classes are subclasses of the Animal class
function bark(myAnimal:Animal) {
  var foo:Dog = Dog(myAnimal);
  foo.bark();
}
var curAnimal:Animal = new Dog();
bark(curAnimal); // Funciona
curAnimal = new Cat();
bark(curAnimal); // No funciona
```

En este ejemplo, ha afirmado al compilador que `foo` es un objeto Dog y, por lo tanto, el compilador da por hecho que `temp.bark()` es una sentencia válida. Sin embargo, el compilador no sabe que la conversión fallará (es decir, que ha intentado convertir un objeto Cat en un tipo Animal), por lo que no se produce ningún error de tiempo de compilación. No obstante, si incluye una comprobación en el script para asegurarse de que la conversión se lleva a cabo correctamente, se podrán detectar errores de conversión durante la ejecución, como se muestra en el siguiente ejemplo.

```
function bark(myAnimal:Animal) {
  var foo:Dog = Dog(myAnimal);
  if (foo) {
    foo.bark();
  }
}
```

Puede convertir una expresión en una interfaz. Si la expresión es un objeto que implementa la interfaz, o tiene una clase base que implementa la interfaz, la conversión se realiza correctamente. De no ser así, la conversión falla.

| NOTA | Convertir a null o undefined devuelve `undefined`. |
|---|---|

No puede anular los tipos de datos simples que se corresponden con una función de conversión global con un operador de conversión del mismo nombre. Esto se debe a que las funciones de conversión global tienen prioridad sobre los operadores de conversión. Por ejemplo, no puede convertir a Array porque la función de conversión `Array()` tiene prioridad sobre el operador de conversión.

En este ejemplo se definen dos variables de cadena (`firstNum` y `secondNum`) que se unen. El resultado inicial es que los números se concatenan en lugar de sumarse porque su tipo de datos es String (cadena). La segunda sentencia trace convierte ambos números al tipo de datos Number antes de realizar la suma que proporciona el resultado correcto. La conversión de datos es importante cuando se utilizan datos cargados mediante XML o FlashVars, como se muestra en el siguiente ejemplo:

```
var firstNum:String = "17";
var secondNum:String = "29";
trace(firstNum + secondNum); // 1729
trace(Number(firstNum) + Number(secondNum)); // 46
```

Para más información sobre funciones de conversión, consulte la entrada de cada función de conversión en *Referencia del lenguaje ActionScript 2.0*: `%{Array function}%`, `%{Boolean function}%`, `%{Number function}%`, `%{Object function}%` y `%{String function}%`.

CAPÍTULO 11

# Trabajo con clips de película

11

Los clips de película son una especie de archivos SWF con contenido propio que se ejecutan de manera independiente y al margen de la línea de tiempo en la que se encuentran. Por ejemplo, si la línea de tiempo principal sólo tiene un fotograma y un clip de película de dicho fotograma tiene diez fotogramas, cuando se reproduzca el archivo SWF principal se reproducirán todos los fotogramas del clip de película. Un clip de película puede, a su vez, contener otros clips de película o *clips anidados*. Los clips de película anidados de esta manera tienen una relación jerárquica, en la cual el *clip principal* contiene uno o varios *clips secundarios*.

Puede asignar nombre a las instancias de clip de película para identificarlas de manera unívoca como objetos que pueden controlarse con ActionScript. Cuando se asigna un *nombre de instancia* a una instancia de un clip de película, el nombre de instancia la identifica como objeto de clase MovieClip. Para controlar el aspecto y el comportamiento de los clips de película en tiempo de ejecución, utilice las propiedades y los métodos de la clase MovieClip.

Los clips de película vienen a ser objetos autónomos que pueden responder a eventos, enviar mensajes a otros objetos de clip de película, mantener su estado y controlar sus clips secundarios. De este modo, los clips de película proporcionan la base de una *arquitectura basada en componentes* en Macromedia Flash Basic 8 y Macromedia Flash Professional 8. De hecho, los componentes disponibles en el panel Componentes (Ventana > Componentes) son clips de película sofisticados que han sido diseñados y programados para tener un aspecto y un comportamiento concretos.

Para más información sobre la utilización de la API de dibujo (métodos de dibujo de la clase MovieClip), filtros, mezclas, animación mediante scripts, etc., consulte el Capítulo 13, “Animaciones, filtros y dibujos”

Para más información sobre clips de película, consulte los temas siguientes:

# Control de clips de película con ActionScript

Para realizar tareas en clips de película, puede utilizar las funciones globales ActionScript o los métodos de la clase MovieClip. Algunos métodos de la clase MovieClip realizan las mismas tareas que las funciones del mismo nombre; otros métodos MovieClip, tales como `hitTest()` y `swapDepths()`, no tienen nombres de función correspondientes.

En el ejemplo siguiente se muestra la diferencia entre el uso de un método y de una función. Cada sentencia duplica la instancia `my_mc`, asigna el nombre `new_mc` al nuevo clip y lo coloca a una profundidad de 5.

```
my_mc.duplicateMovieClip("new_mc", 5);
duplicateMovieClip(my_mc, "new_mc", 5);
```

Si una función y un método tienen comportamientos similares, puede utilizar cualquiera de los dos para controlar los clips de película. La elección depende de lo que prefiera y de lo familiarizado que esté con la creación de scripts en ActionScript. Tanto si utiliza una función como un método, la línea de tiempo de destino debe cargarse en Flash Player al llamar a la función o al método.

Para utilizar un método, debe activarlo indicando la ruta de destino del nombre de instancia seguida de un punto (.) y, después, del nombre del método y los parámetros, como se muestra en las sentencias siguientes:

```
myMovieClip.play();
parentClip.childClip.gotoAndPlay(3);
```

En la primera sentencia, `play()` mueve la cabeza lectora de la instancia `myMovieClip`. En la segunda sentencia, `gotoAndPlay` envía la cabeza lectora de `childClip` (que depende de la instancia `parentClip`) al fotograma 3 y continúa moviendo la cabeza lectora.

Las acciones globales que controlan una línea de tiempo tienen un parámetro de *destino* que permite especificar la ruta de destino a la instancia que se desea controlar. Por ejemplo, en el script siguiente `startDrag()` utiliza la instancia en la que se coloca el código y hace que se pueda arrastrar:

```
my_mc.onPress = function() {
  startDrag(this);
};
my_mc.onRelease = function() {
  stopDrag();
};
```

Las funciones siguientes utilizan clips de película: `loadMovie()`, `unloadMovie()`, `loadVariables()`, `setProperty()`, `startDrag()`, `duplicateMovieClip()` y `removeMovieClip()`. Para utilizar estas funciones, es necesario introducir una ruta de destino para el parámetro *target* de la función, a fin de indicar el destino de la misma.

Los métodos MovieClip siguientes pueden controlar clips de película o niveles cargados y no tienen funciones equivalentes. `MovieClip.attachMovie()`, `MovieClip.createEmptyMovieClip()`, `MovieClip.createTextField()`, `MovieClip.getBounds()`, `MovieClip.getBytesLoaded()`, `MovieClip.getBytesTotal()`, `MovieClip.getDepth()`, `MovieClip.getInstanceAtDepth()`, `MovieClip.getNextHighestDepth()`, `MovieClip.globalToLocal()`, `MovieClip.localToGlobal()`, `MovieClip.hitTest()`, `MovieClip.setMask()`, `MovieClip.swapDepths()`.

Para más información sobre estas funciones y métodos, consulte sus correspondientes entradas en *Referencia del lenguaje ActionScript 2.0*.

Para ver un ejemplo de animación mediante scripts en Flash, encontrará un archivo de origen de ejemplo, animation.fla, en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Animation.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Animation.

Puede encontrar muestras de aplicaciones de galerías de fotos en el disco duro. Estos archivos proporcionan ejemplos de cómo utilizar ActionScript para controlar dinámicamente clips de película mientras se cargan archivos de imagen en un archivo SWF que incluye animación mediante scripts. Los archivos de origen de muestra gallery_tree.fla y gallery_tween.fla se pueden encontrar en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Galleries.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Galleries.

# Llamada a varios métodos en un solo clip de película

Puede utilizar la sentencia `with` para referirse a un clip de película una vez y después ejecutar una serie de métodos en ese clip. La sentencia `with` funciona en todos los objetos de ActionScript (por ejemplo, Array, Color y Sound), no solamente en clips de película.

La sentencia `with` toma un clip de película como parámetro. El objeto que especifique se añadirá al final de la ruta de destino actual. Todas las acciones anidadas dentro de una sentencia `with` se llevan a cabo dentro de la nueva ruta de destino o ámbito. Por ejemplo, en el siguiente script, el objeto `donut.hole` pasa a la sentencia `with` para cambiar las propiedades de `hole`:

```
with (donut.hole) {
  _alpha = 20;
  _xscale = 150;
  _yscale = 150;
}
```

El script se comporta como si a las sentencias que hay en la sentencia `with` se las llamara desde la línea de tiempo de la instancia `hole`. El código anterior equivale al siguiente ejemplo:

```
donut.hole._alpha = 20;
donut.hole._xscale = 150;
donut.hole._yscale = 150;
```

El código anterior también equivale al siguiente ejemplo:

```
with (donut) {
  hole._alpha = 20;
  hole._xscale = 150;
  hole._yscale = 150;
}
```

# Carga y descarga de archivos SWF

Para reproducir películas SWF adicionales sin cerrar Flash Player o pasar de un archivo SWF a otro sin cargar otra página HTML, puede utilizar una de las siguientes opciones:

- La función global `loadMovie()` o el método `loadMovie()` de la clase MovieClip.
- El método `loadClip()` de la clase MovieClipLoader. Para más información sobre la clase MovieClipLoader, consulte `%{MovieClipLoader}%` en *Referencia del lenguaje ActionScript 2.0*.

Puede utilizar el método `loadMovie()` para enviar variables a un script CGI que genera un archivo SWF como resultado del CGI. Por ejemplo, puede utilizar este procedimiento para cargar archivos SWF o de imagen dinámicos basados en variables específicas dentro de un clip de película. Cuando carga un archivo SWF, puede especificar el nivel o clip de película en el que se carga el archivo SWF. Si carga un archivo SWF en un destino, el archivo SWF cargado heredará las propiedades del clip de película de destino. Una vez que se ha cargado la película de Flash, puede cambiar dichas propiedades.

El método `unloadMovie()` elimina un archivo SWF cargado previamente con el método `loadMovie()`. La descarga de modo explícito de archivos SWF con `unloadMovie()` garantiza una transición gradual entre los archivos SWF y puede reducir la memoria que requiere Flash Player. Puede resultar más eficiente en algunas situaciones establecer la propiedad `_visible` del clip de película con el valor `false` en lugar descargar el clip. En el caso de que reutilice el clip posteriormente, establezca la propiedad `_visible` con el valor `false` y luego establézcala con el valor `true` cuando sea necesario.

Utilice `loadMovie()` para seguir uno de estos procedimientos:

- Reproducir una secuencia de anuncios publicitarios en forma de archivos SWF colocando una función `loadMovie()` en un archivo SWF contenedor que carga y descarga secuencialmente los archivos publicitarios SWF.
- Desarrollar una interfaz de bifurcación con vínculos que permita al usuario elegir entre distintos archivos SWF utilizados para mostrar el contenido de un sitio.
- Crear una interfaz de navegación con controles de navegación en el nivel 0 que carguen contenido en otros niveles. La carga de contenido en niveles contribuye a producir transiciones más suaves entre páginas de contenido que la carga de nuevas páginas HTML en un navegador.

Para más información sobre la carga de archivos SWC, consulte "Carga de archivos de imagen y SWF externos" en la página 629.

Para más información, consulte los siguientes temas:

- "Especificación de una línea de tiempo raíz para archivos SWF cargados" en la página 378
- "Carga de archivos de imagen en clips de película" en la página 379

# Especificación de una línea de tiempo raíz para archivos SWF cargados

La propiedad de ActionScript `_root` especifica o contiene una referencia a la línea de tiempo raíz de un archivo SWF. Si un archivo SWF tiene varios niveles, la línea de tiempo raíz está en el nivel que contiene el script que se está ejecutando. Por ejemplo, si un script del nivel 1 consulta el valor de `_root`, se devuelve `_level1`. No obstante, la línea de tiempo que `_root` especifica puede cambiar en función de si un archivo SWF se ejecuta de forma independiente (en su propio nivel) o de si se carga en una instancia de clip de película mediante una llamada `loadMovie()`.

En el siguiente ejemplo, piense en un archivo denominado container.swf que tenga una instancia de clip de película denominada `target_mc` en su línea de tiempo principal. El archivo container.swf declara una variable denominada `userName` en su línea de tiempo principal; a continuación, el mismo script carga otro archivo denominado contents.swf en el clip de película `target_mc`.

```
// En container.swf:
_root.userName = "Tim";
target_mc.loadMovie("contents.swf");
my_btn.onRelease = function():Void {
  trace(_root.userName);
};
```

En el siguiente ejemplo, el archivo SWF cargado, contents.swf, también declara una variable denominada `userName` en su línea de tiempo raíz:

```
// En contents.swf:
_root.userName = "Mary";
```

Tras cargarse contents.swf en el clip de película de container.swf, el valor de `userName` asociado a la línea de tiempo raíz del archivo SWF que lo aloja (container.swf) se establecería en `"Mary"` en lugar de `"Tim"`. Esto puede ocasionar que el código de container.swf (así como contents.swf) no funcione correctamente.

Para hacer que `_root` indique siempre el valor de línea de tiempo del archivo SWF cargado, en lugar de la línea de tiempo raíz, utilice la propiedad `_lockroot`. Puede establecer esta propiedad mediante la carga del archivo SWF o el archivo SWF que se está cargando. Cuando `_lockroot` se establece en `true` en una instancia de clip de película, dicho clip de película actúa como `_root` para cualquier SWF que esté cargado en él. Cuando `_lockroot` se establece en `true` en un archivo SWF, dicho archivo SWF actúa como su propia raíz al margen de qué otro archivo SWF la cargue. Puede establecerse `_lockroot` en `true` en cualquier clip de película y en tantos como se desee. De forma predeterminada, esta propiedad es `false`.

Por ejemplo, el autor de container.swf podría asociar el código siguiente del fotograma 1 de la línea de tiempo principal:

```
// Añadido al fotograma 1 de container.swf:
target_mc._lockroot = true;
```

Este paso garantiza que las referencias a `_root` existentes en contents.swf (o en cualquier archivo SWF cargado en `target_mc`) hagan referencia a su propia línea de tiempo, no a la línea de tiempo raíz de container.swf. Ahora, al hacer clic en el botón, aparece "`Tim`".

Como alternativa, el autor de contents.swf podría añadir el código siguiente a su línea de tiempo principal:

```
// Añadido al fotograma 1 de contents.swf:
this._lockroot = true;
```

Esto garantizaría que, al margen de donde estuviera cargada contents.swf, las referencias que hiciera a `_root` se aplicarían a su propia línea de tiempo principal, no a la línea de tiempo del archivo SWF que aloja.

Para más información, consulte %{_lockroot (propiedad MovieClip._lockroot)}%.

## Carga de archivos de imagen en clips de película

Puede utilizar la función `loadMovie()`, o el método MovieClip del mismo nombre, para cargar archivos de imagen en una instancia de clip de película. También puede utilizar la función `loadMovieNum()` para cargar un archivo de imagen en un nivel.

Al cargar una imagen en un clip de película, la esquina superior izquierda de la imagen se coloca en el punto de registro del clip de película. Dado que este punto suele ser el centro del clip de película, la imagen cargada no aparecerá en el centro. Además, cuando se carga una imagen en una línea de tiempo raíz, la esquina superior izquierda de la imagen se sitúa en la esquina superior izquierda del escenario. La imagen cargada hereda la rotación y la escala del clip de película, pero el contenido original del clip de película se elimina.

Para más información, consulte %{`loadMovie function`}%, %{loadMovie (`método MovieClip.loadMovie`)}% y %{`loadMovieNum function`}% en *Referencia del lenguaje ActionScript 2.0* y "Carga de archivos de imagen y SWF externos" en la página 629.

# Modificación de la posición y el aspecto de un clip de película

Para cambiar las propiedades de un clip de película a medida que se reproduce, escriba una sentencia que asigne un valor a una propiedad o utilice la función `setProperty()`. Por ejemplo, en el código siguiente se establece la rotación de la instancia `mc` en 45:

```
my_mc._rotation = 45;
```

Esto es equivalente al código siguiente, en el que se utiliza la función `setProperty()`:

```
setProperty("my_mc", _rotation, 45);
```

Algunas propiedades, llamadas *propiedades de sólo lectura*, tienen valores que se pueden leer pero no establecer. (Estas propiedades se especifican como de sólo lectura en las correspondientes entradas de *Referencia del lenguaje ActionScript 2.0*.) A continuación, se muestran propiedades de sólo lectura: `_currentframe`, `_droptarget`, `_framesloaded`, `_parent`, `_target`, `_totalframes`, `_url`, `_xmouse` e `_ymouse`.

Puede escribir sentencias para establecer cualquier propiedad que no sea de sólo lectura. La sentencia siguiente establece la propiedad `_alpha` de la instancia de clip de película `wheel_mc`, que es secundaria de la instancia `car_mc`:

```
car_mc.wheel_mc._alpha = 50;
```

Además, puede escribir sentencias que obtengan el valor de una propiedad de clip de película. Por ejemplo, la sentencia siguiente obtiene el valor de la propiedad `_xmouse` de la línea de tiempo del nivel actual y establece la propiedad `_x` de la instancia `my_mc` en ese valor:

```
this.onEnterFrame = function() {
  my_mc._x = _root._xmouse;
};
```

Esto es equivalente al código siguiente, en el que se utiliza la función `getProperty()`:

```
this.onEnterFrame = function() {
  my_mc._x = getProperty(_root, _xmouse);
};
```

Las propiedades `_x`, `_y`, `_rotation`, `_xscale`, `_yscale`, `_height`, `_width`, `_alpha` y `_visible` se ven afectadas por las transformaciones del elemento principal del clip de película y modifican el clip de película y cualquiera de los elementos secundarios del clip. Las propiedades `_focusrect`, `_highquality`, `_quality` y `_soundbuftime` son globales, solamente pertenecen al nivel 0 de la línea de tiempo principal. Todas las demás propiedades pertenecen a cada clip de película o nivel cargado.

Para ver una lista de propiedades de clip de película, consulte el resumen de propiedades de la clase %{MovieClip}% en *Referencia del lenguaje ActionScript 2.0*.

Para ver un ejemplo de animación mediante scripts en Flash, encontrará un archivo de origen de ejemplo, animation.fla, en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Animation.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Animation.

Puede encontrar muestras de aplicaciones de galerías de fotos en el disco duro. Estos archivos proporcionan ejemplos de cómo utilizar ActionScript para controlar dinámicamente clips de película mientras se cargan archivos de imagen en un archivo SWF que incluye animación mediante scripts. Los archivos de origen de muestra gallery_tree.fla y gallery_tween.fla se pueden encontrar en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Galleries.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Galleries.

# Clips de película que se pueden arrastrar

Puede utilizar la función `startDrag()` global o el método `MovieClip.startDrag()` para que sea posible arrastrar un clip de película. Por ejemplo, puede crear un clip de película que se pueda arrastrar para juegos, funciones de arrastrar y soltar, interfaces personalizables, barras de desplazamiento y controles deslizantes.

Un clip de película se puede arrastrar hasta que esta acción se detenga explícitamente con `stopDrag()` o hasta que se utilice otro clip de película con `startDrag`. Sólo es posible mover clips de película de uno en uno en un archivo SWF.

Para crear comportamientos de arrastrar y soltar más complejos, puede probar la propiedad `_droptarget` del clip de película que se está arrastrando. Por ejemplo, puede examinar la propiedad `_droptarget` para ver si el clip de película se arrastró a un clip de película específico (como un clip de película de tipo "papelera") y, a continuación, desencadenar otra acción, como se muestra en el siguiente ejemplo:

```
// Arrastrar un residuo.
garbage_mc.onPress = function() {
  this.startDrag(false);
};
// Cuando el residuo se arrastra sobre la papelera, hacerlo invisible.
garbage_mc.onRelease = function() {
  this.stopDrag();
  // Convertir la notación con barras en notación con puntos utilizando
  eval.
  if (eval(this._droptarget) == trashcan_mc) {
    garbage_mc._visible = false;
  }
};
```

Para más información, consulte %{`startDrag function`}% o %{startDrag (`método MovieClip.startDrag`)}% en *Referencia del lenguaje ActionScript 2.0*.

Puede encontrar una muestra de aplicación de galería de fotos en el disco duro. Este archivo proporciona un ejemplo de cómo utilizar ActionScript para controlar dinámicamente clips de película mientras se cargan archivos de imagen en un archivo SWF, que añade la función de arrastre a cada clip de película. El archivo de origen de muestra, gallery_tween.fla, se puede encontrar en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Galleries.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Galleries.

# Creación de clips de película en tiempo de ejecución

Además de crear instancias de clips de película en el entorno de creación de Flash, también se pueden crear en tiempo de ejecución de las siguientes formas:

- "Creación de un clip de película vacío" en la página 384
- "Duplicación o eliminación de un clip de película" en la página 385
- "Asociación de un símbolo de clip de película al escenario" en la página 386

Cada instancia de clip de película que cree en tiempo de ejecución deberá tener un nombre de instancia y un valor de profundidad (apilamiento u orden *z*). La profundidad especificada determina la manera en que el nuevo clip se solapa con otros clips en la misma línea de tiempo. También permite sobrescribir los clips de película que residen a la misma profundidad. (Consulte "Gestión de las profundidades de los clips de película" en la página 390.)

Puede encontrar una muestra de aplicación de galería de fotos en el disco duro. Este archivo proporciona un ejemplo de cómo utilizar ActionScript para controlar dinámicamente clips de película mientras se cargan archivos de imagen en un archivo SWF, que añade la función de crear clips de película en tiempo de ejecución. El archivo de origen de muestra, gallery_tween.fla, se puede encontrar en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Galleries.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Galleries.

Para ver un ejemplo de archivo de origen que crea y elimina varios clips de película en tiempo de ejecución, puede encontrar un archivo de origen de muestra, animation.fla, en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Animation.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Animation.

Para más información, consulte los siguientes temas:

- "Creación de un clip de película vacío" en la página 384
- "Duplicación o eliminación de un clip de película" en la página 385
- "Asociación de un símbolo de clip de película al escenario" en la página 386

## Creación de un clip de película vacío

Para crear un clip de película nuevo vacío en el escenario, utilice el método `createEmptyMovieClip()` de la clase MovieClip. Este método crea un clip de película como elemento secundario del clip que llama al método. El punto de registro de un clip de película vacío recién creado se encuentra en la esquina superior izquierda.

Por ejemplo, en el código siguiente se crea un nuevo clip de película secundario, denominado `new_mc`, a una profundidad de 10 pulgadas en el clip de película denominado `parent_mc`:

```
parent_mc.createEmptyMovieClip("new_mc", 10);
```

Con el código siguiente se crea un nuevo clip de película, denominado `canvas_mc`, en la línea de tiempo raíz del archivo SWF en el que se ejecuta el script y, a continuación, se activa la acción `loadMovie()` para cargar en ella un archivo JPEG externo:

```
this.createEmptyMovieClip("canvas_mc", 10);
canvas_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
```

Como se muestra en el siguiente ejemplo, puede cargar la imagen image2.jpg en un clip de película y utilizar el método `MovieClip.onPress()` para que la imagen actúe como un botón. La carga de una imagen mediante `loadMovie()` reemplaza el clip de película por la imagen, pero no le proporciona acceso a los métodos de clip de película. Para obtener acceso a métodos de clip de película, debe crear un clip de película principal vacío y un clip de película secundario como contenedor. Cargue la imagen en el contenedor y coloque el controlador de eventos en el clip de película principal.

```
// Crea un clip de película principal que incluye al contenedor.
this.createEmptyMovieClip("my_mc", 0);

// Crea un clip de película secundario dentro de "my_mc".
// Este es el clip de película al que reemplazará la imagen.
my_mc.createEmptyMovieClip("container_mc",99);

// Utilizar MovieClipLoader para cargar la imagen.
var my_mcl:MovieClipLoader = new MovieClipLoader();
my_mcl.loadClip("http://www.helpexamples.com/flash/images/image2.jpg",
  my_mc.container_mc);

// Colocar el controlador de eventos en el clip de película principal my_mc.
my_mc.onPress = function():Void {
  trace("It works");
};
```

Para más información, consulte %{createEmptyMovieClip (`método MovieClip.createEmptyMovieClip)}%` en *Referencia del lenguaje ActionScript 2.0*.

Para ver un ejemplo de archivo de origen que crea y elimina varios clips de película en tiempo de ejecución, puede encontrar un archivo de origen de muestra, animation.fla, en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Animation.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Animation.

## Duplicación o eliminación de un clip de película

Para duplicar o eliminar instancias de clip de película, utilice las funciones globales `duplicateMovieClip()` o `removeMovieClip()`, o bien los métodos de la clase MovieClip que llevan el mismo nombre. Con el método `duplicateMovieClip()` se crea una nueva instancia de una instancia de clip de película existente, se le asigna un nuevo nombre de instancia y se le da una profundidad o un orden *z*. Un clip de película duplicado siempre comienza en el fotograma 1, aunque el clip de película original se encontrase en otro fotograma cuando se duplicó, y siempre se encuentra delante de todos los clips de película definidos anteriormente situados en la línea de tiempo.

Para borrar un clip de película creado con `duplicateMovieClip()`, utilice `removeMovieClip()`. Los clips de película duplicados también se eliminan si se borra el clip de película principal.

Para más información, consulte %{`duplicateMovieClip function`}% y %{`removeMovieClip function`}% en *Referencia del lenguaje ActionScript 2.0*.

Para ver un ejemplo de archivo de origen que crea y elimina varios clips de película en tiempo de ejecución, puede encontrar un archivo de origen de muestra, animation.fla, en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Animation.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Animation.

# Asociación de un símbolo de clip de película al escenario

Una nueva manera de crear instancias de clip de película en tiempo de ejecución consiste en utilizar el método `attachMovie()`. El método `attachMovie()` asocia al escenario una instancia de un símbolo de clip de película de la biblioteca del archivo SWF. El nuevo clip pasa a ser un clip secundario del clip que lo ha asociado.

Para utilizar ActionScript a fin de asociar un símbolo de clip de película de la biblioteca, es necesario exportar el símbolo para ActionScript y asignarle un identificador de vínculo exclusivo. Para ello, deberá utilizar el cuadro de diálogo Propiedades de vinculación.

De forma predeterminada, todos los clips de película que se exportan para usarlos con ActionScript se cargan antes del primer fotograma del archivo SWF que los contiene. Esto puede producir cierta demora en la reproducción del primer fotograma. Cuando asigne un identificador de vínculo a un elemento, también puede especificar si este contenido debe añadirse antes del primer fotograma. Si no se añade en el primer fotograma, debe incluir una instancia de éste en algún otro fotograma del archivo SWF; de lo contrario, el elemento no se exporta al archivo SWF.

### Para asignar un identificador de vínculo a un clip de película:

1. Seleccione Ventana > Biblioteca, para abrir el panel Biblioteca.
2. Seleccione un clip de película del panel Biblioteca.
3. En el panel Biblioteca, elija Vinculación en el menú emergente del panel Biblioteca.

   Aparecerá el cuadro de diálogo Propiedades de vinculación.
4. En Vinculación, seleccione Exportar para ActionScript.
5. En Identificador, introduzca un ID para el clip de película.

   De forma predeterminada, el identificador y el símbolo tienen el mismo nombre.

   También puede asignar una clase de ActionScript al símbolo del clip de película Esto permite al clip de película heredar los métodos y propiedades de una clase específica. (Consulte "Asignación de una clase a un símbolo de clip de película" en la página 404.)
6. Si no desea que el clip de película se cargue antes que el primer fotograma, deseleccione la opción Exportar en primer fotograma.

   Si anula la selección de esta opción, coloque una instancia del clip de película en el fotograma de la línea de tiempo donde desee que esté disponible. Por ejemplo, si el script que está escribiendo no hace referencia al clip de película hasta el fotograma 10, coloque una instancia del símbolo en el fotograma 10 o antes del mismo en la línea de tiempo.
7. Haga clic en Aceptar.

Cuando haya asignado un identificador de vínculo a un clip de película, podrá asociar una instancia del símbolo al escenario en tiempo de ejecución utilizando `attachMovie()`.

**Para adjuntar un clip de película a otro clip de película:**

1. Asigne un identificador de vínculo a un símbolo de biblioteca de clips de película, tal como se ha descrito en el ejemplo anterior.
2. Con el panel Acciones abierto (Ventana > Acciones), seleccione un fotograma en la línea de tiempo.
3. En el panel Script del panel Acciones, escriba el nombre del clip de película o el nivel al que desea asociar el nuevo clip de película.

   Por ejemplo, para asociar el clip de película a la línea de tiempo raíz, escriba **this**.
4. En la caja de herramientas Acciones (a la izquierda del panel Acciones), seleccione Clases de ActionScript 2.0 > Película > MovieClip > Métodos y elija `attachMovie()`.
5. Utilizando las sugerencias para el código que aparecen como guía, introduzca valores para los parámetros siguientes:
   - En `idName`, especifique el identificador introducido en el cuadro de diálogo Propiedades de vinculación.
   - En `newName`, introduzca un nombre de instancia para el clip asociado para poder utilizarlo.
   - En `depth`, introduzca el nivel en el que el clip de película duplicado se asociará al clip de película. Cada clip de película asociado tiene su propio orden de apilamiento, siendo el nivel 0 el nivel del clip de película que lo originó. Los clips de película asociados siempre se sitúan sobre el clip de película original, como se muestra en el siguiente ejemplo:

     ```
     this.attachMovie("calif_id", "california_mc", 10);
     ```

Para más información, consulte %{attachMovie (`método MovieClip.attachMovie)}%` en *Referencia del lenguaje ActionScript 2.0*.

# Adición de parámetros a clips de película creados de forma dinámica

Al utilizar `MovieClip.attachMovie()` y `MovieClip.duplicateMovie()` para crear o duplicar un clip de película dinámicamente, puede rellenar el clip de película con parámetros de otro objeto. El parámetro *`initObject`* de `attachMovie()` y `duplicateMovie()` permite que los clips de película creados de forma dinámica reciban parámetros de clip.

Para más información, consulte `%{`attachMovie (método MovieClip.attachMovie)}% y `%{duplicateMovieClip (método MovieClip.duplicateMovieClip)}%` en *Referencia del lenguaje ActionScript 2.0*.

### Para rellenar un clip de película creado dinámicamente con parámetros de un objeto específico:

Realice uno de los siguientes pasos:

- Utilice la sintaxis siguiente con `attachMovie()`:
  `myMovieClip.attachMovie(idName, newName, depth [, initObject]);`
- Utilice la sintaxis siguiente con `duplicateMovie()`:
  `myMovieClip.duplicateMovie(idName, newName, depth [, initObject]);`

Con el parámetro *initObject* se especifica el nombre del objeto cuyos parámetros se van a utilizar para rellenar el clip de película creado de forma dinámica.

### Para rellenar un clip de película con parámetros mediante attachMovie():

1. En un nuevo documento de Flash, cree un nuevo símbolo de clip de película seleccionando Insertar > Nuevo símbolo.
2. Escriba **dynamic_mc** en el cuadro de texto Nombre y seleccione el comportamiento del clip de película.
3. Dentro del símbolo, cree un campo de texto dinámico en el escenario, con el nombre de instancia **name_txt**.

   Asegúrese de que este campo de texto se encuentra más abajo y a la derecha del punto de registro.
4. Seleccione el fotograma 1 de la línea de tiempo del clip de película y abra el panel Acciones (Ventana > Acciones).
5. Cree una nueva variable llamada `name_str` y asigne su valor a la propiedad `text` de `name_txt`, tal como se muestra en el siguiente ejemplo:

   ```
   var name_str:String;
   name_txt.text = name_str;
   ```
6. Seleccione Edición > Editar documento para volver a la línea de tiempo principal.
7. Seleccione el símbolo de clip de película de la biblioteca y, a continuación, seleccione Vinculación en el menú emergente Biblioteca.

   Aparecerá el cuadro de diálogo Propiedades de vinculación.
8. Seleccione Exportar para ActionScript y Exportar en primer fotograma.
9. Escriba **dynamic_id** en el cuadro de texto Identificador y haga clic en Aceptar.

10. Seleccione el primer fotograma de la línea de tiempo principal y añada el código siguiente al panel Script del panel Acciones:

```
/* Asocia un nuevo clip de película y lo mueve a la coordenada x e y 50
  */

this.attachMovie("dynamic_id", "newClip_mc", 99, {name_str:"Erick",
  _x:50, _y:50});
```

11. Pruebe el documento de Flash (Control > Probar película).

    El nombre especificado en la llamada a `attachMovie()` aparece dentro del campo de texto del nuevo clip de película.

Puede encontrar una muestra de aplicación de galería de fotos en el disco duro. Este archivo proporciona un ejemplo de cómo utilizar ActionScript para controlar dinámicamente clips de película mientras se cargan archivos de imagen en un archivo SWF, que añade la función de crear clips de película en tiempo de ejecución. El archivo de origen de muestra, gallery_tween.fla, se puede encontrar en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Galleries.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Galleries.

Para ver un ejemplo de archivo de origen que crea y elimina varios clips de película en tiempo de ejecución, puede encontrar un archivo de origen de muestra, animation.fla, en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Animation.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Animation.

# Gestión de las profundidades de los clips de película

Cada clip de película tiene su propio espacio de orden *z* que determina la forma en la que los objetos se solapan en el archivo SWF o clip de película principal. Cada clip de película tiene asociado un valor de profundidad que determina si dicho clip se presenta delante o detrás de otros clips de película dentro de la misma línea de tiempo del clip de película. Cuando se crea un clip de película en tiempo de ejecución mediante el %{attachMovie (método MovieClip.attachMovie)}%, %{duplicateMovieClip (método MovieClip.duplicateMovieClip)}% o %{createEmptyMovieClip (método MovieClip.createEmptyMovieClip)}%, siempre se especifica una profundidad para el nuevo clip como parámetro del método. Por ejemplo, el código siguiente asocia un nuevo clip de película a la línea de tiempo de un clip de película denominado `container_mc` con el valor de profundidad 10.

```
container_mc.attachMovie("symbolID", "clip1_mc", 10);
```

Este ejemplo crea un nuevo clip de película con la profundidad 10 en el espacio de orden *z* de `container_mc`.

El código siguiente asocia dos nuevos clips de película a `container_mc`. El primer clip, llamado `clip1_mc`, se representa detrás de `clip2_mc`, puesto que se le asignó un valor inferior de profundidad.

```
container_mc.attachMovie("symbolID", "clip1_mc", 10);
container_mc.attachMovie("symbolID", "clip2_mc", 15);
```

Los valores de profundidad para los clips de película pueden oscilar entre -16384 y 1048575. Si crea o asocia un nuevo clip de película en una profundidad que ya tiene un clip de película, el clip nuevo o asociado sobrescribirá el contenido del existente. Para evitar este problema, emplee el método `MovieClip.getNextHighestDepth()`; sin embargo, no debe emplear este método con componentes que utilicen un sistema de gestión de profundidad distinto. En su lugar, utilice "DepthManager, clase" con instancias de componentes.

La clase MovieClip ofrece varios métodos para gestionar profundidades de clips de película; para más información, consulte %{getNextHighestDepth (método MovieClip.getNextHighestDepth)}%, %{getInstanceAtDepth (método MovieClip.getInstanceAtDepth)}%, %{getDepth (método MovieClip.getDepth)}% y %{swapDepths (método MovieClip.swapDepths)}% en *Referencia del lenguaje ActionScript 2.0*.

Para más información sobre profundidades de clips de película, consulte los siguientes temas:


- "Determinación del siguiente valor superior de profundidad disponible" en la página 391
- "Determinación de la instancia a una profundidad determinada" en la página 392
- "Determinación de la profundidad de una instancia" en la página 392
- "Intercambio de profundidades de clip de película" en la página 392


## Determinación del siguiente valor superior de profundidad disponible

Para determinar el siguiente valor superior de profundidad disponible de un clip de película, consulte `MovieClip.getNextHighestDepth()`. El valor entero devuelto por este método indica la próxima profundidad disponible que se representará delante del resto de los objetos del clip de película.

Con el código siguiente se asocia un nuevo clip de película, con un valor de profundidad 10, en la línea de tiempo denominada `file_mc`. A continuación, determina la siguiente profundidad superior disponible de ese mismo clip de película y crea un nuevo clip de película denominado `edit_mc` en esa profundidad.

```
this.attachMovie("menuClip","file_mc", 10, {_x:0, _y:0});
trace(file_mc.getDepth()); // 10
var nextDepth:Number = this.getNextHighestDepth();
this.attachMovie("menuClip", "edit_mc", nextDepth, {_x:200, _y:0});
trace(edit_mc.getDepth()); // 11
```

En este caso, la variable denominada `nextDepth` contiene el valor 11, puesto que es el siguiente valor de profundidad superior disponible para el clip de película `edit_mc`.

No utilice `MovieClip.getNextHighestDepth()` con componentes; en su lugar, emplee el gestor de profundidad. Para más información, consulte "DepthManager, clase" en *Referencia del lenguaje de componentes*. Para más información sobre `MovieClip.getNextHighestDepth()`, consulte `%{getNextHighestDepth (método MovieClip.getNextHighestDepth)}%`.

Para obtener el valor actual más alto de profundidad ocupado, reste 1 al valor devuelto por `getNextHighestDepth()`, tal como se muestra en la sección siguiente.

## Determinación de la instancia a una profundidad determinada

Para determinar la instancia a una profundidad determinada, utilice `MovieClip.getInstanceAtDepth()`. Este método devuelve una referencia a la instancia MovieClip a la profundidad especificada.

El código siguiente combina `getNextHighestDepth()` y `getInstanceAtDepth()` para determinar el clip de película en el valor de profundidad (actual) más alto ocupado de la línea de tiempo raíz.

```
var highestOccupiedDepth:Number = this.getNextHighestDepth() - 1;
var instanceAtHighestDepth:MovieClip =
  this.getInstanceAtDepth(highestOccupiedDepth);
```

Para más información, consulte %{`getInstanceAtDepth (método MovieClip.getInstanceAtDepth)`}% en *Referencia del lenguaje ActionScript 2.0*.

## Determinación de la profundidad de una instancia

Para determinar la profundidad de una instancia de clip de película, utilice `MovieClip.getDepth()`.

El código siguiente se repite en todos los clips de película de la línea de tiempo principal de un archivo SWF y muestra el nombre de instancia y el valor de profundidad correspondiente de cada clip en el panel Salida:

```
for (var item:String in _root) {
  var obj:Object = _root[item];
  if (obj instanceof MovieClip) {
    var objDepth:Number = obj.getDepth();
    trace(obj._name + ":" + objDepth)
  }
}
```

Para más información, consulte %{`getDepth (método MovieClip.getDepth)`}% en *Referencia del lenguaje ActionScript 2.0*.

## Intercambio de profundidades de clip de película

Para intercambiar las profundidades de dos clips de película en la misma línea de tiempo, utilice `MovieClip.swapDepths()`. Los ejemplos siguientes muestran cómo dos instancias de clip de película pueden intercambiar profundidades en tiempo de ejecución.

### Para intercambiar profundidades de clip de película:

1. Cree un nuevo documento de Flash denominado **swap.fla**.
2. Dibuje un círculo azul en el escenario.
3. Seleccione el círculo azul y, a continuación, elija Modificar > Convertir en símbolo.
4. Seleccione la opción Clip de película y haga clic en Aceptar.
5. Seleccione la instancia en el escenario y escriba **first_mc** en el cuadro de texto Nombre de instancia del inspector de propiedades.
6. Dibuje un círculo rojo en el escenario y luego elija Modificar > Convertir en símbolo.
7. Seleccione la opción Clip de película y haga clic en Aceptar.
8. Seleccione la instancia en el escenario y escriba **second_mc** en el cuadro de texto Nombre de instancia del inspector de propiedades.
9. Arrastre las dos instancias para que se solapen ligeramente en el escenario.
10. Seleccione el fotograma 1 de la línea de tiempo y, en el panel Acciones, introduzca el siguiente código:

```
first_mc.onRelease = function() {
    this.swapDepths(second_mc);
};
second_mc.onRelease = function() {
  this.swapDepths(first_mc);
};
```

11. Seleccione Control > Probar película para probar el documento.

    Al hacer clic en las instancias del escenario, éstas intercambian profundidades. Verá las dos instancias cambiar, el clip de la parte superior por el otro.

Para más información, consulte %{swapDepths (método MovieClip.swapDepths)}% en *Referencia del lenguaje ActionScript 2.0*.

# Asignación de caché y desplazamiento de clips de película con ActionScript

El tamaño de sus diseños en Flash irá creciendo, tanto si está creando una aplicación como si realiza complejas animaciones mediante scripts, por lo que deberá tener en cuenta el rendimiento y la optimización. Si tiene contenido que permanece estático (como un clip de película rectangular), Flash no optimizará el contenido. Por consiguiente, al cambiar la posición del clip de película rectangular, Flash vuelve a dibujar todo el rectángulo en Flash Player 7 y las versiones anteriores.

En Flash Player 8, puede asignar caché a botones y clips de película especificados para mejorar el rendimiento del archivo SWF. El botón o clip de película es una *superficie*, básicamente una versión de mapa de bits de los datos vectoriales de la instancia, que son datos que no deseará que cambien mucho a lo largo del archivo SWF. Por consiguiente, las instancias para las que está activada la caché no se redibujan continuamente mientras se reproduce el archivo SWF, lo que permite que el archivo SWF se represente rápidamente.

| NOTA | Puede actualizar los datos vectoriales, momento en el cual se recrea la superficie. Por consiguiente, los datos vectoriales para los que se asigna la caché en la superficie no tienen porqué permanecer intactos durante todo el archivo SWF. |
|---|---|

Puede utilizar código ActionScript para activar la caché o el desplazamiento, así como para controlar fondos. Puede utilizar el inspector de propiedades para activar la caché de una instancia de clip de película. Para guardar en caché clips de película o botones sin utilizar ActionScript, puede seleccionar la opción Utilizar caché de mapa de bits en el inspector de propiedades.

La siguiente tabla contiene descripciones breves de las nuevas propiedades para las instancias de clip de película:

| Propiedad | Descripción |
|---|---|
| cacheAsBitmap | Hace que la instancia de clip de película almacene en caché una representación de mapa de bits de sí misma. Flash crea un objeto de superficie para la instancia, que es un mapa de bits almacenado en caché en lugar de los datos vectoriales. Si cambia los límites del clip de película, la superficie se recrea en lugar de modificarse su tamaño. Para más información y ver un ejemplo, consulte "Asignación de caché para un clip de película" en la página 398. |
| opaqueBackground | Permite especificar un color de fondo para la instancia de clip de película opaca. Si establece esta propiedad con un valor numérico, la instancia de clip de película tendrá una superficie opaca (no transparente). Un mapa de bits opaco no tiene canal alfa (transparencia) y se muestra más rápido. Para más información y ver un ejemplo, consulte "Definición del fondo de un clip de película" en la página 401. |

| Propiedad | Descripción |
|---|---|
| scrollRect | Permite desplazar con rapidez el contenido del clip de película y abrir una ventana que muestre mayor cantidad de contenido. El contenido del clip de película se corta y la instancia se desplaza con una anchura, altura y espacio de desplazamiento especificados. Esto permite al usuario desplazar rápidamente el contenido del clip de película y abrir una ventana que muestre mayor cantidad de contenido de lo que permite el área del escenario. Los campos de texto y el contenido complejo que se muestran en la instancia pueden desplazarse más rápido, ya que Flash no regenera todos los datos vectoriales del clip de película. Para más información y ver un ejemplo, consulte `%{scrollRect (propiedad MovieClip.scrollRect)}%`. |

Estas tres propiedades son independientes entre sí, sin embargo, las propiedades `opaqueBackground` y `scrollRect` funcionan mejor cuando un objeto está almacenado en caché como mapa de bits. Sólo se obtienen ventajas de rendimiento de las propiedades `opaqueBackground` y `scrollRect` cuando se establece `cacheAsBitmap` como `true`.

Para crear una superficie que también sea desplazable, debe establecer las propiedades `cacheAsBitmap` y `scrollRect` para la instancia del clip de película. Las superficies pueden anidarse dentro de otras superficies. La superficie copia el mapa de bits en su superficie principal.

Para obtener información sobre el enmascaramiento del canal alfa, que requiere que establezca la propiedad `cacheAsBitmap` como `true`, consulte "Enmascaramiento del canal alfa" en la página 403.

**NOTA**

No se puede aplicar caché directamente a campos de texto. Deberá colocar el texto dentro de un clip de película para poder hacer uso de esta función. Para ver un ejemplo, consulte el archivo de muestra en *directorio de instalación de Flash*\Samples and Tutorials\Samples\ActionScript\FlashType.

Puede consultar un archivo de origen de ejemplo que muestra cómo se puede aplicar la caché de mapa de bits a una instancia. Busque el archivo denominado cacheBitmap.fla en la carpeta Samples del disco duro.

- En Windows, desplácese a unidad de inicio\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\CacheBitmap.
- En Macintosh, desplácese a Disco duro de Macintosh/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/CacheBitmap.

También puede consultar un archivo de origen de ejemplo que muestra cómo aplicar la caché de mapa de bits al texto desplazable. Busque el archivo denominado flashtype.fla en la carpeta Samples del disco duro.

- En Windows, desplácese a unidad de inicio\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\FlashType.
- En Macintosh, desplácese a Disco duro de Macintosh/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/FlashType.

## Cuándo es conveniente activar la caché

La activación de la caché para un clip de película crea una superficie, lo que presenta varias ventajas, como es la mayor velocidad de representación de animaciones vectoriales complejas. Existen varias situaciones en las que deseará activar la caché. Podría parecer que siempre es preferible activar la caché para mejorar el rendimiento de los archivos SWF; sin embargo, hay situaciones en las que la activación de la caché no mejora el rendimiento e incluso lo reduce. En esta sección se describen situaciones en las que debe utilizarse la activación de la caché y en las que se debe emplear clips de película normales.

El rendimiento global de los datos almacenados en caché depende de la complejidad de los datos vectoriales de las instancias, de la cantidad de datos que cambie y de si ha establecido la propiedad `opaqueBackground`. Si cambia zonas pequeñas, la diferencia entre el uso de una superficie y el uso de datos vectoriales puede ser insignificante. Es aconsejable probar ambas situaciones antes de desplegar la aplicación.

Para obtener información sobre el enmascaramiento del canal alfa, que requiere que establezca la propiedad `cacheAsBitmap` como `true`, consulte "Enmascaramiento del canal alfa" en la página 403.

### Cuándo es conveniente utilizar la caché de mapa de bits

A continuación se incluyen situaciones típicas en las que pueden apreciarse ventajas significativas al activar la caché de mapa de bits.

**Imagen de fondo compleja** Una aplicación que contiene una imagen de fondo compleja y detallada de datos de vectoriales (quizás una imagen en la que aplica el comando Trazar mapa de bits o ilustraciones que ha creado en Adobe Illustrator). Podría animar los caracteres del fondo, lo que ralentizaría la animación porque el fondo necesita la regeneración constante de los datos vectoriales. Para mejorar el rendimiento, puede seleccionar el contenido, almacenarlo en un clip de película y establecer la propiedad `opaqueBackground` como `true`. El fondo se representa como mapa de bits y puede volverse a dibujar rápidamente, por lo que la animación se reproduce con mucha mayor velocidad.

**Campo de texto con desplazamiento** Una aplicación que muestra una gran cantidad de texto en un campo de texto con desplazamiento. Puede colocar el campo de texto en un clip de película que establezca como desplazable con límites con desplazamiento (la propiedad `scrollRect`). De este modo, permite un desplazamiento rápido por los píxeles de la instancia especificada. Cuando el usuario se desplaza por la instancia del clip de película, Flash mueve hacia arriba los píxeles desplazados y genera la región recién expuesta en lugar de regenerar todo el campo de texto.

**Sistema de ventanas** Una aplicación con un complejo sistema de ventanas superpuestas. Cada ventana puede abrirse o cerrarse (por ejemplo, las ventanas de un navegador Web) Si marca cada ventana como una superficie (establece la propiedad `cacheAsBitmap` como `true`), cada ventana se aísla y se almacena en caché. Los usuarios pueden arrastrar las ventanas para que se puedan superponer y cada ventana no necesita regenerar el contenido vectorial.

Todas estas situaciones mejoran el nivel de respuesta e interactividad de la aplicación al optimizar los gráficos vectoriales.

Puede consultar un archivo de origen de ejemplo que muestra cómo se puede aplicar la caché de mapa de bits a una instancia. Busque el archivo denominado cacheBitmap.fla en la carpeta Samples del disco duro.

- En Windows, desplácese a unidad de inicio\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\CacheBitmap.
- En Macintosh, desplácese a Disco duro de Macintosh/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/CacheBitmap.

También puede consultar un archivo de origen de ejemplo que muestra cómo aplicar la caché de mapa de bits al texto desplazable. Busque el archivo denominado flashtype.fla en la carpeta Samples del disco duro.

- En Windows, desplácese a unidad de inicio\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\FlashType.
- En Macintosh, desplácese a Disco duro de Macintosh/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/FlashType.

### Cuándo es conveniente evitar utilizar la caché de mapa de bits

La utilización inadecuada de esta función puede afectar negativamente al archivo SWF. Al desarrollar un archivo FLA que utilice superficies, tenga en cuenta las siguientes directrices:

- No haga un uso abusivo de las superficies (clips de película para los que está activada la caché). Cada superficie utiliza más memoria que un clip de película normal, lo que significa que sólo deberá utilizar las superficies cuando necesite mejorar el rendimiento de la representación.

  Un mapa de bits en caché utiliza bastante más memoria que una instancia de clip de película normal. Por ejemplo, si el clip de película del escenario tiene un tamaño de 250 por 250 píxeles, al almacenarse en caché podría utilizar 250 KB en lugar de 1 KB cuando se trata de una instancia de clip de película normal (no está en caché).
- Evite aplicar zoom a las superficies en caché. Si utiliza en exceso la caché de mapa de bits, se consume una gran cantidad de memoria (consulte el apartado anterior), especialmente si aumenta el contexto.
- Utilice superficies para instancias de clips de película que sean principalmente estáticas (sin animación). Puede arrastrar o mover la instancia, pero el contenido de la instancia no debe incluir demasiada animación ni cambiar mucho. Por ejemplo, si gira o transforma una instancia, ésta cambia entre la superficie y los datos vectoriales, lo que dificulta el procesamiento y afecta de forma negativa al archivo SWF.
- Si mezcla superficies con datos vectoriales, aumentará la cantidad de procesamiento que debe llevar a cabo Flash Player (y, a veces, el equipo). Agrupe las superficies en la medida de lo posible; por ejemplo, al crear aplicaciones de ventanas.

## Asignación de caché para un clip de película

Para asignar caché a una instancia de clip de película, necesita establecer la propiedad `cacheAsBitmap` como `true`. Una vez que haya establecido la propiedad `cacheAsBitmap` con el valor `true`, puede que observe que la instancia de clip de película ajusta automáticamente los píxeles a coordenadas enteras. Cuando pruebe el archivo SWF, debería apreciar un aumento considerable en la velocidad de representación de animaciones vectoriales complejas.

Si se dan una o varias de las siguientes condiciones, no se crea ninguna superficie (mapa de bits en caché) aunque `cacheAsBitmap` se haya establecido como `true`:

- El mapa de bits tiene una altura o una anchura superior a 2880 píxeles.
- El mapa de bits no puede asignarse (error de memoria insuficiente).

### Para asignar caché a un clip de película:

1. Cree un nuevo documento de Flash y asigne al archivo el nombre **cachebitmap.fla**.
2. Escriba **24** en el cuadro de texto fps del inspector de propiedades (Ventana > Propiedades > Propiedades).
3. Cree o importe un gráfico vectorial complejo al archivo FLA.

   Encontrará un gráfico vectorial complejo en el archivo de origen terminado para este ejemplo en el siguiente directorio:

   - En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\CacheBitmap.
   - En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/CacheBitmap.
4. Seleccione el gráfico vectorial y luego Modificar > Convertir en símbolo.
5. Escriba **star** en el cuadro de texto Nombre y haga clic en Avanzado (si el cuadro de diálogo no está aún expandido).
6. Seleccione Exportar para ActionScript (que también selecciona Exportar en primer fotograma).
7. Introduzca **star_id** en el cuadro de texto Identificador.
8. Haga clic en Aceptar para crear el símbolo del clip de película, con el identificador de vinculación de Star.
9. Seleccione el fotograma 1 de la línea de tiempo y añada el siguiente código ActionScript en el panel Acciones:

```
import mx.transitions.Tween;

var star_array:Array = new Array();
for (var i:Number = 0; i < 20; i++) {
  makeStar();
}
function makeStar():Void {
  var depth:Number = this.getNextHighestDepth();
  var star_mc:MovieClip = this.attachMovie("star_id", "star" + depth,
  depth);
  star_mc.onEnterFrame = function() {
    star_mc._rotation += 5;
  }
  star_mc._y = Math.round(Math.random() * Stage.height - star_mc._height
  / 2);
  var star_tween:Tween = new Tween(star_mc, "_x", null, 0, Stage.width,
  (Math.random() * 5) + 5, true);
  star_tween.onMotionFinished = function():Void  {
    star_tween.yoyo();
  };
  star_array.push(star_mc);
}
```

```
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function():Void {
  var star_mc:MovieClip;
  for (var i:Number = 0; i < star_array.length; i++) {
    star_mc = star_array[i];
    star_mc.cacheAsBitmap = !star_mc.cacheAsBitmap;
  }
}
Mouse.addListener(mouseListener);
```

10. Seleccione Control > Probar película para probar el documento.
11. Haga clic en cualquier parte del escenario para activar la caché de mapa de bits.

    Advertirá que la animación cambia de aparecer moviéndose a un fotograma por segundo, a una animación suave donde las instancias se animan hacia atrás y hacia adelante en el escenario. Cuando hace clic en el escenario, cambia el valor de `cacheAsBitmap` entre `true` y `false`.

Si activa y desactiva la caché, como se ha mostrado en el ejemplo anterior, se liberan los datos asignados a la caché. También puede aplicar este código a la instancia Button. Consulte `%{cacheAsBitmap (propiedad Button.cacheAsBitmap)}%` en *Referencia del lenguaje ActionScript 2.0*.

Para ver ejemplos de desplazamiento de clips de película, consulte `%{scrollRect (propiedad MovieClip.scrollRect)}%` en *Referencia del lenguaje ActionScript 2.0*. Para obtener información sobre el enmascaramiento del canal alfa, que requiere que establezca la propiedad `cacheAsBitmap` como `true`, consulte "Enmascaramiento del canal alfa" en la página 403.

Puede consultar un archivo de origen de ejemplo que muestra cómo se puede aplicar la caché de mapa de bits a una instancia. Busque el archivo denominado cacheBitmap.fla en la carpeta Samples del disco duro.

- En Windows, desplácese a unidad de inicio\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\CacheBitmap.
- En Macintosh, desplácese a Disco duro de Macintosh/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/CacheBitmap.

También puede consultar un archivo de origen de ejemplo que muestra cómo aplicar la caché de mapa de bits al texto desplazable. Busque el archivo denominado flashtype.fla en la carpeta Samples del disco duro.

- En Windows, desplácese a unidad de inicio\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\FlashType.
- En Macintosh, desplácese a Disco duro de Macintosh/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/FlashType.

# Definición del fondo de un clip de película

Puede definir un fondo opaco para el clip de película. Por ejemplo, cuando tiene un fondo que contiene complejos gráficos vectoriales, puede establecer la propiedad `opaqueBackground` en un color especificado (normalmente el mismo color del escenario). El fondo se trata entonces como un mapa de bits, lo que ayuda a optimizar el rendimiento.

Cuando establece la propiedad `cacheAsBitmap` como `true` y la propiedad `opaqueBackground` en un color especificado, la propiedad opaqueBackground permite que el mapa de bits interno sea opaco y se represente más rápido. Si no establece `cacheAsBitmap` como true, la propiedad `opaqueBackground` añade una forma cuadrada vectorial opaca al fondo de la instancia del clip de película. No crea un mapa de bits automáticamente.

El ejemplo siguiente muestra cómo definir el fondo de un clip de película para optimizar el rendimiento.

### Para definir el fondo de un clip de película:

1. Cree un nuevo documento de Flash denominado **background.fla**.
2. Dibuje un círculo azul en el escenario.
3. Seleccione el círculo azul y, a continuación, elija Modificar > Convertir en símbolo.
4. Seleccione la opción Clip de película y haga clic en Aceptar.
5. Seleccione la instancia en el escenario y escriba **my_mc** en el cuadro de texto Nombre de instancia del inspector de propiedades.
6. Seleccione el fotograma 1 de la línea de tiempo y, en el panel Acciones, introduzca el siguiente código:

```
/* When you set cacheAsBitmap, the internal bitmap is opaque and renders
  faster. */
my_mc.cacheAsBitmap = true;
my_mc.opaqueBackground = 0xFF0000;
```

7. Seleccione Control > Probar película para probar el documento.

   El clip de película aparece en el escenario con el color de fondo que haya especificado.

Para más información sobre esta propiedad, consulte `%{opaqueBackground (propiedad MovieClip.opaqueBackground)}%` en *Referencia del lenguaje ActionScript 2.0*.

# Utilización de clips de película como máscaras

Puede utilizar un clip de película como una máscara para crear un agujero a través del cual se ve el contenido de otro clip de película. El clip de película de máscara reproduce todos los fotogramas de su línea de tiempo, igual que un clip de película normal. Puede hacer que el clip de película de máscara se pueda arrastrar, animarlo a lo largo de una guía de movimiento, utilizar formas separadas en una sola máscara o cambiar el tamaño de una máscara de forma dinámica. También puede utilizar ActionScript para activar y desactivar una máscara.

No puede utilizar una máscara para enmascarar otra máscara ni establecer la propiedad `_alpha` de un clip de película de máscara. En un clip de película que se usa como máscara, sólo se pueden utilizar los rellenos; los trazos se pasan por alto.

### Para crear una máscara:

1. Cree un cuadrado en el escenario con la herramienta Rectángulo.
2. Seleccione el cuadrado y presione F8 para convertirlo en un clip de película.

   Esta instancia es su máscara.
3. En el inspector de propiedades, escriba **mask_mc** en el cuadro de texto Nombre de instancia.

   El clip de película con máscara se revela bajo todas las zonas opacas (no transparentes) del clip de película que actúa como máscara.
4. Seleccione el fotograma 1 de la línea de tiempo.
5. Abra el panel Acciones (Ventana > Acciones), si todavía no está abierto.
6. En el panel Acciones, especifique el siguiente código:

```
System.security.allowDomain("http://www.helpexamples.com");

this.createEmptyMovieClip("img_mc", 10);
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip):Void {
  target_mc.setMask(mask_mc);
}
var my_mcl:MovieClipLoader = new MovieClipLoader();
my_mcl.addListener(mclListener);
my_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img_mc);
```

7. Seleccione Control > Probar película para probar el documento.

   Un archivo JPEG externo se carga en el archivo SWF durante la ejecución y queda enmascarado por la figura que ha dibujado anteriormente en el escenario.

Para obtener información detallada, consulte %{setMask (`método MovieClip.setMask`)}% en *Referencia del lenguaje ActionScript 2.0*.

## Enmascaramiento de fuentes de dispositivo

Puede utilizar un clip de película para enmascarar un texto configurado en una fuente de dispositivo. Para que la máscara de un clip de película de una fuente de dispositivo funcione correctamente, el usuario deberá tener Flash Player 6 (6.0.40.0) o posterior.

Cuando se utiliza un clip de película para enmascarar texto configurado en una fuente de dispositivo, el recuadro de delimitación rectangular de la máscara se utiliza como la forma de máscara. Es decir, si crea una máscara de clip de película que no es rectangular para la fuente de dispositivo en el entorno de edición de Flash, la máscara que aparecerá en el archivo SWF tendrá la forma del recuadro de delimitación rectangular y no la de la máscara en sí.

Las fuentes de dispositivo sólo se pueden enmascarar utilizando un clip de película como máscara. No se pueden enmascarar fuentes de dispositivo mediante una capa de máscara en el escenario.

## Enmascaramiento del canal alfa

Se admite el enmascaramiento del canal alfa si tanto la máscara como los clips de película enmascarados utilizan la caché de mapa de bits. De este modo, le permite también utilizar un filtro en la máscara independientemente del filtro que se aplica al propio enmascarado.

Para ver un enmascaramiento del canal alfa, descargue el archivo de ejemplo en www.macromedia.com/go/flash_samples_es.

En este archivo de ejemplo, la máscara es un óvalo (`oval_mask`) al que se le aplica un valor de alfa de 50% y un filtro de desenfoque. El archivo enmascarado (`flower_maskee`) tiene un valor alfa de 100% y no se le ha aplicado ningún filtro. A los dos clips de película se les ha aplicado la caché de mapa de bits en tiempo de ejecución en el inspector de propiedades.

En el panel Acciones, se coloca el siguiente código en el fotograma 1 de la línea de tiempo:

```
flower_maskee.setMask(oval_mask);
```

Al probar el documento (Control > Probar película), el enmascarado se mezcla con alfa a través de la máscara.

**NOTA** Las capas de máscara no admiten el enmascaramiento del canal alfa. Debe utilizar código ActionScript para aplicar una máscara y utilizar la caché de mapa de bits en tiempo de ejecución.

# Gestión de eventos de clip de película

Los clips de película pueden responder a eventos de usuario, tales como presionar los botones del ratón o teclas, así como a eventos a nivel de sistema, tales como la carga inicial de un clip de película en el escenario. ActionScript proporciona dos maneras de gestionar eventos de clip de película: a través de los métodos de controlador de eventos y los controladores de eventos `onClipEvent()` y `on()`. Para más información sobre la gestión de eventos de clips de película, consulte el Capítulo 9, "Gestión de eventos".

# Asignación de una clase a un símbolo de clip de película

Mediante ActionScript 2.0, puede crear una clase para ampliar el comportamiento de la clase MovieClip incorporada y, después, utilizar el cuadro de diálogo Propiedades de vinculación para asignar dicha clase a un símbolo de biblioteca de clip de película. Siempre que se crea una instancia del clip de película al que se asignó la clase, ésta adopta las propiedades y los comportamientos definidos por la clase asignada al mismo. (Para más información sobre ActionScript 2.0, consulte el "Ejemplo: Escritura de clases personalizadas" en la página 233.)

En una subclase de la clase MovieClip, puede proporcionar definiciones de método para los métodos MovieClip incorporados y los controladores de eventos, como `onEnterFrame` y `onRelease`. Con el procedimiento siguiente se crea una clase denominada MoveRight que se amplía a la clase MovieClip; MoveRight define un controlador `onPress` que mueve el clip 20 píxeles hacia la derecha cuando el usuario hace clic en el clip de película. El segundo procedimiento consiste en crear un símbolo de clip de película en un nuevo documento de Flash (FLA) y en asignar la clase MoveRight a dicho símbolo.

### Para crear un subclase de clip de película:

1. Cree un nuevo directorio llamado **BallTest**.
2. Seleccione Archivo > Nuevo y seleccione Archivo ActionScript de la lista de tipos de documentos para crear un nuevo archivo ActionScript.
3. Introduzca el código siguiente en el archivo de script:

```
// La clase MoveRight mueve el clip hacia la derecha 20 píxeles al hacer
  clic
class MoveRight extends MovieClip {
  public function onPress() {
    this._x += 20;
  }
}
```

4. Guarde el documento como MoveRight.as en el directorio BallTest.

**Para asignar la clase a un símbolo de clip de película:**

1. En Flash, seleccione Archivo > Nuevo y, en la lista de tipos de archivo, seleccione Documento de Flash y pulse Aceptar.
2. Dibuje un círculo en el escenario con la herramienta Óvalo.
3. Seleccione el círculo y luego Modificar > Convertir en símbolo.
4. En el cuadro de diálogo Convertir en símbolo, seleccione Clip de película como comportamiento del símbolo y escriba **ball_mc** en el cuadro de texto Nombre.
5. Seleccione Avanzado para mostrar las opciones de Vinculación, si es que no están ya visibles.
6. Seleccione la opción Exportar para ActionScript y escriba **MoveRight** en el cuadro de texto Clase. Haga clic en Aceptar.
7. Guarde el archivo como ball.fla en el directorio BallTest (el directorio que contiene el archivo MoveRight.as).
8. Pruebe el documento de Flash (Control > Probar película).

   Cada vez que haga clic en el clip de película, éste se moverá 20 píxeles a la derecha.

Si crea propiedades de componente para una clase y desea que un clip de película herede dichas propiedades, deberá realizar un paso adicional: con el símbolo del clip de película seleccionado en el panel Biblioteca, seleccione Definición de componente del menú emergente de Biblioteca y escriba el nombre de la nueva clase en el cuadro Clase.

# Inicialización de las propiedades de clase

En el ejemplo presentado en el segundo procedimiento en "Asignación de una clase a un símbolo de clip de película", ha añadido la instancia del símbolo Ball al escenario durante el proceso de edición. Como se ha descrito en "Adición de parámetros a clips de película creados de forma dinámica" en la página 387, puede asignar parámetros a los clips que cree en tiempo de ejecución mediante el parámetro *initObject* de `attachMovie()` y `duplicateMovie()`. Puede utilizar esta función para inicializar las propiedades de la clase que vaya a asignar a un clip de película.

Por ejemplo, la clase siguiente, denominada MoveRightDistance, es una variación de la clase MoveRight (consulte "Asignación de una clase a un símbolo de clip de película" en la página 404). La diferencia es una nueva propiedad denominada `distance`, cuyo valor determina el número de píxeles que se desplaza un clip de película cada vez que se hace clic en él.

### Para pasar argumentos a una clase personalizada:

1. Cree un nuevo documento ActionScript y guárdelo como **MoveRightDistance.as**.
2. Escriba el siguiente código ActionScript en la ventana Script:

```
// La clase MoveRightDistance mueve el clip hacia la derecha 5 píxeles
  cada fotograma
class MoveRightDistance extends MovieClip {
  // La propiedad distance determina cuántos
  // píxeles debe desplazarse el clip cada vez que se presiona el botón
  del ratón.
  var distance:Number;
  function onPress() {
    this._x += this.distance;
  }
}
```

3. Guarde el trabajo.
4. Cree un nuevo documento de Flash y guárdelo como **MoveRightDistance.fla** en el mismo directorio que el del archivo de clase.
5. Cree un símbolo de clip de película que contenga una forma vectorial, como un óvalo, y elimine todo el contenido del escenario.

   Sólo necesita un símbolo de clip de película en la biblioteca para este ejemplo.
6. En el panel Biblioteca, haga clic con el botón derecho (Windows) o haga clic con la tecla Control presionada (Macintosh) en el símbolo y seleccione Vinculación en el menú contextual.
7. Asigne el identificador de vinculación **Ball** al símbolo.
8. Escriba **MoveRightDistance** en el cuadro de texto Clase de AS 2.0.
9. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
this.attachMovie("Ball", "ball50_mc", 10, {distance:50});
this.attachMovie("Ball", "ball125_mc", 20, {distance:125});
```

   Este código crea dos nuevas instancias del símbolo en la línea de tiempo raíz del archivo SWF. La primera instancia, denominada `ball50_mc`, se mueve 50 píxeles cada vez que se hace clic en ella; la segunda, denominada `ball125_mc`, se mueve 125 píxeles cada vez que se hace clic en ella.
10. Seleccione Control > Probar película para probar el archivo SWF.

CAPÍTULO 12

# Utilización de texto y cadenas

12

Muchas de las aplicaciones, presentaciones o gráficos que cree con Macromedia Flash Professional 8 o Macromedia Flash Basic 8 incluirán algún tipo de texto. Puede utilizar muchos tipos de texto diferentes. Podría utilizar texto estático en los diseños y texto dinámico para fragmentos de texto de mayor tamaño. O bien podría utilizar texto de entrada para capturar la entrada del usuario y texto en una imagen para el diseño del fondo. Puede crear campos de texto con la herramienta de edición de Flash o mediante ActionScript.

Una forma de mostrar texto consiste en manipular cadenas mediante código antes de que se carguen y aparezcan en el escenario durante la ejecución. Existen varias formas de utilizar cadenas en una aplicación, como, por ejemplo, enviarlas a un servidor y recuperar una respuesta, analizar cadenas de una matriz o validar cadenas introducidas por un usuario en un campo de texto.

En este capítulo se describen varias formas de utilizar texto y cadenas en las aplicaciones, prestando especial atención al uso de código para manipular texto.

En la siguiente lista se describe la terminología utilizada en este capítulo.

**Mapa de bits** El texto de mapa de bits no utiliza variaciones de color para que sus bordes dentados se muestren más suaves, a diferencia del texto suavizado (consulte la definición siguiente).

**Suavizado** Se utiliza para afinar el texto de modo que los bordes de los caracteres no aparezcan excesivamente dentados en la pantalla. La opción de suavizado en Flash permite obtener un texto más legible alineando los contornos del texto a los límites de los píxeles y resulta especialmente efectivo para representar con mayor claridad los tamaños de fuente más pequeños.

**Caracteres** Son letras, numerales y puntuación que se combinan para formar cadenas.

**Fuentes de dispositivo** Son fuentes especiales en Flash que no están incorporadas en un archivo SWF. En lugar de ello, Flash Player utiliza la fuente disponible en el equipo local más parecida a la fuente de dispositivo. Dado que los contornos de fuente no están incorporados, el tamaño del archivo SWF es menor que cuando se utilizan contornos de fuente incorporados. No obstante, como las fuentes de dispositivo no están incorporadas, el aspecto del texto que se cree puede ser diferente al esperado en los sistemas que no tengan instalada una fuente que corresponda a la fuente de dispositivo. Flash incluye tres fuentes de dispositivo: _sans (similar a la Helvetica y Arial), _serif (similar a la Times Roman) y _typewriter (similar a la Courier).

**Fuentes** Conjuntos de caracteres con un tipo de fuente, un estilo y un tamaño similar.

**Cadena** Secuencia de caracteres.

**Texto** Serie de una o más cadenas que se puede mostrar en un campo de texto o dentro de un componente de la interfaz de usuario.

**Campos de texto** Elemento visual del escenario que permite mostrar texto al usuario. De forma similar a lo que ocurre con los campos de introducción de texto o los controles de formulario de área de texto en HTML, Flash permite definir los campos de texto como editables (sólo lectura), aplicar formato HTML, activar la compatibilidad con varias líneas, crear máscaras de contraseñas o aplicar una hoja de estilos CSS (hoja de estilos en cascada, Cascading Style Sheet) al texto con formato HTML.

**Formato de texto** Se puede aplicar formato a un campo de texto o a ciertos caracteres dentro de un campo de texto. A continuación se muestran algunos ejemplos de opciones de formato de texto que se pueden aplicar: alineación, sangrado, negrita, color, tamaño de fuente, anchura de márgenes, cursiva y espaciado entre caracteres.

Para más información sobre el texto, consulte los siguientes temas:

# Campos de texto

Un campo de texto dinámico o de entrada es un objeto TextField (una instancia de la clase TextField). Al crear un campo de texto en el entorno de edición, puede asignarle un nombre de instancia en el inspector de propiedades. Puede utilizar el nombre de instancia en sentencias de ActionScript para establecer, modificar y dar formato al campo de texto y a su contenido mediante las clases TextField y TextFormat.

Los campos de texto se pueden crear utilizando la interfaz de usuario o mediante ActionScript. Puede crear los siguientes tipos de campos de texto en Flash:

**Texto estático** Utilice texto estático para mostrar caracteres que no vayan a cambiar, textos breves o para mostrar fuentes especiales no disponibles en la mayoría de los equipos. También puede mostrar fuentes no habituales incorporando caracteres para campos de texto dinámico.

**Texto dinámico** Utilice campos de texto dinámico si necesita mostrar caracteres que se actualizan o cambian durante la ejecución. Asimismo, puede cargar texto en campos de texto dinámico.

**Texto introducido** Utilice campos de introducción de texto si necesita capturar los datos introducidos por el usuario. Los usuarios pueden escribir en estos campos de texto.

**Componentes de texto** Puede utilizar componentes TextArea o TextInput para mostrar o capturar texto en las aplicaciones. El componente TextArea es similar a un campo de texto dinámico con barras de desplazamiento incorporadas. El componente TextInput es similar a un campo de introducción de texto. Ambos componentes tienen funcionalidad adicional con respecto a los campos de texto equivalentes; sin embargo, añaden un tamaño de archivo mayor a la aplicación.

NOTA

Todos los campos de texto son compatibles con Unicode. Para obtener información sobre Unicode, consulte "Cadenas y la clase String" en la página 480.

Los métodos de la clase TextField permiten establecer, seleccionar y manipular texto de un campo de texto dinámico o de entrada que se cree durante la edición o la ejecución. Para más información, consulte "Utilización de la clase TextField" en la página 410. Para obtener información sobre depuración de campos de texto durante la ejecución, consulte "Visualización de las propiedades de un campo de texto para la depuración" en la página 772.

ActionScript también proporciona diversas maneras de dar formato a los textos durante la ejecución. La clase TextFormat permite definir el formato de carácter y de párrafo para los objetos TextField (consulte "Utilización de la clase TextFormat" en la página 448). Flash Player también admite un subconjunto de etiquetas HTML que puede utilizar para dar formato al texto (consulte "Utilización de texto en formato HTML" en la página 465). Flash Player 7 y las versiones posteriores admiten la etiqueta HTML `img`, que permite incorporar no sólo imágenes externas, sino también archivos SWF externos, así como clips de película que residen en la biblioteca (consulte "Etiqueta de imagen" en la página 469).

En Flash Player 7 y versiones posteriores, puede aplicar estilos CSS a los campos de texto mediante la clase TextField.StyleSheet. Puede utilizar estilos CSS para aplicar un estilo a las etiquetas HTML incorporadas, definir nuevas etiquetas de formato o aplicar estilos. Para más información sobre el uso de CSS, consulte "Aplicación de formato al texto con hojas de estilos en cascada" en la página 451.

También puede asignar texto con formato HTML, que opcionalmente puede utilizar estilos CSS, directamente a un campo de texto. En Flash Player 7 y versiones posteriores, el texto HTML que asigna a un campo de texto puede contener elementos multimedia incorporados (clips de película, archivos SWF y archivos JPEG). En Flash Player 8, también puede cargar dinámicamente imágenes PNG, GIF y JPEG *progresivas* (Flash Player 7 no es compatible con las imágenes JPEG progresivas). El texto se ajustará alrededor del elemento multimedia incorporado, igual que los navegadores Web ajustan texto alrededor del elemento multimedia incorporado en un documento HTML. Para más información, consulte "Etiqueta de imagen" en la página 469.

Para ver una comparación de los términos texto, cadenas, etc., consulte la introducción de este capítulo, "Utilización de texto y cadenas" en la página 407.

## Utilización de la clase TextField

La clase TextField representa cualquier campo de texto dinámico o de introducción de texto (editable) creado mediante la herramienta Texto de Flash. Utilice los métodos y propiedades de esta clase para controlar los campos de texto durante la ejecución. Los objetos TextField admiten las mismas propiedades que los objetos MovieClip, a excepción de las propiedades `_currentframe`, `_droptarget`, `_framesloaded` y `_totalframes`. Puede obtener y establecer propiedades e invocar métodos para campos de texto de forma dinámica.

Para controlar un campo de texto dinámico o de introducción de texto mediante ActionScript, debe asignar al campo de texto un nombre de instancia en el inspector de propiedades. Con ello, podrá hacer referencia al campo de texto por el nombre de instancia y utilizar los métodos y propiedades de la clase TextField para controlar el contenido o el aspecto básico del campo de texto.

También puede crear objetos TextField durante la ejecución y asignarles nombres de instancia mediante el método `MovieClip.createTextField()`. Para más información, consulte "Creación de campos de texto durante la ejecución" en la página 414.

Para más información sobre la utilización de la clase TextField, consulte los siguientes temas:

- "Asignación de texto a un campo de texto durante la ejecución" en la página 411
- "Nombres de instancia y de variable de los campos de texto" en la página 413

Puede encontrar archivos de origen de ejemplo que muestran cómo utilizar los campos de texto mediante ActionScript. Los archivos de origen se denominan textfieldsA.fla y textfieldsB.fla y se encuentran en la carpeta Samples del disco duro:

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\TextFields.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/TextFields.

## Asignación de texto a un campo de texto durante la ejecución

Cuando cree aplicaciones con Flash, puede que desee cargar texto desde un origen externo, como un archivo de texto, un archivo XML o incluso un servicio Web remoto. Flash permite controlar en gran medida cómo se crea y se muestra el texto en el escenario, por ejemplo, admitiendo texto sin formato o con formato HTML y XML y hojas de estilos externas. O bien, también se puede utilizar ActionScript para definir una hoja de estilos.

Para asignar texto a un campo de texto, puede utilizar las propiedades `TextField.text` o `TextField.htmlText`. Si introdujo un valor en el campo de texto de variable del inspector de propiedades, también puede asignar un valor al campo de texto creando una variable con el nombre especificado. Si utiliza la versión 2 de la arquitectura de componentes de Macromedia en el documento de Flash, también puede asignar valores creando vinculaciones entre los componentes.

En el siguiente ejercicio se asigna texto a un campo de texto durante la ejecución.

### Para asignar texto a un campo de texto durante la ejecución:

1. Con la herramienta Texto, cree un campo de texto en el escenario.
2. Con el campo de texto seleccionado, en el inspector de propiedades (Ventana > Propiedades > Propiedades), seleccione Introducción de texto en el menú emergente Tipo de texto e introduzca **headline_txt** en el cuadro de texto Nombre de instancia.

   Los nombres de instancia pueden constar únicamente de letras, números, caracteres de subrayado (_) y símbolos de dólar ($).

3. Seleccione el fotograma 1 en la línea de tiempo y abra el panel Acciones (Ventana > Acciones).
4. Introduzca el código siguiente en el panel Acciones:

```
headline_txt.text = "New articles available on Developer Center";
```

5. Seleccione Control > Probar película para probar el documento de Flash.

También puede crear un campo de texto con código ActionScript y luego asignarle texto. Escriba el siguiente código ActionScript en el fotograma 1 de la línea de tiempo principal:

```
this.createTextField("headline_txt", this.getNextHighestDepth(), 100, 100,
  300, 20);
headline_txt.text = "New articles available on Developer Center";
```

Este código crea un nuevo campo de texto con el nombre de instancia `headline_txt`. El campo de texto se crea en la siguiente profundidad superior disponible, en las coordenadas *x* e *y* 100, 100, con una anchura de 200 píxeles y una altura de 20 píxeles. Al probar el archivo SWF (Control > Probar película), aparece el texto "New articles available on Developer Center" en el escenario.

### Para crear un campo de texto con formato HTML:

Siga uno de los dos pasos siguientes para activar la aplicación de formato HTML al campo de texto:

- Seleccione un campo de texto y haga clic en Generar texto como HTML en el inspector de propiedades.
- Defina la propiedad `html` del campo de texto en `true` mediante ActionScript (consulte el siguiente código de muestra).

Para aplicar formato HTML a un campo de texto mediante ActionScript, escriba el siguiente código ActionScript en el fotograma 1 de la línea de tiempo:

```
this.createTextField("headline_txt", this.getNextHighestDepth(), 100, 100,
  300, 20);
headline_txt.html = true;
headline_txt.htmlText = "New articles available on <i>Developer Center</
  i>.";
```

El código anterior crea dinámicamente un nuevo campo de texto, activa la aplicación de formato HTML y muestra el texto "New articles available on Developer Center" en el escenario, con las palabras "Developer Center" en cursiva.

**ATENCIÓN**

Cuando utiliza texto con formato HTML con un campo de texto (no con componentes) en el escenario, debe asignar el texto a la propiedad `htmlText` del campo de texto en lugar de a la propiedad del texto.

Puede encontrar archivos de origen de ejemplo que muestran cómo utilizar los campos de texto mediante ActionScript. Los archivos de origen se denominan textfieldsA.fla y textfieldsB.fla y se encuentran en la carpeta Samples del disco duro:

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\TextFields.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/TextFields.

## Nombres de instancia y de variable de los campos de texto

En el cuadro de texto Nombre de instancia del inspector de propiedades, deberá asignar un nombre de instancia a un campo de texto para invocar métodos y obtener y establecer propiedades para dicho campo de texto.

En el cuadro de texto Var del inspector de propiedades, puede asignar un nombre de variable a un campo dinámico o de entrada de texto. Posteriormente podrá asignar valores a la variable. Se trata de una funcionalidad desfasada que podría utilizar al crear aplicaciones para versiones anteriores de Flash Player (como Flash Player 4). Al escribir para reproductores más recientes, deberá hacer referencia al texto de un campo de texto mediante su nombre de instancia y código ActionScript.

No confunda el nombre de instancia de un campo de texto con su nombre de variable. El nombre de variable de un campo de texto es una referencia de variable al texto contenido en dicho campo, no una referencia a un objeto.

Por ejemplo, si ha asignado el nombre de variable `myTextVar` a un campo de texto, puede establecer el contenido del campo de texto mediante el código siguiente:

```
var myTextVar:String = "This is what will appear in the text field";
```

Sin embargo, no puede utilizar el nombre de variable `myTextVar` para establecer la propiedad `text` del campo de texto. Deberá utilizar el nombre de instancia como se muestra en el siguiente código:

```
// Esto no funcionará.
myTextVar.text = "A text field variable is not an object reference";

// Para un campo de introducción de texto con el nombre de instancia
  "myField", sí funcionará.
myField.text = "This sets the text property of the myField object";
```

Utilice la propiedad `TextField.text` para controlar el contenido de un campo de texto, a menos que se vaya a utilizar en una versión de Flash Player que no admita la clase TextField. Esto reduce las probabilidades de que se produzcan conflictos de nombres de variables, que podrían provocar un comportamiento inesperado durante la ejecución.

Puede encontrar archivos de origen de ejemplo que muestran cómo utilizar los campos de texto mediante ActionScript. Los archivos de origen se denominan textfieldsA.fla y textfieldsB.fla y se encuentran en la carpeta Samples del disco duro:

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\TextFields.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/TextFields.

## Creación de campos de texto durante la ejecución

Puede utilizar el método `createTextField()` de la clase MovieClip para crear un campo de texto vacío en el escenario durante la ejecución. Este campo de texto nuevo se asocia a la línea de tiempo del clip de película que llama al método.

### Para crear un campo de texto dinámicamente utilizando código ActionScript:

1. Seleccione Archivo > Nuevo y, a continuación, seleccione Documento de Flash para crear un nuevo archivo FLA.
2. Escriba el siguiente código ActionScript en el fotograma 1 de la línea de tiempo principal:

   ```
   this.createTextField("test_txt", 10, 0, 0, 300, 100);
   ```

   Este código crea un campo de texto de 300 x 100 píxeles denominado `test_txt` en la ubicación (0, 0) con una profundidad (orden *z*) de 10.
3. Para acceder a los métodos y propiedades del campo de texto que acaba de crear, utilice el nombre de instancia especificado en el primer parámetro del método `createTextField()`.

   Por ejemplo, el código siguiente crea un campo de texto denominado `test_txt` y, a continuación, modifica sus propiedades para que sea un campo de texto multilínea con ajuste de texto que se expanda para adaptarse al texto insertado. Posteriormente, asigna texto mediante la propiedad `text` del campo de texto:

   ```
   test_txt.multiline = true;
   test_txt.wordWrap = true;
   test_txt.autoSize = "left";
   test_txt.text = "Create new text fields with the
     MovieClip.createTextField() method.";
   ```
4. Seleccione Control > Probar película para ver el campo de texto.

   El texto se crea durante la ejecución y aparece en el escenario.

Puede utilizar el método `TextField.removeTextField()` para eliminar un campo de texto creado con `createTextField()`. El método `removeTextField()` no funciona en los campos de texto colocados por la línea de tiempo durante la edición.

Para más información, consulte %{createTextField (método MovieClip.createTextField)}% y %{removeTextField (método TextField.removeTextField)}% en *Referencia del lenguaje ActionScript 2.0*.

| NOTA | Algunas propiedades de TextField, como `_rotation`, no están disponibles cuando se crean campos de texto durante la ejecución. Sólo se puede girar un campo de texto si éste utiliza fuentes incorporadas. Consulte "Para incorporar un símbolo de fuente:" en la página 427. |
|---|---|

Puede encontrar archivos de origen de ejemplo que muestran cómo utilizar los campos de texto mediante ActionScript. Los archivos de origen se denominan textfieldsA.fla y textfieldsB.fla y se encuentran en la carpeta Samples del disco duro:

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\TextFields.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/TextFields.

## Manipulación de campos de texto

Existen distintas formas de manipular los campos de texto creados en un archivo FLA. Puede manipular un campo de texto siempre y cuando asigne un nombre de instancia en el inspector de propiedades, o bien puede asignárselo mediante código si crea el campo utilizando código. El siguiente es un ejemplo sencillo en el que se crea un campo de texto, se le asigna texto y se modifica la propiedad border del campo:

```
this.createTextField("pigeon_txt", this.getNextHighestDepth(), 100, 100,
  200, 20);
pigeon_txt.text = "I like seeds";
pigeon_txt.border = true;
```

Para ver la lista completa de propiedades de la clase TextField, consulte *Referencia del lenguaje ActionScript 2.0*.

Para ver ejemplos de cómo manipular campos de texto, consulte las siguientes secciones:

- "Cambio de la posición de un campo de texto" en la página 416
- "Cambio de las dimensiones de un campo de texto durante la ejecución" en la página 417

Puede encontrar archivos de origen de ejemplo que muestran cómo utilizar los campos de texto mediante ActionScript. Los archivos de origen se denominan textfieldsA.fla y textfieldsB.fla y se encuentran en la carpeta Samples del disco duro:

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\TextFields.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/TextFields.

## Cambio de la posición de un campo de texto

Puede cambiar la posición de un campo de texto en el escenario durante la ejecución. Deberá establecer valores nuevos para las propiedades _x y _y del campo de texto, como se muestra en el siguiente ejemplo.

### Para cambiar la posición de un campo de texto utilizando código ActionScript:

1. Cree un archivo FLA nuevo y guárdelo como **positionText.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
this.createTextField("my_txt", 10, 0, 0, 300, 200);
my_txt.border = true;
my_txt.text = "Hello world";
my_txt._x = (Stage.width - my_txt._width) / 2;
my_txt._y = (Stage.height - my_txt._height) / 2;
```

3. Guarde el documento de Flash y seleccione Control > Probar película para ver el campo de texto centrado en el escenario.

Puede encontrar archivos de origen de ejemplo que muestran cómo utilizar los campos de texto mediante ActionScript. Los archivos de origen se denominan textfieldsA.fla y textfieldsB.fla y se encuentran en la carpeta Samples del disco duro:

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\TextFields.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/TextFields.

## Cambio de las dimensiones de un campo de texto durante la ejecución

Es posible que tenga que obtener o establecer las dimensiones de un campo de texto dinámicamente durante la ejecución en lugar hacerlo a través del entorno de edición. El siguiente ejemplo crea un campo de texto en una línea de tiempo y establece sus dimensiones iniciales en 100 píxeles de anchura por 21 píxeles de altura. Posteriormente, se modifica el tamaño del campo de texto a 300 píxeles de anchura por 200 píxeles de altura y se cambia su posición al centro del escenario.

### Para cambiar el tamaño de un campo de texto utilizando código ActionScript:

1. Cree un nuevo documento de Flash y guárdelo como **resizeText.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
this.createTextField("my_txt", 10, 0, 0, 100, 21);
my_txt.border = true;
my_txt.multiline = true;
my_txt.text = "Hello world";
my_txt.wordWrap = true;
my_txt._width = 300;
my_txt._height = 200;
my_txt._x = (Stage.width - my_txt._width) / 2;
my_txt._y = (Stage.height - my_txt._height) / 2;
```

3. Guarde el documento de Flash y seleccione Control > Probar película para comprobar el resultado en el entorno de edición.

En el ejemplo anterior, se modificó el tamaño de un campo de texto creado dinámicamente a 300 píxeles por 200 píxeles durante la ejecución, pero al cargar contenido de un sitio Web externo sin conocer la cantidad de contenido que se va a devolver, esta técnica podría no ser adecuada para sus necesidades. Afortunadamente, Flash incluye una propiedad `TextField.autoSize` que puede utilizar para cambiar automáticamente el tamaño de un campo de texto para que se ajuste al contenido. En el siguiente ejemplo se demuestra cómo puede utilizar la propiedad `TextField.autoSize` para cambiar el tamaño del campo de texto después de que se añada texto al campo.

### Para cambiar automáticamente el tamaño de los campos de texto en función de su contenido:

1. Cree un nuevo documento de Flash y guárdelo como **resizeTextAuto.fla**.
2. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
this.createTextField("my_txt", 10, 10, 10, 160, 120);
my_txt.autoSize = "left";
my_txt.border = true;
my_txt.multiline = true;
my_txt.text = "Lorem ipsum dolor sit amet, consectetur adipisicing elit,
  sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
  enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi
  ut aliquip ex ea commodo consequat. Duis aute irure dolor in
  reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
  pariatur. Excepteur sint occaecat cupidatat non proident, sunt in
  culpa qui officia deserunt mollit anim id est laborum.";
my_txt.wordWrap = true;
```

> **NOTA** Si pega este código directamente en el panel Acciones desde algunas versiones de la Ayuda de Flash, puede que aparezcan saltos de línea en la cadena de texto larga. En este caso, el código no se compilará. Si se encuentra con esta situación, active Caracteres ocultos en el menú emergente del panel Acciones y elimine los caracteres de salto de línea en la cadena de texto larga.

3. Guarde el documento de Flash y seleccione Control > Probar película para ver el documento de Flash en el entorno de edición.

   Flash cambia el tamaño del campo de texto verticalmente, de manera que se muestre todo el contenido sin que los límites del campo lo corten. Si establece la propiedad `my_txt.wordWrap` en `false`, el campo de texto cambia su tamaño horizontalmente para dar cabida al texto.

   Para aplicar una altura máxima al campo de texto con tamaño automático (de forma que la altura del campo de texto no supere los límites del escenario), utilice el siguiente código.

```
if (my_txt._height > 160) {
  my_txt.autoSize = "none";
  my_txt._height = 160;
}
```

Debe añadir alguna funcionalidad de desplazamiento, como una barra, para permitir que los usuarios vean el resto del texto. Como alternativa, puede desplazar el puntero del ratón sobre el texto; este método suele resultar adecuado mientras se prueba este código.

Puede encontrar archivos de origen de ejemplo que muestran cómo utilizar los campos de texto mediante ActionScript. Los archivos de origen se denominan textfieldsA.fla y textfieldsB.fla y se encuentran en la carpeta Samples del disco duro:

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\TextFields.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/TextFields.

# Carga de texto y variables en los campos de texto

Existen varias formas de cargar texto en un documento de Flash, entre las que se encuentran FlashVars, LoadVars, XML o servicios Web. Posiblemente, el método más simple para pasar texto a un documento de Flash sea utilizar la propiedad FlashVars, que pasa cadenas de texto cortas a un documento de Flash a través de las etiquetas `object` y `embed` en el código HTML empleado para incorporar el archivo SWF en una página HTML. Otra forma sencilla de cargar texto o variables en un documento de Flash es utilizar la clase LoadVars, que puede cargar bloques grandes de texto o una serie de variables con codificación URL desde un archivo de texto.

Como puede observarse en los ejemplos anteriores de esta sección, algunas formas de cargar texto en un archivo SWF son más sencillas que otras. Sin embargo, si toma datos de sitios externos, puede que no tenga la posibilidad de elegir el formato de los datos que necesita cargar.

Cada método de carga y/o envío de datos hacia o desde un archivo SWF presenta ventajas e inconvenientes. XML, los servicios Web y Flash Remoting son los métodos de carga de datos externos más versátiles, aunque también resultan los más complicados de aprender. Para obtener información sobre Flash Remoting, consulte www.macromedia.com/support/flashremoting.

FlashVars y LoadVars son mucho más simples, como se muestra en “Utilización de FlashVars para cargar y mostrar texto” en la página 420 y “Utilización de LoadVars para cargar y mostrar texto” en la página 422, aunque pueden resultar mucho más limitados en lo que se refiere a los tipos y formatos de datos que pueden cargar. Asimismo, debe seguir las restricciones de seguridad al enviar y cargar datos. Para obtener información sobre seguridad, consulte el Capítulo 17, “Aspectos básicos de la seguridad” Para más información sobre la carga de datos externos, consulte el Capítulo 16, “Trabajo con datos externos”

En las siguientes secciones se muestran diversas formas de cargar texto y variables en los documentos:

- "Utilización de FlashVars para cargar y mostrar texto" en la página 420
- "Utilización de LoadVars para cargar y mostrar texto" en la página 422
- "Carga de variables mediante LoadVars" en la página 423
- "Carga y visualización de texto de un documento XML" en la página 424

Puede encontrar archivos de origen de ejemplo que muestran cómo utilizar los campos de texto mediante ActionScript. Los archivos de origen se denominan loadText.fla y formattedText.fla y se encuentran en la carpeta Samples del disco duro:

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\LoadText.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/LoadText.

También puede encontrar un archivo de origen que carga texto y aplica formato suavizado además de caché de mapa de bits. El archivo de origen de ejemplo se denomina flashtype.fla y se encuentra en la carpeta Samples del disco duro:

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\FlashType.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/FlashType.

# Utilización de FlashVars para cargar y mostrar texto

El uso de FlashVars es sencillo, pero exige la publicación de los archivos SWF junto con los documentos HTML. Se modifica el código HTML generado y se incluyen las propiedades de FlashVars tanto en la etiqueta `object` como en la etiqueta `embed`. A continuación, se puede probar el documento de Flash mediante la visualización del documento HTML modificado en el navegador Web.

### Para utilizar FlashVars con el fin de pasar variables del código HTML al documento de Flash:

1. Cree un nuevo documento de Flash y guárdelo como **flashvars.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
this.createTextField("my_txt", 10, 10, 10, 100, 21);
my_txt.text = _level0.username;
```

3. Guarde el documento de Flash y seleccione Archivo > Publicar para generar los archivos HTML y SWF.

| NOTA | Un documento HTML se publica, de manera predeterminada, en el mismo directorio que el archivo FLA. Si no se publica un documento HTML, seleccione Archivo > Configuración de publicación y seleccione la ficha Formatos. Asegúrese de que selecciona HTML. |
|---|---|

4. Abra el documento flashvars.html en un editor de texto o HTML.
5. En el documento HTML, modifique el código situado dentro de la etiqueta `object` para que coincida con el siguiente.

   El código que debe añadir está en **negrita**.

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
  codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/
  swflash.cab#version=8,0,0,0" width="550" height="400" id="flashvars"
  align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="flashvars.swf" />
<param name="FlashVars" value="username=Thomas" />
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<embed src="flashvars.swf" FlashVars="username=Thomas" quality="high"
  bgcolor="#ffffff" width="550" height="400" name="flashvars"
  align="middle" allowScriptAccess="sameDomain" type="application/x-
  shockwave-flash" pluginspage="http://www.macromedia.com/go/
  getflashplayer" />
</object>
```

6. Guarde los cambios en el documento HTML.
7. Abra el archivo HTML modificado en un navegador Web.

   El archivo SWF muestra el nombre “Thomas” en el campo de texto creado dinámicamente en el escenario.

Para obtener información sobre seguridad, consulte el Capítulo 17, “Aspectos básicos de la seguridad”

## Utilización de LoadVars para cargar y mostrar texto

También puede cargar contenido en un archivo SWF utilizando la clase LoadVars, que carga texto o variables de un archivo externo en el mismo servidor, o incluso contenido de un servidor diferente. En el siguiente ejemplo se muestra cómo crear dinámicamente un campo de texto y llenarlo con contenido de un archivo de texto remoto.

### Para utilizar LoadVars con el fin de llenar un campo de texto con texto externo:

1. Cree un nuevo documento de Flash y guárdelo como **loadvarsText.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
this.createTextField("my_txt", 10, 10, 10, 320, 100);
my_txt.autoSize = "left";
my_txt.border = true;
my_txt.multiline = true;
my_txt.wordWrap = true;

var lorem_lv:LoadVars = new LoadVars();
lorem_lv.onData = function (src:String):Void {
  if (src != undefined) {
    my_txt.text = src;
  } else {
    my_txt.text = "Unable to load external file.";
  }
}
lorem_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

   El primer bloque de código del fragmento anterior crea un nuevo campo de texto en el escenario y activa varias líneas y el ajuste de texto. El segundo bloque de código define un nuevo objeto LoadVars que se utiliza para cargar un archivo de texto (lorem.txt) de un servidor Web remoto y mostrar su contenido en el campo my_txt creado anteriormente.

3. Guarde el documento de Flash y seleccione Control > Probar película para probar el archivo SWF.

   Tras un breve retardo, Flash muestra el contenido del archivo remoto en el campo de texto del escenario.

Para obtener información sobre seguridad, consulte el Capítulo 17, “Aspectos básicos de la seguridad”

# Carga de variables mediante LoadVars

La clase LoadVars también permite cargar variables en formato con codificación URL, algo similar al paso de variables en una cadena de consulta en un navegador Web. El siguiente ejemplo muestra cómo cargar un archivo de texto remoto en un archivo SWF y mostrar sus variables, `monthNames` y `dayNames`.

### Para cargar variables de un archivo de texto mediante LoadVars:

1. Cree un nuevo documento de Flash y guárdelo como **loadvarsVariables.fla**.
2. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
this.createTextField("my_txt", 10, 10, 10, 320, 100);
my_txt.autoSize = "left";
my_txt.border = true;
my_txt.multiline = true;
my_txt.wordWrap = true;

var lorem_lv:LoadVars = new LoadVars();
lorem_lv.onLoad = function (success:Boolean):Void {
  if (success) {
    my_txt.text = "dayNames: " + lorem_lv.dayNames + "\n\n";
    my_txt.text += "monthNames: " + lorem_lv.monthNames;
  } else {
    my_txt.text = "Unable to load external file.";
  }
}
/* contents of params.txt:
  &monthNames=January,February,...&dayNames=Sunday,Monday,...
*/
lorem_lv.load("http://www.helpexamples.com/flash/params.txt");
```

3. Guarde el documento de Flash y seleccione Control > Probar película del menú principal.

   Dado que está utilizando el método `LoadVars.onLoad()` de `LoadVars.onData()`, Flash analiza las variables y crea variables dentro de la instancia de objeto LoadVars. El archivo de texto externo contiene dos variables, `monthNames` y `dayNames`, ambas con cadenas.

Para obtener información sobre seguridad, consulte el Capítulo 17, “Aspectos básicos de la seguridad”

# Carga y visualización de texto de un documento XML

Los datos XML son un medio de distribución de contenido en Internet que goza de gran popularidad, en parte debido a su estándar de organización y análisis de datos, que cuentan con amplia aceptación. Por esta razón, XML constituye una excelente elección para el envío y la recepción de datos desde Flash; sin embargo, XML resulta algo más difícil de aprender que los métodos LoadVars y FlashVars de carga de datos y visualización de texto.

### Para cargar texto en Flash desde un documento XML externo:

1. Cree un nuevo documento de Flash y guárdelo como **xmlReviews.fla**.
2. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
this.createTextField("my_txt", 10, 10, 10, 320, 100);
my_txt.autoSize = "left";
my_txt.border = true;
my_txt.multiline = true;
my_txt.wordWrap = true;

var reviews_xml:XML = new XML();
reviews_xml.ignoreWhite = true;
reviews_xml.onLoad = function (success:Boolean):Void {
  if (success) {
    var childItems:Array = reviews_xml.firstChild.childNodes;
    for (var i:Number = 0; i < childItems.length; i++) {
      my_txt.text += childItems[i].firstChild.firstChild.nodeValue +
  "\n";
    }
  } else {
    my_txt.text = "Unable to load external file.";
  }
}
reviews_xml.load("http://www.helpexamples.com/flash/xml/reviews.xml");
```

   El primer bloque de código del fragmento anterior crea un nuevo campo de texto en el escenario. Este campo de texto se utiliza para mostrar diversas partes del documento XML que se carga posteriormente. El segundo bloque de código se ocupa de la creación de un objeto XML que se utiliza para cargar el contenido XML. Una vez que Flash ha cargado completamente y analizado los datos, se invoca el controlador de eventos `XML.onLoad()`, que muestra el contenido del paquete XML en el campo de texto.

3. Guarde el documento de Flash y seleccione Control > Probar película para probar el archivo SWF.

   Flash muestra la siguiente salida en el campo de texto del escenario:

```
Item 1
Item 2
...
Item 8
```

Para obtener información sobre seguridad, consulte el Capítulo 17, “Aspectos básicos de la seguridad”

# Utilización de fuentes

Las fuentes son conjuntos de caracteres con un tipo de fuente, un estilo y un tamaño similar. Con independencia de lo que cree mediante Flash Basic 8 o Flash Professional 8, probablemente utilice texto con al menos una o dos fuentes en las aplicaciones de Flash. Si crea animaciones y desconoce si los usuarios finales tendrán una fuente específica instalada en sus sistemas, deberá comprender los aspectos básicos de la incorporación de fuentes.

En las siguientes secciones se muestra cómo incorporar caracteres, fuentes completas, fuentes compartidas y otras técnicas de uso de fuentes en Flash 8.

Para más información sobre las fuentes, consulte las secciones siguientes:


- "Incorporación de caracteres" en la página 426
- "Incorporación de fuentes" en la página 427
- "Creación de conjuntos de caracteres personalizados" en la página 429
- "Utilización de métodos TextField con fuentes incorporadas" en la página 432
- "Fuentes compartidas" en la página 433


El siguiente ejemplo muestra cómo añadir y eliminar caracteres y conjuntos de caracteres incorporados en un documento de Flash.

### Para añadir y eliminar caracteres y conjuntos de caracteres incorporados:

1. Cree un nuevo documento de Flash y guárdelo como **embedding.fla**.
2. Cree un campo de texto dinámico en el escenario con la herramienta Texto.
3. Haga clic en el botón Incorporar para iniciar el cuadro de diálogo Incorporación de caracteres.
4. Seleccione el conjunto de caracteres específico que desee incorporar haciendo clic en él con el puntero del ratón.

   Para seleccionar varios conjuntos de caracteres, utilice la tecla Mayús o Control mientras selecciona los elementos con el puntero del ratón. Para seleccionar un bloque de conjuntos de caracteres, seleccione un conjunto con el puntero del ratón, mantenga presionada la tecla Mayús y haga clic en el nuevo conjunto de caracteres. Con la tecla Mayús se selecciona cada conjunto de caracteres de los dos seleccionados. Para seleccionar varios conjuntos de caracteres no secuenciales, mantenga presionada la tecla Control mientras selecciona los conjuntos de caracteres. También puede seleccionar rápidamente varios conjuntos de caracteres eligiendo un conjunto con el ratón y, manteniendo presionado el botón del ratón, arrastrándolo sobre varios conjuntos de caracteres.

5. Para eliminar un conjunto de caracteres específico que ha añadido anteriormente, mantenga presionada la tecla Control y anule la selección del conjunto de caracteres haciendo clic en él con el puntero del ratón.
6. Para eliminar todos los conjuntos de caracteres y cualquier carácter especificado en el campo de introducción de texto Incluir los siguientes caracteres, haga clic en No incorporar.

   No incorporar elimina cualquier carácter individual o conjunto de caracteres especificados anteriormente.

| ATENCIÓN | Al hacer clic en No incorporar en el cuadro de diálogo Incorporación de caracteres, se elimina sin solicitar la confirmación del usuario cualquier carácter o conjunto de caracteres incorporados especificados que se haya seleccionado anteriormente. |
|---|---|

## Incorporación de caracteres

Si trabaja con fuentes incorporadas y sabe exactamente qué caracteres precisa, puede reducir el tamaño de archivo incorporando sólo dichos caracteres en lugar de los contornos de fuentes no utilizadas adicionales. Para incorporar ciertos caracteres en un campo de texto y no un conjunto completo de caracteres, utilice el cuadro de diálogo Incorporación de caracteres para especificar los caracteres deseados.

### Para incorporar caracteres específicos y utilizarlos en un campo de texto:

1. Cree un nuevo documento de Flash y guárdelo como **charembed.fla**.
2. Con la herramienta Texto, cree un campo de texto en el escenario y defina el tipo de texto de dicho campo como Texto dinámico o Introducción de texto.
3. Con el campo de texto seleccionado aún en el escenario, haga clic en Incorporar en el inspector de propiedades para abrir el cuadro de diálogo Incorporación de caracteres.

   El cuadro de diálogo Incorporación de caracteres permite definir qué conjuntos de caracteres se incorporarán al documento de Flash (así como el número de glifos por conjunto de caracteres) y especificar caracteres concretos, e indica el número total de glifos que se incorporarán en este campo de texto.
4. Escriba la cadena **hello world** en el cuadro de texto Incluir los siguientes caracteres.

   El cuadro de diálogo indica que se incorporará un total de 8 glifos para este campo de texto. Aunque la cadena “hello world” contiene 11 caracteres, Flash sólo incorpora glifos exclusivos, de forma que las letras l y o se incorporan una vez en lugar de varias veces.
5. Haga clic en Aceptar para aplicar los cambios y volver al documento.
6. Con la herramienta Texto, cree un nuevo campo de texto en el escenario.

7. Defina el tipo de texto del campo como Texto dinámico en el inspector de propiedades.
8. Escriba la cadena **hello world** en el campo de texto en el escenario.
9. Haga clic en Incorporar en el inspector de propiedades para volver a abrir el cuadro de diálogo Incorporación de caracteres.
10. Haga clic en Relleno automático para rellenar automáticamente el cuadro de texto Incluir los siguientes caracteres.

    Verá la cadena "helo wrd". En lugar de tener que indicar a Flash qué caracteres desea incluir, el programa puede determinar automáticamente todos los caracteres exclusivos del campo de texto especificado.

| SUGERENCIA | Flash sólo puede determinar automáticamente qué caracteres debe incluir si el campo de texto contiene texto en el escenario. Si el campo de texto se ha rellenado con código ActionScript, es necesario especificar manualmente qué caracteres se deben incorporar al mismo. |
|---|---|

11. Haga clic en Aceptar.

## Incorporación de fuentes

Al incorporar fuentes, Flash almacena toda la información de fuentes en el archivo SWF, de manera que la fuente se muestre correctamente aunque no esté instalada en el equipo del usuario. Si utiliza una fuente en el archivo FLA que no está instalada en el equipo de un usuario y no incorpora la fuente en el archivo SWF, Flash Player selecciona automáticamente una fuente sustituta.

| NOTA | Sólo tendrá que incorporar una fuente si utiliza campos dinámicos o de introducción de texto. Si utiliza un campo de texto estático, no será necesario incorporar la fuente. |
|---|---|

### Para incorporar un símbolo de fuente:

1. Seleccione Ventana > Biblioteca para abrir la biblioteca del archivo FLA actual.

   Abra la biblioteca a la que desea añadir el símbolo de fuente.
2. Seleccione Nueva fuente del menú emergente de la biblioteca (en la esquina superior derecha del panel Biblioteca).
3. Escriba un nombre para el símbolo de fuente en el cuadro de texto Nombre del cuadro de diálogo Propiedades de Símbolo de Fuente.
4. Seleccione una fuente del menú Fuente o introduzca el nombre de una fuente en el cuadro de texto Nombre.

5. Seleccione Negrita, Cursiva o Texto de mapa de bits si desea aplicar un estilo a la fuente.
6. Introduzca el tamaño de la fuente que desee incorporar y haga clic en Aceptar para aplicar los cambios y regresar al documento.

   La fuente aparecerá en la biblioteca del documento actual.

Una vez que haya incorporado una fuente en la biblioteca, podrá utilizarla con un campo de texto del escenario.

### Para utilizar un símbolo de fuente incorporado en el documento de Flash:

1. Siga los pasos del procedimiento de "Incorporación de fuentes" en la página 427 para incorporar una fuente en la biblioteca.
2. Utilice la herramienta Texto para crear un campo de texto en el escenario.
3. Escriba texto en el campo de texto.
4. Seleccione el campo de texto y abra el inspector de propiedades.
   a. Establezca el campo de texto como de línea única.
   b. Seleccione el nombre de la fuente incorporada utilizando el menú desplegable Fuente.

   Las fuentes incorporadas presentan un asterisco (*) detrás de su nombre.
5. Haga clic en Incorporar en el inspector de propiedades para iniciar el cuadro de diálogo Incorporación de caracteres.

   El cuadro de diálogo Incorporación de caracteres le permite seleccionar caracteres individuales o conjuntos de caracteres que desea incorporar para el campo de texto seleccionado. Para especificar los caracteres que desea incorporar, escríbalos en el cuadro de texto del cuadro de diálogo o bien haga clic en Relleno automático para rellenar automáticamente el campo de texto con los caracteres exclusivos actualmente existentes en el mismo. Si no está seguro de los caracteres que va a necesitar (por ejemplo, en el caso de que el texto se cargue de un archivo externo o un servicio Web), puede seleccionar la incorporación de conjuntos de caracteres completos, como Mayúsculas [A..Z], Minúsculas [a..z], Numerales [0..9] o Puntuación [!@#%...], y conjuntos de caracteres para diferentes idiomas.

   | NOTA | Cada conjunto de caracteres que seleccione aumentará el tamaño final del archivo SWF, ya que Flash tendrá que almacenar toda la información de cada conjunto de caracteres que utilice. |
   |---|---|

6. Seleccione los caracteres individuales o conjuntos de caracteres que desee incorporar y luego haga clic en Aceptar para aplicar los cambios y regresar al documento.
7. Seleccione Control > Probar película para probar el documento de Flash en el entorno de edición.

La fuente incorporada se muestra en el campo de texto en el escenario. Para probar correctamente que se ha incorporado la fuente, puede que tenga que probarlo en un equipo en el que no esté instalada la fuente incorporada.

O bien puede establecer las propiedades `TextField._alpha` o `TextField._rotation` para el campo de texto que tiene fuentes incorporadas, ya que estas propiedades sólo funcionan con fuentes incorporadas (consulte los siguientes pasos).

8. Cierre el archivo SWF y regrese al entorno de edición.
9. Seleccione el campo de texto en el escenario y abra el inspector de propiedades.
   a. Establezca el tipo de texto como Texto dinámico.
   b. Escriba **font_txt** en el cuadro de texto Nombre de instancia.
10. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
font_txt._rotation = 45;
```

11. Seleccione Control > Probar película para ver los cambios en el entorno de edición.

    La fuente incorporada gira 45 grados en el sentido de las agujas del reloj y seguirá viendo el texto porque está incorporado en el archivo SWF.

**ATENCIÓN** Si no incorpora una fuente en el documento de Flash y Flash Player elige automáticamente una fuente de sustitución en el equipo del usuario, la propiedad TextField.font devuelve la fuente original utilizada en el archivo FLA, no el nombre de la fuente sustituta.

**NOTA** Si utiliza fuentes incorporadas con distintos estilos en los campos de texto, debe incorporar el estilo que desee emplear. Por ejemplo, si utiliza una fuente incorporada denominada Times y desea que una palabra aparezca en cursiva, debe asegurarse de incorporar tanto los contornos de carácter normal como en cursiva. De lo contrario, el texto no aparecerá en el campo.

## Creación de conjuntos de caracteres personalizados

Además de utilizar el conjunto de caracteres predeterminado de Flash, también puede crear conjuntos propios y añadirlos al cuadro de diálogo Incorporación de caracteres. Por ejemplo, puede que necesite que algunos campos incluyan Latín extendido para admitir distintos caracteres acentuados. Pero quizás no necesite numerales y puntuación o sólo precise caracteres en mayúscula. En lugar de incorporar conjuntos completos de caracteres, puede crear un conjunto personalizado que contenga sólo aquellos caracteres que necesita. De esta forma puede mantener el menor tamaño de archivo SWF posible, ya que no almacena ninguna información de fuente adicional para los caracteres que no precisa.

Para crear un conjunto de caracteres personalizado, debe editar el archivo UnicodeTable.xml, que se encuentra en el directorio C:\Archivos de programa\Macromedia\Flash 8\<*idioma*>\First Run\FontEmbedding\. Este archivo define el conjunto de caracteres predeterminado y los rangos de caracteres y los caracteres que contienen.

Antes de crear un conjunto de caracteres personalizado, debe comprender la estructura XML necesaria. Los siguientes nodos XML definen el conjunto de caracteres Mayúsculas [A..Z]:

```
<glyphRange name="Mayúsculas [A..Z] " id="1" >
  <range min="0x0020" max ="0x0020" />
  <range min="0x0041" max ="0x005A" />
</glyphRange>
```

Observe que el nodo `glyphRange` incluye `name`, `Mayúsculas [A..Z]` e `id`. Un nodo `glyphRange` puede incluir tantos nodos secundarios de *rango* como sea necesario. Un rango puede ser un único carácter, como `0x0020` (el carácter de espacio), que aparece en el fragmento anterior, o un rango de caracteres, como el segundo nodo secundario. Para incorporar sólo un carácter, defina los valores `min` y `max` en el mismo valor de carácter Unicode.

Otro ejemplo de nodo `glyphRange` XML es el nodo `Numerales [0..9]`:

```
<glyphRange name="Numerales [0..9] " id="3" >
  <range min="0x0030" max ="0x0039" />
  <range min="0x002E" max ="0x002E" />
</glyphRange>
```

Este rango de caracteres incluye los valores Unicode 0x0030 (cero) a 0x0039 (9), así como 0x002E (.).

Antes de crear un conjunto de caracteres predeterminado, debe conocer los caracteres y sus correspondientes valores Unicode. El mejor sitio para buscar valores Unicode es el sitio Web de estándares Unicode, www.unicode.org, que contiene una tabla de códigos de caracteres Unicode para docenas de idiomas.

**ATENCIÓN** Para añadir conjuntos de caracteres personalizados debe editar un archivo XML en la carpeta de instalación de Flash. Antes de ello, haga una copia de seguridad por si necesitara recuperar la tabla Unicode original.

**ATENCIÓN** Macromedia recomienda no modificar los conjuntos de caracteres existentes instalados con Flash y, en su lugar, crear conjuntos personalizados que incluyan la puntuación y los caracteres necesarios.

### Para crear y utilizar un conjunto de caracteres personalizado:

1. Abra el documento UnicodeTable.xml, que se encuentra en *<directorio de instalación de Flash>**<idioma>*\First Run\FontEmbedding\, con un editor de texto o XML como Bloc de notas o TextEdit.

   **NOTA** No olvide realizar una copia de seguridad de este documento, por si necesitara recuperar el archivo original instalado con Flash.

2. Desplácese al final del documento XML y añada el siguiente código XML directamente antes del nodo de cierre `</fontEmbeddingTable>`:

```
<glyphRange name="Mayúsculas y numerales [A..Z,0..9] " id="100" >
  <range min="0x0020" max ="0x0020" />
  <range min="0x002E" max ="0x002E" />
  <range min="0x0030" max ="0x0039" />
  <range min="0x0041" max ="0x005A" />
</glyphRange>
```

3. Guarde los cambios en UnicodeTable.xml.

   Si tiene Flash abierto, reinicie la aplicación antes de utilizar el nuevo conjunto de caracteres.

4. Abra o reinicie Flash y cree un documento de Flash nuevo.
5. Añada una nueva instancia de TextField en el escenario con la herramienta Texto.
6. Defina el tipo de texto de TextField como Texto dinámico en el inspector de propiedades y haga clic en Incorporar para abrir el cuadro de diálogo Incorporación de caracteres.
7. Desplácese hasta el final del cuadro de diálogo Incorporación de caracteres y seleccione el nuevo conjunto de caracteres, Mayúsculas y numerales [A..Z,0..9] (38 glifos).
8. Elija otros conjuntos de caracteres deseados y haga clic en Aceptar.

   Si selecciona el conjunto de caracteres personalizado, Mayúsculas y numerales [A..Z,0..9], así como los predeterminados Mayúsculas [A..Z] o Numerales [0..9], observe que el número de glifos incorporados no cambia. Esto se debe a que los caracteres en mayúscula se incluyen en el conjunto personalizado y Flash no incluye caracteres duplicados, lo que mantiene el tamaño de archivo lo más reducido posible. Si selecciona el conjunto de caracteres Puntuación, que incluye 52 glifos, junto con el conjunto de caracteres personalizado, que incluye 38, Flash almacena información sólo de 88 glifos en lugar de 90. Esto ocurre porque dos caracteres que se solapan, el espacio y el punto, ya están incluidos en el conjunto personalizado.

   **SUGERENCIA** La posición del conjunto de caracteres en el cuadro de diálogo Incorporación de caracteres la determina su posición en el documento XML. Puede cambiar el orden de los conjuntos de caracteres, incluidos los personalizados, moviendo los paquetes ‹glyphRange› en el archivo XML.

# Utilización de métodos TextField con fuentes incorporadas

Los métodos de la clase TextField proporcionan funcionalidad de gran utilidad para las aplicaciones. Por ejemplo, puede controlar el grosor de un campo de texto empleando código ActionScript, como se muestra en el siguiente ejemplo.

### Para establecer el grosor de un campo de texto utilizando código ActionScript:

1. Cree un nuevo documento de Flash y guárdelo como **textfieldThickness.fla**.
2. Abra el panel Biblioteca y seleccione Nueva fuente en el menú emergente (situado en la esquina superior derecha del panel Biblioteca).

   Se abrirá el cuadro de diálogo Propiedades de Símbolo de Fuente. Este cuadro de diálogo le permite seleccionar una fuente para incorporarla en el archivo SWF (incluido un estilo y un tamaño de fuente). También puede asignar un nombre de fuente que aparece en la biblioteca del documento y el menú desplegable de fuentes del inspector de propiedades (si tiene un campo de texto seleccionado en el escenario).

   a. Seleccione la fuente Times New Roman del menú desplegable Fuente.
   b. Asegúrese de que no estén seleccionadas las opciones Negrita y Cursiva.
   c. Establezca el tamaño en 30 píxeles.
   d. Introduzca el nombre de fuente **Times (embedded)**
   e. Haga clic en Aceptar.
3. En la biblioteca, haga clic con el botón derecho del ratón en el símbolo de la fuente y seleccione Vinculación en el menú contextual.

   Flash abrirá el cuadro de diálogo Propiedades de vinculación.
4. Seleccione las opciones Exportar para ActionScript y Exportar en primer fotograma y haga clic en Aceptar.
5. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
// 1
this.createTextField("thickness_txt", 10, 0, 0, Stage.width, 22);
this.createTextField("lorem_txt", 20, 0, 20, Stage.width, 0);
lorem_txt.autoSize = "left";
lorem_txt.embedFonts = true;
lorem_txt.antiAliasType = "advanced";
lorem_txt.text = "Lorem ipsum dolor sit amet, consectetur adipisicing
  elit, sed do eiusmod tempor incididunt ut labore et dolore magna
  aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco
  laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor
  in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
  pariatur. Excepteur sint occaecat cupidatat non proident, sunt in
  culpa qui officia deserunt mollit anim id est laborum.";
lorem_txt.wordWrap = true;
```

```
// 2
var style_fmt:TextFormat = new TextFormat();
style_fmt.font = "Times (embedded)";
style_fmt.size = 30;
lorem_txt.setTextFormat(style_fmt);

// 3
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  // Los valores de TextField.thickness pueden ir de -200 a +200.
  lorem_txt.thickness = Math.round(_xmouse * (400 / Stage.width) - 200);
  thickness_txt.text = "TextField.thickness = " + lorem_txt.thickness;
};
Mouse.addListener(mouseListener);
```

El primer bloque de código crea dos campos de texto, `thickness_txt` y `lorem_txt` y los sitúa en el escenario. El campo de texto `lorem_txt` define su propiedad `embedFonts` en el valor `true` y rellena el campo de texto con un bloque de texto.

El segundo bloque de código define un formato de texto con el tipo de letra Times New Roman, establece el tamaño de fuente en 30 píxeles y aplica el formato de texto al campo de texto `lorem_txt`.

El tercer y último bloque de código define y asigna un detector de ratón para el evento `onMouseMove`. Cuando el puntero del ratón se mueve horizontalmente por el escenario, la propiedad `TextField.thickness` cambia entre -200 y +200 en función del valor actual de `_xmouse`.

6. Guarde los cambios en el archivo FLA.
7. Seleccione Control > Probar película para probar el documento de Flash.

   Cuando se mueve el puntero del ratón a la mitad izquierda del escenario, el grosor de la fuente disminuye. Cuando se mueve el puntero del ratón a la mitad derecha del escenario, el grosor de la fuente aumenta.

## Fuentes compartidas

Para utilizar una fuente como un elemento de biblioteca compartida, puede crear un símbolo de fuente en el panel Biblioteca y luego asignar los siguientes atributos al símbolo de fuente:

- Una cadena identificadora
- Una URL en la que se encuentra el documento que contiene el símbolo de fuente

De esta forma, puede vincular la fuente y utilizarla en una aplicación de Flash sin que la fuente esté almacenada dentro del archivo FLA.

# Representación de fuentes y texto suavizado

La representación de fuentes en Flash controla la forma en que se muestra el texto en un archivo SWF, es decir, cómo se representa (o *dibuja*) durante la ejecución. La tecnología de representación avanzada de fuentes utilizada en Flash Player 8 se denomina FlashType. FlashType permite hacer que el texto resulte legible y claro con tamaños de fuente de pequeños a normales, por ejemplo, cuando se aplica el suavizado avanzado a los campos de texto. Esta tecnología se describe con más detalle más adelante en esta misma sección.

El suavizado de texto permite suavizar el texto para que los bordes de los caracteres mostrados en pantalla no aparezcan dentados, lo que puede resultar de gran ayuda si se desea mostrar texto utilizando tamaños de texto pequeños. La opción de suavizado permite obtener un texto más legible alineando los contornos del texto a los límites de los píxeles y resulta especialmente efectivo para representar con mayor claridad las fuentes de tamaños más pequeños. El suavizado se aplica a cada campo de texto, no a cada uno de los caracteres.

Esta opción está permitida para texto estático, texto dinámico e introducción de texto si el usuario final dispone de Flash Player 7 o posterior. Si, por el contrario, el usuario sólo dispone de una versión anterior de Flash Player, la opción únicamente estará disponible para texto estático. Las opciones de suavizado avanzadas están disponibles para Flash Player 8.

Flash Basic 8 y Flash Professional 8 incluyen una tecnología significativamente mejorada de conversión y representación de fuentes, denominada FlashType, para trabajar con fuentes suavizadas. Flash 9 incluye cinco métodos de representación de fuentes, que sólo están disponibles cuando se publican archivos SWF para Flash Player 8. Si publica archivos para su uso con Flash Player 7 o versiones anteriores, sólo podrá utilizar la función Suavizado para animación con los campos de texto.

FlashType es una tecnología de representación de fuentes de gran calidad que se puede activar mediante la herramienta de edición o el código ActionScript de Flash 8. FlashType permite representar tipos de letra con alta calidad en tamaños reducidos con mayor control. Puede aplicar FlashType a la representación de fuentes incorporadas para campos de texto estáticos, dinámicos y de introducción de texto. Las prestaciones mejoradas hacen que el texto incorporado se muestre con el mismo nivel de calidad que el texto de dispositivo y que las fuentes se muestren iguales en distintas plataformas.

Los métodos de representación de fuentes disponibles en Flash Player 8 son fuentes de dispositivo, texto de mapa de bits (sin suavizado), suavizado para animación, suavizado para legibilidad o suavizado personalizado, que permite definir un valor personalizado de grosor y nitidez. Para más información sobre estas opciones, consulte "Opciones de representación de fuentes en Flash" en la página 436.

NOTA

Al abrir archivos FLA en Flash 8, el texto no adopta automáticamente la opción Suavizado para legibilidad, sino que es preciso seleccionar individualmente los campos de texto y cambiar manualmente la definición del suavizado para aprovechar la tecnología de representación de FlashType.

Las funciones de suavizado avanzado y personalizado son compatibles con lo siguiente:

- Texto con escala y girado
- Todas las fuentes (normal, negrita o cursiva) con un tamaño de hasta 255 pt
- Exportación de archivos a la mayoría de formatos (como, por ejemplo, archivos JPEG o GIF)

Las funciones de suavizado avanzado y personalizado no son compatibles con lo siguiente:

- Flash Player 7 o anteriores
- Texto sesgado o volteado
- Impresión
- Exportación de archivos al formato de archivo PNG

NOTA

Si hay animación de texto, el reproductor desactiva el suavizado avanzado para mejorar el aspecto del texto mientras éste se está moviendo. Una vez finalizada la animación, vuelve a activarse el suavizado.

Un archivo de ejemplo que se encuentra en el disco duro muestra cómo aplicar y manipular el texto suavizado en una aplicación. Se utiliza la tecnología de representación de FlashType para crear texto pequeño que aún resulta muy legible. En el archivo de ejemplo también se muestra cómo los campos de texto pueden desplazarse rápida y gradualmente cuando se utiliza la propiedad `cacheAsBitmap`.

Puede encontrar el archivo de origen de muestra, flashtype.fla, en la carpeta Samples del disco duro.

En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\FlashType.

En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/FlashType.

## Opciones de representación de fuentes en Flash

Hay cinco opciones de representación de fuentes disponibles en Flash 8. Para elegir una opción, seleccione el campo de texto y abra el inspector de propiedades. Seleccione una opción del menú emergente Método de representación de fuentes.

**Utilizar fuentes del dispositivo** Produce un tamaño de archivo SWF más pequeño. La representación se realiza utilizando fuentes que están actualmente instaladas en el equipo del usuario final.

**Texto de mapa de bits (sin suavizado)** Genera un texto con bordes afilados, sin suavizado. Esta opción produce archivos SWF de un tamaño mayor, ya que los contornos de fuentes se incluyen en el archivo SWF.

**Suavizado para animación** Produce texto suavizado que permite una animación armonizada. La animación del texto también es más rápida en algunas situaciones, ya que la alineación y el suavizado no se aplican mientras tiene lugar la animación del texto. No apreciará ninguna mejora en el rendimiento si utiliza fuentes grandes con muchas letras o fuentes con escala. Esta opción produce archivos SWF de un tamaño mayor, ya que los contornos de fuentes se incluyen en el archivo SWF.

**Suavizado para legibilidad** Para esta opción, se utiliza un motor de suavizado avanzado. Esta opción ofrece el texto de más alta calidad y más legible. El tamaño de archivo SWF también es el más grande, ya que incluye los contornos de fuentes, además de información especial de suavizado.

**Suavizado personalizado** Es la misma que Suavizado para legibilidad, con la diferencia de que puede manipular visualmente los parámetros de suavizado para lograr un aspecto concreto. Esta opción es útil para lograr el mejor aspecto posible con fuentes nuevas o poco habituales.

Para ver un ejemplo de la forma de utilizar el suavizado con ActionScript, consulte "Definición de suavizado con ActionScript" en la página 437.

## Modulación continua de trazo

La tecnología de representación de fuentes de FlashType aprovecha las propiedades inherentes de los campos de distancia para proporcionar una modulación CSM (Modulación continua de trazo, Continuous Stroke Modulation); por ejemplo, la modulación continua del grosor del trazo y la nitidez del borde del texto. CSM emplea dos parámetros de representación para controlar la asignación de distancias ADF (Campos de distancia con muestreo adaptable, Adaptively Sampled Distance Field) a valores de densidad de glifo. Los valores óptimos para estos parámetros son muy subjetivos; pueden depender de las preferencias del usuario, las condiciones de iluminación, las propiedades de la pantalla, las fuentes, los colores de fondo y en primer plano y el tamaño en puntos. La función que asigna distancias ADF a valores de densidad tiene un corte externo, por debajo del cual los valores se establecen en cero, y un corte interno, por encima del cual la densidad se define en un valor máximo (como 255).

## Definición de suavizado con ActionScript

Flash 8 ofrece dos tipos de suavizado: el normal y el avanzado. El suavizado avanzado sólo está disponible en Flash Player 8 y versiones posteriores y sólo se puede emplear si se incorpora la fuente a la biblioteca y la propiedad `embedFonts` del campo de texto se ha establecido en `true`. En Flash Player 8, la configuración predeterminada para los campos de texto creados con ActionScript es `normal`.

Para establecer los valores de la propiedad `TextField.antiAliasType`, utilice los siguientes valores de cadena:

**normal** Aplica el suavizado de texto regular. Equivale al tipo de suavizado que utilizaba Flash Player en la versión 7 y anteriores.

**advanced** Aplica suavizado avanzado para mejorar la legibilidad del texto, disponible desde Flash Player 8. El suavizado avanzado permite reproducir las fuentes con una calidad muy buena en tamaños pequeños. Ofrece mejores resultados con aplicaciones que tienen gran cantidad de texto pequeño.

| SUGERENCIA | Macromedia no recomienda utilizar el suavizado avanzado con fuentes mayores de 48 puntos. |
|---|---|

Para utilizar ActionScript para establecer el texto suavizado, consulte el siguiente ejemplo.

### Para utilizar suavizado avanzado:

1. Cree un nuevo documento de Flash y guárdelo como **antialiastype.fla**.
2. Cree dos clips de película en el escenario y asígneles los nombres de instancia **normal_mc** y **advanced_mc**.

   Utilizará estos dos clips para alternar entre los dos tipos de suavizado: el normal y el avanzado.
3. Abra el panel Biblioteca y seleccione Nueva fuente en el menú emergente, situado en la esquina superior derecha del panel Biblioteca.

   Se abrirá el cuadro de diálogo Propiedades de Símbolo de Fuente, en el que puede seleccionar una fuente para incorporarla en el archivo SWF (incluido un estilo y un tamaño de fuente). También puede asignar un nombre de fuente que aparece en la biblioteca del documento y el menú desplegable Fuente del inspector de propiedades (si tiene un campo de texto seleccionado en el escenario).
   a. Seleccione la fuente Arial del menú desplegable Fuente.
   b. Asegúrese de no seleccionar las opciones de negrita ni de cursiva.
   c. Establezca el tamaño en 10 píxeles.
   d. Introduzca el nombre de fuente **Arial-10 (embedded)**.
   e. Haga clic en Aceptar.
4. En la biblioteca, haga clic con el botón derecho del ratón en el símbolo de la fuente y seleccione Vinculación en el menú contextual.

   Aparecerá el cuadro de diálogo Propiedades de vinculación.
5. Seleccione las opciones Exportar para ActionScript y Exportar en primer fotograma, introduzca el identificador de vinculación **Arial-10** y haga clic en Aceptar.
6. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
var text_fmt:TextFormat = new TextFormat();
text_fmt.font = "Arial-10";
text_fmt.size = 10;

this.createTextField("my_txt", 10, 20, 20, 320, 240);
my_txt.autoSize = "left";
my_txt.embedFonts = true;
my_txt.selectable = false;
my_txt.setNewTextFormat(text_fmt);
my_txt.multiline = true;
my_txt.wordWrap = true;
```

```
var lorem_lv:LoadVars = new LoadVars();
lorem_lv.onData = function(src:String) {
  if (src != undefined) {
    my_txt.text = src;
  } else {
    my_txt.text = "unable to load text file.";
  }
};
lorem_lv.load("http://www.helpexamples.com/flash/lorem.txt");

normal_mc.onRelease = function() {
  my_txt.antiAliasType = "normal";
};
advanced_mc.onRelease = function() {
  my_txt.antiAliasType = "advanced";
};
```

El código anterior se divide en cuatro áreas principales. El primer bloque de código crea un nuevo objeto TextFormat, que especifica la fuente y el tamaño de fuente que se emplearán para un campo de texto que se va a crear. La fuente especificada, Arial-10, es el identificador de vinculación del símbolo de fuente incorporado en el paso anterior.

El segundo bloque de código crea un nuevo campo de texto con el nombre de instancia `my_txt`. Para que la fuente se pueda incorporar correctamente, debe establecer `embedFonts` en `true` para la instancia del campo de texto. El código también establece el formato del texto del nuevo campo en el objeto TextFormat creado anteriormente.

El tercer bloque de código define una instancia LoadVars que rellena el campo de texto en el escenario con el contenido de un archivo de texto externo. Una vez se ha cargado completamente el documento (pero no se ha analizado), todo su contenido se copia en la propiedad `my_txt.text` para que se muestre en el escenario.

El cuarto y último bloque de código define controladores de eventos `onRelease` para los clips de película `normal_mc` y `advanced_mc`. Cuando el usuario hace clic y libera cualquiera de estas opciones, cambia el tipo de suavizado del campo de texto en el escenario.

7. Guarde los cambios en el archivo FLA.
8. Seleccione Control > Probar película para probar el documento de Flash.
9. Haga clic en el clip de película `advanced_mc` en el escenario.

Al hacerlo, el clip de película cambia el tipo de suavizado de normal (el predeterminado) a avanzado. Cuando se utilizan campos de texto con tamaños de fuente menores, el establecimiento del suavizado en avanzado puede mejorar enormemente la legibilidad del texto.

| SUGERENCIA | El suavizado avanzado permite representar los tipos de fuentes en tamaño pequeño con una calidad muy elevada. Ofrece mejores resultados con aplicaciones que tienen gran cantidad de texto pequeño. Macromedia no recomienda utilizar el suavizado avanzado con fuentes mayores de 48 puntos. |
|---|---|

Para más información sobre la aplicación de formato suavizado al texto, consulte "Utilización de un tipo de ajuste de cuadrícula" en la página 446 y "Aplicación de formato suavizado al texto" en la página 444.

Un archivo de ejemplo que se encuentra en el disco duro muestra cómo aplicar y manipular el texto suavizado en una aplicación. Se utiliza la tecnología de representación de FlashType para crear texto pequeño que aún resulta muy legible. En el archivo de ejemplo también se muestra cómo los campos de texto pueden desplazarse rápida y gradualmente cuando se utiliza la propiedad `cacheAsBitmap`.

Puede encontrar el archivo de origen de muestra, flashtype.fla, en la carpeta Samples del disco duro.

En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\FlashType.

En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/FlashType.

## Definición de tablas para fuentes

Si crea fuentes para utilizarlas en archivos SWF o distribuirlas a desarrolladores de Flash, puede que necesite definir tablas para fuentes a fin de controlar la forma en la que se representan en el escenario.

El suavizado avanzado utiliza campos de distancia con muestreo adaptable (ADF) para representar los contornos que delimitan un glifo (un carácter). Flash emplea dos valores:

- Un valor de corte externo, por debajo del cual las densidades se establecen en cero.
- Un valor de corte interno, por encima del cual las densidades se definen en un valor de densidad máximo, como 255.

Entre estos dos valores de corte, la función de asignación es una curva lineal que va desde cero en el corte externo hasta la densidad máxima en el corte interno.

El ajuste de los valores de corte externo e interno afecta al grosor del trazo y a la nitidez del borde. El espaciado entre estos dos parámetros es comparable al doble del radio del filtro de los métodos de suavizado clásicos; un espaciado menor resulta en un borde más nítido, mientras que un espaciado mayor ofrece bordes más suaves y filtrados. Cuando el espaciado es cero, la imagen de densidad resultante es un mapa de bits en dos niveles. Si el espaciado es muy reducido, la imagen de densidad resultante tiene bordes diluidos tipo acuarela.

Por lo general, el usuario prefiere los bordes nítidos muy contrastados en tamaños de punto pequeños, y bordes más suaves para texto animado y tamaños de punto grandes.

El corte externo suele tener un valor negativo, el corte interno un valor positivo y su punto intermedio un valor próximo a cero. El ajuste de estos parámetros para desplazar el punto intermedio hacia un valor infinito negativo aumentará el grosor del trazo; el desplazamiento del punto intermedio hacia un valor infinito positivo reducirá el grosor del trazo.

| NOTA | El corte externo siempre debe tener un valor menor o igual que el corte interno. |
|---|---|

Flash Player incluye valores de suavizado avanzado para diez fuentes básicas y, de ellas, solamente para los tamaños de fuente entre 6 y 20. Para estas últimas, se utiliza el valor 6 para todos los tamaños por debajo de 6, y 20 para todos los tamaños por encima de 20. Las demás fuentes se asignan a los datos de fuentes suministrados. El método `setAdvancedAntialiasingTable()` permite definir datos de suavizado personalizados para otras fuentes y tamaños de fuente, o bien sustituir los valores predeterminados de las fuentes proporcionadas. Para más información sobre la creación de una tabla de suavizado, consulte el siguiente ejemplo:

### Para crear una tabla de suavizado avanzado para una fuente incorporada:

1. Cree un nuevo documento de Flash y guárdelo como **advancedaatable.fla**.
2. Seleccione Nueva fuente en el menú emergente Biblioteca.
3. Seleccione Arial en el menú emergente Fuente y establezca el tamaño de la fuente en **32** puntos.
4. Seleccione las opciones de negrita y cursiva.
5. Escriba el nombre de fuente **Arial (embedded)** en el cuadro de texto Nombre y haga clic en Aceptar.
6. Haga clic con el botón derecho del ratón (Windows) o con la tecla Control presionada (Macintosh) en el símbolo de fuente de la biblioteca y seleccione Vinculación.

7. En el cuadro de diálogo Propiedades de vinculación:
   a. Escriba **Arial-embedded** en el cuadro de texto Identificador.
   b. Active las opciones Exportar para ActionScript y Exportar en primer fotograma.
   c. Haga clic en Aceptar.
8. Seleccione el fotograma 1 de la línea de tiempo principal y añada el siguiente código ActionScript en el panel Acciones:

```
import flash.text.TextRenderer;
var arialTable:Array = new Array();
arialTable.push({fontSize:16.0, insideCutoff:0.516,
  outsideCutoff:0.416});
arialTable.push({fontSize:32,0, insideCutoff:2,8, outsideCutoff:-2,8});
TextRenderer.setAdvancedAntialiasingTable("Arial", "bolditalic", "dark",
  arialTable);

var my_fmt:TextFormat = new TextFormat();
my_fmt.align = "justify";
my_fmt.font = "Arial-embedded";
my_fmt.size = 32;

this.createTextField("my_txt", 999, 10, 10, Stage.width-20,
  Stage.height-20);
my_txt.antiAliasType = "advanced";
my_txt.embedFonts = true;
my_txt.multiline = true;
my_txt.setNewTextFormat(my_fmt);
my_txt.sharpness = 0;
my_txt.thickness = 0;
my_txt.wordWrap = true;

var lorem_lv:LoadVars = new LoadVars();
lorem_lv.onData = function(src:String):Void {
  if (src != undefined) {
    my_txt.text = src + "\n\n" + src;
  } else {
    trace("error downloading text file");
  }
};
lorem_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

El código anterior se divide en cuatro secciones. La primera sección del código importa la clase TextRenderer y define una nueva tabla de suavizado para dos tamaños diferentes de la fuente Arial. La segunda sección define un nuevo objeto TextFormat, que se utiliza para aplicar formato al campo de texto (que se crea en la siguiente sección del código). La siguiente sección del código crea un nuevo campo de texto con el nombre de instancia `my_txt`, activa el suavizado avanzado, aplica el objeto de formato de texto (creado anteriormente) y habilita el texto multilínea y el ajuste de texto. El último bloque de código define un objeto LoadVars que se emplea para cargar texto de un archivo de texto externo y rellenar el campo de texto en el escenario.

9. Seleccione Control > Probar película para probar el documento de Flash.

   Después de que el texto se ha cargado desde el servidor remoto, Flash muestra algún texto en el campo de texto y se puede comprobar que las propiedades de la tabla de suavizado se han aplicado al mismo. La fuente incorporada en el escenario debe aparecer con un ligero efecto de desenfoque debido a los valores `insideCutoff` y `outsideCutoff` existentes.

# Diseño y formato de texto

Puede controlar el diseño y el formato del texto mediante código ActionScript. La clase TextFormat proporciona un gran control sobre cómo se muestra el texto durante la ejecución, además de otros tipos de formato como hojas de estilos (consulte "Aplicación de formato al texto con hojas de estilos en cascada" en la página 451) y texto HTML (consulte "Utilización de texto en formato HTML" en la página 465).

También puede controlar la forma en la que los caracteres se ajustan a la cuadrícula mediante código ActionScript cuando se utiliza texto suavizado en un archivo SWF. Esto permite controlar la apariencia de los caracteres durante la ejecución. Para ver un ejemplo de cómo utilizar un tipo de ajuste de cuadrícula en las aplicaciones, consulte "Utilización de un tipo de ajuste de cuadrícula" en la página 446.

Para obtener información general sobre los campos de texto, consulte "Campos de texto" en la página 409. Para obtener información sobre la aplicación de formato al texto, consulte "Aplicación de formato suavizado al texto" en la página 444. Para más información sobre la clase TextFormat, consulte "Utilización de la clase TextFormat" en la página 448 y `%{TextFormat}%` en *Referencia del lenguaje ActionScript 2.0*.

Para más información sobre el diseño y el formato del texto mediante la utilización de la clase TextFormat, consulte las siguientes secciones:

- "Aplicación de formato suavizado al texto" en la página 444
- "Utilización de un tipo de ajuste de cuadrícula" en la página 446
- "Utilización de la clase TextFormat" en la página 448
- "Propiedades predeterminadas de los nuevos campos de texto" en la página 450

## Aplicación de formato suavizado al texto

Flash 8 introduce dos nuevas propiedades que se pueden emplear cuando se aplica formato a los campos de texto con el suavizado avanzado activado: `sharpness` y `thickness`. La primera de ellas, la nitidez, hace referencia a la cantidad de suavizado que se aplica a una instancia de campo de texto. Un valor de nitidez alto hace que el borde de la fuente incorporada aparezca dentado y afilado. Un valor menor hace que la apariencia de la fuente sea más suave y con mayor desenfoque. Establecer un valor de nitidez de la fuente es similar a activar el formato de negrita para un campo de texto. Cuanto mayor sea la nitidez, más en negrita aparecerá la fuente.

El ejemplo siguiente carga dinámicamente un archivo de texto y muestra texto en el escenario. Si se mueve el puntero del ratón por el eje *x*, se establece la nitidez entre -400 y 400. Si se mueve por el eje *y*, se establece entre -200 y 200.

### Para modificar la nitidez y el grosor de un campo de texto:

1. Cree un nuevo documento de Flash y guárdelo como **sharpness.fla**.
2. Seleccione Nueva Fuente del menú emergente en la esquina superior derecha del panel Biblioteca.
3. Seleccione Arial en el menú desplegable Fuente y establezca el tamaño de la fuente en 24 puntos.
4. Escriba el nombre de fuente **Arial-24 (embedded)** en el cuadro de texto Nombre y haga clic en Aceptar.
5. Haga clic con el botón derecho del ratón en el símbolo de fuente en la biblioteca y seleccione Vinculación para abrir el cuadro de diálogo Propiedades de vinculación.
6. Defina el identificador de vinculación en Arial-24, seleccione las casillas de verificación Exportar para ActionScript y Exportar en primer fotograma y haga clic en Aceptar.
7. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.size = 24;
my_fmt.font = "Arial-24";

this.createTextField("lorem_txt", 10, 0, 20, Stage.width, (Stage.height
  - 20));
lorem_txt.setNewTextFormat(my_fmt);
lorem_txt.text = "loading...";
lorem_txt.wordWrap = true;
lorem_txt.autoSize = "left";
lorem_txt.embedFonts = true;
lorem_txt.antiAliasType = "advanced";

this.createTextField("debug_txt", 100, 0, 0, Stage.width, 20);
debug_txt.autoSize = "left";
debug_txt.background = 0xFFFFFF;
```

```
var lorem_lv:LoadVars = new LoadVars();
lorem_lv.onData = function(src:String) {
  lorem_txt.text = src;
}
lorem_lv.load("http://www.helpexamples.com/flash/lorem.txt");

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  lorem_txt.sharpness = (_xmouse * (800 / Stage.width)) - 400;
  lorem_txt.thickness = (_ymouse * (400 / Stage.height)) - 200;
  debug_txt.text = "sharpness=" + Math.round(lorem_txt.sharpness) +
    ", thickness=" + Math.round(lorem_txt.thickness);
};
Mouse.addListener(mouseListener);
```

Este código ActionScript se puede dividir en cinco secciones principales. La primera sección del código define una nueva instancia de TextFormat que se aplicará a un campo de texto creado dinámicamente. Las dos secciones siguientes crean dos campos de texto nuevos en el escenario. El primero, `lorem_txt`, aplica el objeto de formato de texto personalizado que se creó anteriormente, activa las fuentes incorporadas y establece la propiedad `antiAliasType` en `true`. El segundo, `debug_txt`, muestra los valores de nitidez y grosor actuales del campo de texto `lorem_txt`. La cuarta sección del código crea un objeto LoadVars, que es el responsable de cargar el archivo de texto externo y de rellenar el campo de texto `lorem_txt`. La quinta y última sección del código define un detector de ratón al que se llama siempre que se mueve el puntero sobre el escenario. Los valores actuales de `sharpness` y `thickness` se calculan en función de la posición del puntero del ratón en el escenario. Las propiedades `sharpness` y `thickness` se establecen para el campo de texto `lorem_txt` y sus valores actuales se muestran en el campo de texto `debug_txt`.

8. Seleccione Control > Probar película para probar el documento.

   Mueva el puntero del ratón por el eje *x* para cambiar la nitidez del campo de texto. Muévalo de izquierda a derecha para aumentar la nitidez y hacer que aparezca más dentado. Mueva el puntero del ratón por el eje *y* para cambiar el grosor del campo de texto.

Para más información sobre el uso del texto suavizado en un archivo SWF, consulte "Definición de suavizado con ActionScript" en la página 437, "Opciones de representación de fuentes en Flash" en la página 436 y "Utilización de un tipo de ajuste de cuadrícula" en la página 446.

Un archivo de ejemplo que se encuentra en el disco duro muestra cómo aplicar y manipular el texto suavizado en una aplicación. Se utiliza la tecnología de representación de FlashType para crear texto pequeño que aún resulta muy legible. En el archivo de ejemplo también se muestra cómo los campos de texto pueden desplazarse rápida y gradualmente cuando se utiliza la propiedad `cacheAsBitmap`.

Puede encontrar el archivo de origen de muestra, flashtype.fla, en la carpeta Samples del disco duro.

En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\FlashType.

En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/FlashType.

## Utilización de un tipo de ajuste de cuadrícula

Cuando se utiliza el suavizado avanzado en un campo de texto, se ofrecen tres tipos de ajuste de cuadrícula:

**none** Especifica que no hay ajuste de cuadrícula. Las líneas horizontales y verticales de los glifos no se ajustan a la cuadrícula de píxeles. Esta suele ser una buena opción para animación o para tamaños de fuente grandes.

**pixel** Especifica que las líneas horizontales y verticales intensas se ajustan a la cuadrícula de píxeles. Esta opción funciona sólo en campos de texto alineado a la izquierda. En general, es la opción que ofrece la mayor legibilidad del texto con alineación a la izquierda.

**subpixel** Especifica que las líneas horizontales y verticales intensas se ajustan a la cuadrícula de subpíxeles en monitores LCD. Suele ser una buena opción para texto dinámico con alineación central o alineación a la derecha, y en ocasiones ofrece un buen equilibrio entre animación y calidad de texto.

En el siguiente ejemplo se muestra cómo establecer un tipo de ajuste de cuadrícula con código ActionScript.

### Para establecer un tipo de ajuste de cuadrícula en un campo de texto:

1. Cree un nuevo documento de Flash y guárdelo como **gridfittype.fla**.
2. Seleccione Nueva Fuente del menú emergente en la esquina superior derecha del panel Biblioteca.
3. Seleccione la fuente Arial en el menú desplegable Fuente y establezca el tamaño de la fuente en 10 puntos.
4. Escriba el nombre de fuente **Arial-10 (embedded)** en el cuadro de texto Nombre y haga clic en Aceptar.

5. Haga clic con el botón derecho del ratón en el símbolo de fuente en la biblioteca y seleccione Vinculación para abrir el cuadro de diálogo Propiedades de vinculación.
6. Defina el identificador de vinculación en Arial-10 y seleccione las casillas de verificación Exportar para ActionScript y Exportar en primer fotograma
7. Haga clic en Aceptar.
8. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
var my_fmt:TextFormat = new TextFormat();
my_fmt.size = 10;
my_fmt.font = "Arial-10";
var h:Number = Math.floor(Stage.height / 3);

this.createTextField("none_txt", 10, 0, 0, Stage.width, h);
none_txt.antiAliasType = "advanced";
none_txt.embedFonts = true;
none_txt.gridFitType = "none";
none_txt.multiline = true;
none_txt.setNewTextFormat(my_fmt);
none_txt.text = "loading...";
none_txt.wordWrap = true;

this.createTextField("pixel_txt", 20, 0, h, Stage.width, h);
pixel_txt.antiAliasType = "advanced";
pixel_txt.embedFonts = true;
pixel_txt.gridFitType = "pixel";
pixel_txt.multiline = true;
pixel_txt.selectable = false;
pixel_txt.setNewTextFormat(my_fmt);
pixel_txt.text = "loading...";
pixel_txt.wordWrap = true;

this.createTextField("subpixel_txt", 30, 0, h*2, Stage.width, h);
subpixel_txt.antiAliasType = "advanced";
subpixel_txt.embedFonts = true;
subpixel_txt.gridFitType = "subpixel";
subpixel_txt.multiline = true;
subpixel_txt.setNewTextFormat(my_fmt);
subpixel_txt.text = "loading...";
subpixel_txt.wordWrap = true;

var lorem_lv:LoadVars = new LoadVars();
lorem_lv.onData = function(src:String):Void {
  if (src != undefined) {
    none_txt.text = "[antiAliasType=none]\n" + src;
    pixel_txt.text = "[antiAliasType=pixel]\n" + src;
    subpixel_txt.text = "[antiAliasType=subpixel]\n" + src;
  } else {
    trace("unable to load text file");
  }
};
lorem_lv.load("http://www.helpexamples.com/flash/lorem.txt");
```

El código ActionScript anterior se puede dividir en cinco secciones. La primera define el objeto de formato de texto que especifica dos propiedades, `size` y `font`. La propiedad `font` hace referencia al identificador de vinculación del símbolo de fuente que aparece en ese momento en la biblioteca del documento. Las secciones segunda, tercera y cuarta del código crean un nuevo campo de texto dinámico en el escenario y establecen algunas propiedades comunes: `antiAliasType` (que se debe establecer en `advanced`), `embedFonts` (establecido en `true`), `multiline` y `wordWrap`. Asimismo, cada sección aplica el objeto de formato de texto creado en la sección anterior y establece el tipo de ajuste de cuadrícula en `normal`, `pixel` o `subpixel`. La quinta y última sección crea una instancia LoadVars que carga el contenido de un archivo de texto externo en cada uno de los campos de texto creados con el código.

9. Guarde el documento de Flash y seleccione Control > Probar película para probar el archivo SWF.

   Cada campo de texto debe inicializarse con el valor "`loading`...". Una vez se ha cargado correctamente el archivo de texto externo, cada campo de texto muestra algún texto con formato mediante un tipo de ajuste de cuadrícula distinto.

| SUGERENCIA | La tecnología de representación de FlashType utiliza el ajuste de cuadrícula con una rotación de 0 grados. |
|---|---|

## Utilización de la clase TextFormat

Puede utilizar la clase TextFormat para establecer propiedades de formato de un campo de texto. La clase TextFormat incorpora información de formato de caracteres y párrafos. La información de formato de carácter describe el aspecto de cada carácter: el nombre de la fuente, el tamaño en puntos, el color y una URL asociada. La información de formato de párrafo describe el aspecto de un párrafo: margen izquierdo, margen derecho, sangría de la primera línea, y alineación a la izquierda, a la derecha o centrado.

Para utilizar la clase TextFormat, primero debe crear un objeto TextFormat y establecer los estilos de formato de carácter y de párrafo. A continuación, aplique el objeto TextFormat a un campo de texto mediante los métodos `TextField.setTextFormat()` o `TextField.setNewTextFormat()`.

El método `setTextFormat()` cambia el formato de texto aplicado a caracteres individuales, a grupos de caracteres o a todo el cuerpo del texto de un campo de texto. No obstante, el texto nuevo insertado (como el texto especificado por un usuario o insertado con ActionScript) no toma el formato especificado por una llamada `setTextFormat()`. Para especificar el formato predeterminado para el texto que se acaba de insertar, utilice `TextField.setNewTextFormat()`. Para más información, consulte %{setTextFormat (método TextField.setTextFormat)}% y %{setNewTextFormat (método TextField.setNewTextFormat)}% en *Referencia del lenguaje ActionScript 2.0*.

### Para dar formato a un campo de texto con la clase TextFormat:

1. En un documento nuevo de Flash, cree un campo de texto en el escenario mediante la herramienta Texto.

   Escriba un texto en el campo de texto en el escenario, como **Texto en negrita y cursiva de 24 puntos**.

2. En el inspector de propiedades, escriba **myText_txt** en el cuadro de texto Nombre de instancia, seleccione Texto dinámico en el menú emergente Tipo de texto y elija Multilínea en el menú emergente Tipo de línea.

3. Seleccione el fotograma 1 en la línea de tiempo y abra el panel Acciones (Ventana > Acciones).

4. Especifique el código siguiente en el panel Acciones para crear un objeto TextFormat, establezca las propiedades `bold` e `italic` en `true` y la propiedad `size` en 24:

   ```
   // Crear objeto TextFormat.
   var txt_fmt:TextFormat = new TextFormat();
   // Especificar formato de párrafo y caracteres.
   txt_fmt.bold = true;
   txt_fmt.italic = true;
   txt_fmt.size = 24;
   ```

5. Aplique el objeto TextFormat al campo de texto que ha creado en el paso 1 utilizando `TextField.setTextFormat()`:

   ```
   myText_txt.setTextFormat(txt_fmt);
   ```

   Esta versión de `setTextFormat()` aplica el formato especificado a todo el campo de texto. Hay otras dos versiones de este método que permiten aplicar formato a caracteres individuales o a grupos de caracteres. Por ejemplo, el código siguiente aplica el formato de negrita, cursiva y 24 puntos a los tres primeros caracteres especificados en el campo de texto:

   ```
   myText_txt.setTextFormat(0, 3, txt_fmt);
   ```

   Para más información, consulte %{setTextFormat (método TextField.setTextFormat)}% en *Referencia del lenguaje ActionScript 2.0*.

6. Seleccione Control > Probar película para probar la aplicación.

Para más información sobre la utilización de la clase TextFormat, consulte los siguientes temas:

- "Propiedades predeterminadas de los nuevos campos de texto" en la página 450
- "Aplicación de formato al texto con hojas de estilos en cascada" en la página 451

# Propiedades predeterminadas de los nuevos campos de texto

Los campos de texto creados durante la ejecución mediante `createTextField()` reciben un objeto TextFormat predeterminado con las propiedades siguientes:

```
align = "left"
blockIndent = 0
bold = false
bullet = false
color = 0x000000
font = "Times New Roman" (default font is Times on Mac OS X)
indent = 0
italic = false
kerning = false
leading = 0
leftMargin = 0
letterSpacing = 0
rightMargin = 0
size = 12
tabStops = [] (empty array)
target = ""
underline = false
url = ""
```

| NOTA | La propiedad de fuente predeterminada en Mac OS X es Times. |
|---|---|

Para obtener una lista completa de métodos TextFormat y sus correspondientes descripciones, consulte `%{TextFormat}%` en *Referencia del lenguaje ActionScript 2.0*.

# Aplicación de formato al texto con hojas de estilos en cascada

Las hojas de estilos en cascada (CSS) son un mecanismo para trabajar con estilos de texto que pueden aplicarse a documentos HTML o XML. Una hoja de estilos es una recopilación de reglas de formato que especifican cómo se debe dar formato a elementos HTML o XML. Cada regla asocia un nombre de estilo o *selector* con una o varias propiedades de estilo y sus valores. Por ejemplo, el estilo siguiente define un selector denominado `bodyText`:

```
.bodyText {
  text-align: left
}
```

Puede crear estilos que redefinan etiquetas de formato HTML integradas que Flash Player emplea (como `<p>` y`<li>`). También puede crear *clases* de estilos para aplicarlas a elementos HTML específicos mediante el atributo `class` de las etiquetas `<p>` o `<span>` o definir nuevas etiquetas.

Utilice la clase TextField.StyleSheet para trabajar con hojas de estilos de texto. Aunque la clase TextField puede utilizarse con Flash Player 6, la clase TextField.StyleSheet requiere que los archivos SWF se utilicen en Flash Player 7 o posteriores. Puede cargar estilos desde un archivo CSS externo o crearlos de forma nativa con ActionScript. Para aplicar una hoja de estilos a un campo de texto que contenga texto con formato HTML o XML, utilice la propiedad `TextField.styleSheet`. Los estilos definidos en la hoja de estilos se asignan automáticamente a las etiquetas definidas en el documento HTML o XML.

La utilización de hojas de estilos conlleva los siguientes tres pasos básicos:

- Crear un objeto de hoja de estilos desde la clase TextField.StyleSheet (para más información, consulte %{StyleSheet (TextField.StyleSheet)}% en *Referencia del lenguaje ActionScript 2.0*).
- Añadir estilos al objeto de hoja de estilos, ya sea cargándolos desde un archivo CSS externo o bien creando nuevos estilos con ActionScript.
- Asignar la hoja de estilos a un objeto TextField que contenga texto con formato HTML o XML.

Para más información, consulte los siguientes temas:


- “Propiedades CSS admitidas” en la página 452
- “Creación de un objeto de hoja de estilos” en la página 453
- “Carga de archivos CSS externos” en la página 454
- “Creación de estilos con ActionScript” en la página 456
- “Aplicación de estilos a un objeto TextField” en la página 456

Puede encontrar un archivo de origen de ejemplo, formattedText.fla, en la carpeta Samples del disco duro, en el que se muestra cómo aplicar formato CSS al texto que se carga en un archivo SWF durante la ejecución.

En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\LoadText.

En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/LoadText.

## Propiedades CSS admitidas

Flash Player admite un subconjunto de propiedades en la especificación CSS1 original (www.w3.org/TR/REC-CSS1). En la tabla siguiente se muestran las propiedades CSS y los valores admitidos, así como los nombres de propiedad de ActionScript correspondientes. Cada nombre de propiedad de ActionScript se deriva del nombre de propiedad CSS correspondiente; el guión se omite y el carácter siguiente va en mayúscula.

| Propiedad CSS | Propiedad de ActionScript | Uso y valores admitidos |
|---|---|---|
| text-align | textAlign | Los valores reconocidos son `left`, `center`, right y `justify`. |
| font-size | fontSize | Sólo se utiliza la parte numérica del valor. Las unidades (px, pt) no se analizan; los píxeles y los puntos son equivalentes. |
| text-decoration | textDecoration | Los valores reconocidos son `none` y `underline`. |
| margin-left | marginLeft | Sólo se utiliza la parte numérica del valor. Las unidades (px, pt) no se analizan; los píxeles y los puntos son equivalentes. |
| margin-right | marginRight | Sólo se utiliza la parte numérica del valor. Las unidades (px, pt) no se analizan; los píxeles y los puntos son equivalentes. |

| Propiedad CSS | Propiedad de ActionScript | Uso y valores admitidos |
|---|---|---|
| font-weight | fontWeight | Los valores reconocidos son `normal` y `bold`. |
| kerning | kerning | Los valores reconocidos son `true` y `false`. |
| font-style | fontStyle | Los valores reconocidos son `normal` e `italic`. |
| letterSpacing | letterSpacing | Sólo se utiliza la parte numérica del valor. Las unidades (px, pt) no se analizan; los píxeles y los puntos son equivalentes. |
| text-indent | textIndent | Sólo se utiliza la parte numérica del valor. Las unidades (px, pt) no se analizan; los píxeles y los puntos son equivalentes. |
| font-family | fontFamily | Lista de fuentes que se deben utilizar, separadas por comas, en orden descendente de conveniencia. Se puede utilizar cualquier nombre de familia de fuentes. Si especifica un nombre de fuente genérico, se convertirá a una fuente de dispositivo adecuada. Hay disponibles las siguientes conversiones de fuentes: `mono` se convierte en `_typewriter`, `sans-serif` se convierte en `_sans` y `serif` se convierte en `_serif`. |
| color | color | Sólo se admiten valores de color hexadecimales. No se admiten los nombres de los colores (como `blue`). Los colores se escriben en el siguiente formato: `#FF0000`. |

## Creación de un objeto de hoja de estilos

Las hojas de estilos CSS están representadas en ActionScript mediante la clase TextField.StyleSheet. Esta clase sólo está disponible para los archivos SWF que se vayan a utilizar en Flash Player 7 o versiones posteriores. Para crear un objeto de hoja de estilos, se llama a la función constructora de la clase TextField.StyleSheet:

```
var newStyle:TextField.StyleSheet = new TextField.StyleSheet();
```

Para añadir estilos a un objeto de hoja de estilos, puede cargar un archivo CSS externo en el objeto o definir los estilos en ActionScript. Consulte “Carga de archivos CSS externos” en la página 454 y “Creación de estilos con ActionScript” en la página 456.

Puede encontrar un archivo de origen de ejemplo, formattedText.fla, en la carpeta Samples del disco duro, en el que se muestra cómo aplicar formato CSS al texto que se carga en un archivo SWF durante la ejecución.

En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\LoadText.

En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/LoadText.

## Carga de archivos CSS externos

Puede definir estilos en un archivo CSS externo y después cargar dicho archivo en un objeto de hoja de estilos. Los estilos definidos en el archivo CSS se añaden al objeto de hoja de estilos. Para cargar un archivo CSS externo, utilice el método `load()` de la clase TextField.StyleSheet. Para determinar cuándo ha finalizado la carga del archivo CSS, utilice el controlador de eventos `onLoad` del objeto de hoja de estilos.

En el ejemplo siguiente, se crea y se carga un archivo CSS externo y se utiliza el método `TextField.StyleSheet.getStyleNames()` para recuperar los nombres de los estilos cargados.

### Para cargar una hoja de estilos externa:

1. En el editor de CSS o de texto que prefiera, cree un nuevo archivo.
2. Añada al archivo las siguientes definiciones de estilo:

```
.bodyText {
  font-family: Arial,Helvetica,sans-serif;
  font-size: 12px;
}

.headline {
  font-family: Arial,Helvetica,sans-serif;
  font-size: 24px;
}
```

3. Guarde el archivo CSS como styles.css.
4. En Flash, cree un nuevo archivo FLA.
5. En la línea de tiempo (Ventana > Línea de tiempo), seleccione Capa 1.
6. Abra el panel Acciones (Ventana > Acciones).

7. Añada el código siguiente al panel Acciones:

```
var styles:TextField.StyleSheet = new TextField.StyleSheet();
styles.onLoad = function(success:Boolean):Void {
  if (success) {
    // mostrar nombres de estilos.
    trace(this.getStyleNames());
  } else {
    trace("Error loading CSS file.");
  }
};
styles.load("styles.css");
```

NOTA

En el fragmento de código anterior, `this.getStyleNames()` hace referencia al objeto `styles` que ha creado en la primera línea de ActionScript.

8. Guarde el archivo FLA en el mismo directorio que contiene el archivo styles.css.
9. Pruebe el documento de Flash (Control > Probar película).

   Se mostrarán los nombres de los dos estilos en el panel Salida:

   ```
   .bodyText,.headline
   ```

   Si aparece "Error loading CSS file." en el panel Salida, asegúrese de que el archivo FLA y el archivo CSS se encuentren en el mismo directorio y de que ha escrito correctamente el nombre del archivo CSS.

Al igual que ocurre con los demás métodos de ActionScript que cargan datos a través de la red, el archivo CSS debe residir en el mismo dominio que el archivo SWF que está cargando el archivo. (Consulte "Acceso a varios dominios y subdominios entre archivos SWF" en la página 736.) Para más información sobre la utilización de CSS con Flash, consulte %{StyleSheet (TextField.StyleSheet)}% en *Referencia del lenguaje ActionScript 2.0*.

Puede encontrar un archivo de origen de ejemplo, formattedText.fla, en la carpeta Samples del disco duro, en el que se muestra cómo aplicar formato CSS al texto que se carga en un archivo SWF durante la ejecución.

En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\LoadText.

En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/LoadText.

## Creación de estilos con ActionScript

Puede crear estilos de texto con ActionScript mediante el método `setStyle()` de la clase TextField.StyleSheet. Este método usa dos parámetros: el nombre del estilo y un objeto que define las propiedades de dicho estilo.

Por ejemplo, el código siguiente crea un objeto de hoja de estilos denominado `styles` que define dos estilos idénticos a los que ya ha importado (consulte "Carga de archivos CSS externos" en la página 454).

```
var styles:TextField.StyleSheet = new TextField.StyleSheet();
styles.setStyle("bodyText",
  {fontFamily: 'Arial,Helvetica,sans-serif',
  fontSize: '12px'}
);
styles.setStyle("headline",
  {fontFamily: 'Arial,Helvetica,sans-serif',
  fontSize: '24px'}
);
```

## Aplicación de estilos a un objeto TextField

Para aplicar un objeto de hoja de estilos a un objeto TextField, asigne el objeto de hoja de estilos a la propiedad `styleSheet` del campo de texto.

```
textObj_txt.styleSheet = styles;
```

NOTA

No confunda la *propiedad* `TextField.styleSheet` con la *clase* TextField.StyleSheet. La combinación de mayúsculas y minúsculas indica la diferencia.

Cuando asigna un objeto de hoja de estilos a un objeto TextField, se producen los cambios siguientes en el comportamiento habitual del campo de texto:

- Las propiedades `text` y `htmlText` del campo de texto, así como las variables asociadas al campo de texto, siempre contienen el mismo valor y se comportan de forma idéntica.
- El campo de texto pasa a ser de sólo lectura y el usuario no puede editarlo.
- Los métodos `setTextFormat()` y `replaceSel()` de la clase TextField ya no funcionan con el campo de texto. La única manera de cambiar el campo es modificando las propiedades `text` o `htmlText` del campo de texto o cambiando la variable asociada al mismo.
- Todo el texto asignado a la propiedad `text` o a la propiedad `htmlText` del campo de texto o toda variable asociada se almacena literalmente; todo lo escrito en estas propiedades puede recuperarse en el formato de texto original.

# Aplicación de una hoja de estilos a un componente TextArea

Para aplicar una hoja de estilos a un componente TextArea, deberá crear un objeto de hoja de estilos y asignarle estilos HTML mediante la clase TextField.StyleSheet. Posteriormente deberá asignar la hoja de estilos a la propiedad `styleSheet` del componente TextArea.

En los siguientes ejemplos se crea un objeto de hoja de estilos, `styles`, que se asigna a la instancia de componente `myTextArea`.

### Utilización de una hoja de estilos con un componente TextArea:

1. Cree un nuevo documento de Flash y guárdelo como **textareastyle.fla**.
2. Arrastre un componente TextArea desde la carpeta User Interface del panel Componentes al escenario y asígnele el nombre de instancia **myTextArea**.
3. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
// Create a new style sheet object and set styles for it.
var styles:TextField.StyleSheet = new TextField.StyleSheet();
styles.setStyle("html", {fontFamily:'Arial,Helvetica,sans-serif',
            fontSize:'12px',
            color:'#0000FF'});
styles.setStyle("body", {color:'#00CCFF',
            textDecoration:'underline'});
styles.setStyle("h1",{fontFamily:'Arial,Helvetica,sans-serif',
            fontSize:'24px',
            color:'#006600'});

/* Assign the style sheet object to myTextArea component. Set html
  property to true, set styleSheet property to the style sheet object.
  */
myTextArea.styleSheet = styles;
myTextArea.html = true;

var myVars:LoadVars = new LoadVars();
// Define onData handler and load text to be displayed.
myVars.onData = function(myStr:String):Void {
  if (myStr != undefined) {
    myTextArea.text = myStr;
  } else {
    trace("Unable to load text file.");
  }
};
myVars.load("http://www.helpexamples.com/flash/myText.htm");
```

   El bloque de código anterior crea una nueva instancia TextField.StyleSheet que define tres estilos para las etiquetas HTML: `html`, `body` y `h1`. A continuación, el objeto de hoja de estilos se aplica al componente TextArea y se activa el formato HTML. El código ActionScript restante define un objeto LoadVars que carga un archivo HTML externo y rellena el área de texto con el texto cargado.
4. Seleccione Control > Probar película para probar el documento de Flash.

## Combinación de estilos

Los estilos CSS en Flash Player son aditivos; es decir, cuando los estilos están anidados, cada nivel de anidación puede aportar información de estilo, que se añade para dar como resultado el formato final.

En el siguiente ejemplo se muestran datos XML asignados a un campo de texto:

```
<sectionHeading>This is a section</sectionHeading>
<mainBody>This is some main body text, with one
<emphasized>emphatic</emphasized> word.</mainBody>
```

Para la palabra *emphatic* del texto anterior, el estilo `emphasized` está anidado dentro del estilo `mainBody`. El estilo `mainBody` aporta las reglas de color, tamaño de fuente y decoración. El estilo `emphasized` añade una regla de grosor de fuente a dichas reglas. Se aplicará el formato a la palabra *enfatizada* mediante una combinación de las reglas especificadas por `mainBody` y `emphasized`.

## Utilización de las clases de estilos

Puede crear "clases" de estilos (no clases reales de ActionScript 2.0) para aplicarlas a una etiqueta `<p>` o `<span>` empleando el atributo `class` de cualquiera de estas etiquetas. Cuando se aplica a una etiqueta `<p>`, el estilo afecta a todo el párrafo. También puede aplicar estilos a un espacio de texto que utiliza una clase de estilo mediante la etiqueta `<span>`.

Por ejemplo, la siguiente hoja de estilos define dos clases de estilos: `mainBody` y `emphasis`:

```
.mainBody {
  font-family: Arial,Helvetica,sans-serif;
  font-size: 24px;
}
.emphasis {
  color: #666666;
  font-style: italic;
}
```

En el texto HTML que asigna a un campo de texto, puede aplicar estos estilos a etiquetas `<p>` y `<span>`, como se muestra en el siguiente fragmento:

```
<p class='mainBody'>This is <span class='emphasis'>really exciting!</
  span></p>
```

# Estilos para etiquetas HTML incorporadas

Flash Player admite un subconjunto de etiquetas HTML. Para más información, consulte "Utilización de texto en formato HTML" en la página 465. Puede asignar un estilo CSS a cada instancia de una etiqueta HTML incorporada que aparezca en un campo de texto. Por ejemplo, el siguiente código define un estilo para la etiqueta HTML incorporada `<p>`. Todas las instancias de dicha etiqueta tendrán el estilo especificado por la regla de estilo.

```
p {
  font-family: Arial,Helvetica,sans-serif;
  font-size: 12px;
  display: inline;
}
```

En la tabla siguiente se muestra a qué etiquetas HTML incorporadas se puede aplicar un estilo y cómo se aplica cada estilo:

| Nombre del estilo | Cómo se aplica el estilo |
|---|---|
| `p` | Afecta a todas las etiquetas `<p>`. |
| `body` | Afecta a todas las etiquetas `<body>`. El estilo `p`, si se especifica, tiene prioridad sobre el estilo `body` . |
| `li` | Afecta a todas las etiquetas de viñeta `<li>`. |
| `a` | Afecta a todas las etiquetas de anclaje `<a>`. |
| `a:link` | Afecta a todas las etiquetas de anclaje `<a>`. Este estilo se aplica después del estilo `a`. |
| `a:hover` | Se aplica a una etiqueta de anclaje `<a>` cuando se coloca el puntero del ratón sobre el vínculo. Este estilo se aplica después de los estilos `a` y `a:link`. Cuando el puntero del ratón se desplaza fuera del vínculo, el estilo `a:hover` desaparece del vínculo. |
| `a:active` | Se aplica a una etiqueta de anclaje `<a>` cuando el usuario hace clic en el vínculo. Este estilo se aplica después de los estilos `a` y `a:link`. Cuando se suelta el botón del ratón, el estilo `a:active` desaparece del vínculo. |

# Ejemplo de utilización de estilos con HTML

En esta sección se ofrece un ejemplo de utilización de estilos con las etiquetas HTML. Puede crear una hoja de estilos que contenga estilos para algunas etiquetas incorporadas y defina algunas clases de estilos. A continuación, podrá aplicar dicha hoja de estilos a un objeto TextField que contenga texto en formato HTML.

### Para aplicar formato a HTML con una hoja de estilos:

1. Cree un archivo nuevo en el editor de texto o de CSS que prefiera.
2. Añada al archivo la siguiente definición de hoja de estilos:

```
p {
  color: #000000;
  font-family: Arial,Helvetica,sans-serif;
  font-size: 12px;
  display: inline;
}

a:link {
  color: #FF0000;
}

a:hover{
  text-decoration: underline;
}

.headline {
  color: #000000;
  font-family: Arial,Helvetica,sans-serif;
  font-size: 18px;
  font-weight: bold;
  display: block;
}

.byline {
  color: #666600;
  font-style: italic;
  font-weight: bold;
  display: inline;
}
```

   Esta hoja de estilos define los estilos para dos etiquetas HTML incorporadas (`<p>` y `<a>`) que se aplicarán a todas las instancias de dichas etiquetas. También define dos clases de estilos (`.headline` y `.byline`) que se aplicarán a párrafos y espacios de texto específicos.
3. Guarde el archivo como **html_styles.css**.
4. Cree un nuevo archivo de texto en un editor de texto o HTML y guarde el documento como **myText.htm**.

Añada el siguiente texto al archivo:

```
<p class='headline'>Flash adds FlashType rendering technology!</
  p><p><span class='byline'>San Francisco, CA</span>--Macromedia Inc.
  announced today a new version of Flash that features a brand new font
  rendering technology called FlashType, most excellent at rendering
  small text with incredible clarity and consistency across platforms.
  For more information, visit the <a href='http://
  www.macromedia.com'>Macromedia Flash web site.</a></p>
```

**NOTA** Si copia y pega esta cadena de texto, asegúrese de que quita los saltos de línea que puedan haberse añadido a la cadena de texto.

5. Cree un nuevo documento de Flash en la herramienta de edición de Flash.
6. Seleccione el primer fotograma de la Capa 1 de la línea de tiempo (Ventana > Línea de tiempo).
7. Abra el panel Acciones (Ventana > Acciones) y añada el siguiente código al mismo:

```
this.createTextField("news_txt", 99, 50, 50, 450, 300);
news_txt.border = true;
news_txt.html = true;
news_txt.multiline = true;
news_txt.wordWrap = true;
// Create a new style sheet and LoadVars object.
var myVars_lv:LoadVars = new LoadVars();
var styles:TextField.StyleSheet = new TextField.StyleSheet();
// Location of CSS and text files to load.
var txt_url:String = "myText.htm";
var css_url:String = "html_styles.css";
// Define onData handler and load text to display.
myVars_lv.onData = function(src:String):Void {
  if (src != undefined) {
    news_txt.htmlText = src;
  } else {
    trace("Unable to load HTML file");
  }
};
myVars_lv.load(txt_url);

// Define onLoad handler and Load CSS file.
styles.onLoad = function(success:Boolean):Void {
  if (success) {
    /* Si la hoja de estilos se ha cargado sin errores,
       asígnela al objeto de texto,
       y asigne el texto HTML al campo de texto. */
    news_txt.styleSheet = styles;
    news_txt.text = storyText;
  } else {
    trace("Unable to load CSS file.");
  }
};
styles.load(css_url);
```

| NOTA | En este código ActionScript, el texto se carga de un archivo externo. Para más información sobre cómo cargar datos externos, consulte el Capítulo 15, "Utilización de imágenes, sonido y vídeo". |
|---|---|

8. Guarde el archivo como **news_html.fla** en el mismo directorio que contiene el archivo CSS creado en el paso 3.
9. Seleccione Control > Probar película para ver los estilos aplicados al texto HTML automáticamente.

## Utilización de estilos para definir etiquetas nuevas

Si define un estilo nuevo en una hoja de estilos, dicho estilo puede utilizarse como etiqueta, igual que utilizaría una etiqueta HTML incorporada. Por ejemplo, si una hoja de estilos define un estilo CSS denominado `sectionHeading`, puede utilizar `<sectionHeading>` como un elemento en cualquier campo de texto asociado con la hoja de estilos. Esto permite asignar texto con formato XML arbitrario directamente a un campo de texto, de modo que se aplica formato al texto de manera automática siguiendo las reglas de la hoja de estilos.

Por ejemplo, la siguiente hoja de estilos crea los estilos `sectionHeading`, `mainBody` y `emphasized`:

```
.sectionHeading {
  font-family: Verdana, Arial, Helvetica, sans-serif;
  font-size: 18px;
  display: block
}
.mainBody {
  color: #000099;
  text-decoration: underline;
  font-size: 12px;
  display: block
}
.emphasized {
  font-weight: bold;
  display: inline
}
```

A continuación, puede rellenar un campo de texto asociado con dicha hoja de estilos con el siguiente texto en formato XML:

```
<sectionHeading>This is a section</sectionHeading>
<mainBody>This is some main body text,
with one <emphasized>emphatic</emphasized> word.
</mainBody>
```

# Ejemplo de utilización de estilos con XML

En esta sección, creará un archivo FLA que incluye texto con formato XML. Creará una hoja de estilos empleando ActionScript en lugar de importar estilos de un archivo CSS, como se muestra en "Ejemplo de utilización de estilos con HTML" en la página 460.

### Para aplicar formato a XML con una hoja de estilos:

1. En Flash, cree un archivo FLA.
2. Con la herramienta Texto, cree un campo de texto de unos 400 píxeles de anchura y 300 píxeles de altura.
3. Abra el inspector de propiedades (Ventana > Propiedades > Propiedades) y seleccione el campo de texto.
4. En el inspector de propiedades, seleccione Texto dinámico en el menú Tipo de texto, seleccione Multilínea en el menú Tipo de línea y elija la opción Generar texto como HTML; a continuación, escriba **news_txt** en el cuadro de texto Nombre de instancia.
5. En la Capa 1 de la línea de tiempo (Ventana > Línea de tiempo), seleccione el primer fotograma.
6. Para crear el objeto de hoja de estilos, abra el panel Acciones (Ventana > Acciones) y añada el siguiente código al mismo:

```
var styles:TextField.StyleSheet = new TextField.StyleSheet();
styles.setStyle("mainBody", {
  color:'#000000',
  fontFamily:'Arial,Helvetica,sans-serif',
  fontSize:'12',
  display:'block'
});
styles.setStyle("title", {
  color:'#000000',
  fontFamily:'Arial,Helvetica,sans-serif',
  fontSize:'18',
  display:'block',
  fontWeight:'bold'
});
styles.setStyle("byline", {
  color:'#666600',
  fontWeight:'bold',
  fontStyle:'italic',
  display:'inline'
});
styles.setStyle("a:link", {
  color:'#FF0000'
});
styles.setStyle("a:hover", {
  textDecoration:'underline'
});
```

Este código crea un nuevo objeto de hoja de estilos denominado `styles` que define estilos mediante el método `setStyle()`. Los estilos son los mismos que los creados en un archivo CSS externo anteriormente en este capítulo.

7. Para crear el texto XML que se debe asignar al campo de texto, abra un editor de texto e introduzca el siguiente texto en un documento nuevo:

```
<story><title>Flash now has FlashType</title><mainBody><byline>San
  Francisco, CA</byline>--Macromedia Inc. announced today a new version
  of Flash that features the new FlashType rendering technology. For
  more information, visit the <a href="http://
  www.macromedia.com">Macromedia Flash website</a></mainBody></story>
```

> **NOTA** Si copia y pega esta cadena de texto, asegúrese de que quita los saltos de línea que puedan haberse añadido a la cadena de texto. Seleccione Caracteres ocultos en el menú emergente del panel Acciones para ver y eliminar cualquier salto de línea adicional.

8. Guarde el archivo de texto como **story.xml**.
9. En Flash, añada el siguiente código en el panel Acciones tras el código del paso 6.

   Este código carga el documento story.xml, asigna el objeto de hoja de estilos a la propiedad `styleSheet` del campo de texto y asigna el texto XML al campo de texto:

```
var my_xml:XML = new XML();
my_xml.ignoreWhite = true;
my_xml.onLoad = function(success:Boolean):Void {
  if (success) {
    news_txt.styleSheet = styles;
    news_txt.text = my_xml;
  } else {
    trace("Error loading XML.");
  }
};
my_xml.load("story.xml");
```

> **NOTA** En este código ActionScript se cargan datos XML de un archivo externo. Para más información sobre cómo cargar datos externos, consulte el Capítulo 15, “Utilización de imágenes, sonido y vídeo”.

10. Guarde el archivo como **news_xml.fla** en la misma carpeta que story.xml.
11. Ejecute el archivo SWF (Control > Probar película) para ver los estilos aplicados automáticamente al texto del campo de texto.

# Utilización de texto en formato HTML

Flash Player admite un subconjunto de etiquetas HTML estándar, como `<p>` y `<li>`, que puede utilizar para aplicar estilo al texto de cualquier campo de texto dinámico o de introducción de texto. Los campos de texto de Flash Player 7 y versiones posteriores también admiten la etiqueta `<img>`, que permite incorporar archivos de imagen (JPEG, GIF, PNG), archivos SWF y clips de película a un campo de texto. Flash Player distribuye automáticamente el texto alrededor de las imágenes incorporadas a los campos de texto, de forma muy similar a como los navegadores Web distribuyen el texto alrededor de las imágenes incorporadas en las páginas HTML. Para más información, consulte "Incorporación de imágenes, archivos SWF y clips de película en campos de texto" en la página 474.

Flash Player también admite la etiqueta `<textformat>`, que permite aplicar estilos de formato de párrafo de la clase TextFormat a los campos de texto compatibles con HTML. Para más información, consulte "Utilización de la clase TextFormat" en la página 448.

Para más información sobre texto con formato HTML, consulte los siguientes temas:



## Propiedades y sintaxis necesarias para utilizar texto con formato HTML

Para utilizar HTML en un campo de texto, deberá establecer varias propiedades del campo de texto, bien en el inspector de propiedades o bien utilizando ActionScript:

- Active el formato HTML del campo de texto seleccionando la opción Generar texto como HTML en el inspector de propiedades o estableciendo la propiedad `html` del campo de texto en `true`.
- Para utilizar etiquetas HTML como `<p>`, `<br>` y `<img>`, deberá convertir el campo de texto en un campo de texto de varias líneas seleccionando la opción Multilínea en el inspector de propiedades o estableciendo la propiedad `multiline` del campo de texto en `true`.
- En ActionScript, establezca el valor de `TextField.htmlText` en la cadena de texto con formato HTML que desea que aparezca.

Por ejemplo, el código siguiente activa la aplicación de formato HTML para un campo de texto denominado `headline_txt` y, a continuación, asigna HTML al campo de texto:

```
this.createTextField("headline_txt", 1, 10, 10, 500, 300);
headline_txt.html = true;
headline_txt.wordWrap = true;
headline_txt.multiline = true;
headline_txt.htmlText = "<font face='Times New Roman' size='25'>This is how
  you assign HTML text to a text field.</font><br>It's very useful.</br>";
```

Para mostrar código HTML correctamente, deberá utilizar la sintaxis correcta. Los atributos de las etiquetas HTML deben escribirse entre comillas dobles (") o simples ('). Los valores de los atributos que no estén entre comillas pueden provocar resultados imprevisibles, como una presentación incorrecta del texto. Por ejemplo, Flash Player *no* genera correctamente el fragmento de código HTML siguiente porque el valor asignado al atributo `align` (`left`) no está entre comillas:

```
this.createTextField("myField_txt", 10, 10, 10, 400, 200);
myField_txt.html = true;
myField_txt.htmlText = "<p align=left>This is left-aligned text</p>";
```

Si escribe los valores de los atributos entre comillas dobles, debe *anular* el valor de las comillas (`\"`). Cualquiera de las siguientes formas de hacerlo resulta aceptable:

```
myField_txt.htmlText = "<p align='left'>This uses single quotes</p>";
myField_txt.htmlText = "<p align=\"left\">This uses escaped double quotes</
  p>";
myField_txt.htmlText = '<p align="left">This uses outer single quotes</p>';
myField_txt.htmlText = '<p align=\'left\'>This uses escaped single quotes</
  p>';
```

No es necesario anular el valor de las comillas dobles si carga texto de un archivo externo; sólo es necesario si asigna una cadena de texto en ActionScript.

## Etiquetas HTML admitidas

En esta sección se enumeran las etiquetas HTML incorporadas admitidas por Flash Player. También puede crear nuevos estilos y etiquetas mediante CSS; consulte "Aplicación de formato al texto con hojas de estilos en cascada" en la página 451.

Para más información sobre las etiquetas con formato HTML admitidas, consulte los siguientes temas:

- "Etiqueta de anclaje" en la página 467
- "Etiqueta de negrita" en la página 468
- "Etiqueta de salto de línea" en la página 468
- "Etiqueta de fuente" en la página 468

## Etiqueta de anclaje

La etiqueta `<a>` crea un vínculo de hipertexto y admite los atributos siguientes:

- `href` Cadena formada por un máximo de 128 caracteres que especifica la URL de la página que debe cargarse en el navegador. La URL puede ser absoluta o relativa a la ubicación del archivo SWF que carga la página. En ejemplo de referencia absoluta a una URL es `http://www.macromedia.com`; un ejemplo de referencia relativa es `/index.html`.
- `target` Especifica el nombre de la ventana de destino en la que se cargará la página. Entre las opciones figuran `_self`, `_blank`, `_parent` y `_top`. La opción `_self` especifica el marco actual de la ventana actual, `_blank` especifica una ventana nueva, `_parent` especifica el nivel superior del marco actual y `_top` especifica el marco del nivel más alto de la ventana actual.

Por ejemplo, el código HTML siguiente crea el vínculo “Ir a página principal”, que abre www.macromedia.com en una ventana de navegador nueva.

```
urlText_txt.htmlText = "<a href='http://www.macromedia.com'
  target='_blank'>Go home</a>";
```

Puede utilizar el protocolo especial `asfunction` para hacer que el vínculo ejecute una función de ActionScript en un archivo SWF en lugar de abrir una URL. Para más información sobre el protocolo `asfunction`, consulte %{asfunction protocol}% en *Referencia del lenguaje ActionScript 2.0*.

También puede definir los estilos `a:link`, `a:hover` y `a:active` para las etiquetas de anclaje mediante las hojas de estilos. Consulte “Estilos para etiquetas HTML incorporadas” en la página 459.

| NOTA | Las URL absolutas deben llevar el prefijo `http://`, ya que, en caso contrario, Flash las considera URL relativas. |
|---|---|

## Etiqueta de negrita

La etiqueta <b> muestra el texto en negrita, como se muestra en el siguiente ejemplo:

```
text3_txt.htmlText = "He was <b>ready</b> to leave!";
```

Debe haber un tipo de letra en negrita disponible para la fuente utilizada para mostrar el texto.

## Etiqueta de salto de línea

La etiqueta <br> crea un salto de línea en el campo de texto. Debe configurar el campo de texto como campo de texto de varias líneas para poder utilizar esta etiqueta.

En el siguiente ejemplo, se produce un salto de línea entre las frases:

```
this.createTextField("text1_txt", 1, 10, 10, 200, 100);
text1_txt.html = true;
text1_txt.multiline = true;
text1_txt.htmlText = "The boy put on his coat.<br />His coat was <font
  color='#FF0033'>red</font> plaid.";
```

## Etiqueta de fuente

La etiqueta <font> especifica una fuente o una lista de fuentes para mostrar el texto.

La etiqueta de fuente admite los atributos siguientes:

- `color` Sólo se admiten los valores de colores hexadecimales (#FFFFFF). Por ejemplo, el código HTML siguiente crea texto de color rojo:

  ```
  myText_txt.htmlText = "<font color='#FF0000'>This is red text</font>";
  ```

- `face` Especifica el nombre de la fuente que se utiliza. Como se muestra en el siguiente ejemplo, puede especificar una lista de nombres de fuentes separados por comas, en cuyo caso Flash Player selecciona la primera fuente disponible:

  ```
  myText_txt.htmlText = "<font face='Times, Times New Roman'>Displays as
    either Times or Times New Roman...</font>";
  ```

  Si la fuente especificada no está instalada en el sistema del usuario o no está incorporada en el archivo SWF, Flash Player elegirá una fuente alternativa.

  Para más información sobre la incorporación de fuentes en aplicaciones de Flash, consulte %{embedFonts (propiedad TextField.embedFonts)}% en *Referencia del lenguaje ActionScript 2.0* y "Establecimiento de opciones de texto dinámico y de entrada" en *Utilización de Flash*.

- `size` Especifica el tamaño de la fuente en píxeles, como se muestra en el siguiente ejemplo:

  ```
  myText_txt.htmlText = "<font size='24' color='#0000FF'>This is blue, 24-
    point text</font>";
  ```

  También puede utilizar tamaños en puntos relativos en lugar de un tamaño en píxeles, como por ejemplo +2 o -4.

## Etiqueta de imagen

La etiqueta `<img>` permite incorporar archivos de imagen externos (JPEG, GIF, PNG), archivos SWF y clips de película en los campos de texto y las instancias de componente TextArea. El texto fluye automáticamente alrededor de las imágenes incorporadas en los campos de texto o componentes. Para utilizar esta etiqueta, debe configurar el campo dinámico o de introducción de texto como campo de varias líneas y con ajuste de texto.

**Para crear un campo de texto multilínea con ajuste de texto, siga uno de estos procedimientos:**

- En el entorno de edición de Flash, seleccione un campo de texto en el escenario y, a continuación, en el inspector de propiedades, seleccione Multilínea en el menú Tipo de texto.
- En el caso de un campo de texto creado durante la ejecución con %{createTextField (método MovieClip.createTextField)}%, defina las propiedades %{multiline (propiedad TextField.multiline)}% y %{multiline (propiedad TextField.multiline)}% de la instancia del campo de texto nuevo en `true`.

La etiqueta `<img>` tiene un solo atributo necesario, `src`, que especifica la ruta a un archivo de imagen, a un archivo SWF o al identificador de vinculación de un símbolo de clip de película en la biblioteca. Todos los demás atributos son opcionales.

La etiqueta `<img>` admite los atributos siguientes:

- `src` Especifica la URL de un archivo de imagen o SWF o el identificador de vínculo de un símbolo de clip de película de la biblioteca. Este atributo es necesario; todos los demás son opcionales. Los archivos externos (JPEG, GIF, PNG y SWF) no se muestran hasta que no se han descargado completamente.
- `id` Especifica el nombre de la instancia de clip de película (creada por Flash Player) que contiene el archivo de imagen, SWF o el clip de película incorporado. Resulta de utilidad si desea controlar el contenido incorporado con ActionScript.
- `width` Anchura, en píxeles, de la imagen, el archivo SWF o el clip de película que se va a insertar.
- `height` Altura, en píxeles, de la imagen, archivo SWF o clip de película que se va a insertar.
- `align` Especifica la alineación horizontal de la imagen incorporada en el campo de texto. Los valores válidos son `left` y `right`. El valor predeterminado es `left`.
- `hspace` Especifica la cantidad de espacio horizontal que rodea a la imagen sin que aparezca texto alguno. El valor predeterminado es 8.
- `vspace` Especifica la cantidad de espacio vertical que rodea a la imagen sin que aparezca texto alguno. El valor predeterminado es 8.

Para más información y ejemplos sobre la utilización de la etiqueta `<img>`, consulte "Incorporación de imágenes, archivos SWF y clips de película en campos de texto" en la página 474.

## Etiqueta de cursiva

La etiqueta `<i>` muestra el texto al que se aplica en cursiva, como se muestra en el siguiente código:

```
That is very <i>interesting</i>.
```

Este código de ejemplo se mostraría de la siguiente forma:

That is very *interesting*.

Debe haber un tipo de letra en cursiva disponible para la fuente utilizada.

## Etiqueta de elemento de lista

La etiqueta `<li>` coloca una viñeta delante del texto incluido en la etiqueta, como se muestra en el siguiente código:

```
Lista de la compra:
<li>Apples</li>
<li>Oranges</li>
<li>Lemons</li>
```

Este código de ejemplo se mostraría de la siguiente forma:

Lista de la compra:

- Apples
- Oranges
- Lemons

**NOTA**

Flash Player no reconoce las listas ordenadas y no ordenadas (etiquetas `<ol>` y `<ul>`), por lo que no afectan al modo en que se muestran las listas. Todos los elementos de lista llevan viñeta.

## Etiqueta de párrafo

La etiqueta `<p>` crea un párrafo nuevo. Debe configurar el campo de texto como campo de texto de varias líneas para poder utilizar esta etiqueta.

La etiqueta `<img>` admite los atributos siguientes:

- `align` Especifica la alineación del texto dentro del párrafo; los valores válidos son `left`, `right`, `justify` y `center`.

- `class` Especifica una clase de estilos CSS definida por un objeto TextField.StyleSheet. Para más información, consulte "Utilización de las clases de estilos" en la página 458.

  En el ejemplo siguiente se utiliza el atributo `align` para alinear el texto a la derecha de un campo de texto.

```
this.createTextField("myText_txt", 1, 10, 10, 400, 100);
myText_txt.html = true;
myText_txt.multiline = true;
myText_txt.htmlText = "<p align='right'>This text is aligned on the
  right side of the text field</p>";
```

  En el ejemplo siguiente se utiliza el atributo `class` para asignar una clase de estilo de texto a una etiqueta `<p>`:

```
var myStyleSheet:TextField.StyleSheet = new TextField.StyleSheet();
myStyleSheet.setStyle(".blue", {color:'#99CCFF', fontSize:18});
this.createTextField("test_txt", 10, 0, 0, 300, 100);
test_txt.html = true;
test_txt.styleSheet = myStyleSheet;
test_txt.htmlText = "<p class='blue'>This is some body-styled text.</
  p>.";
```

## Etiqueta de espacio

La etiqueta `<span>` sólo se puede utilizar con los estilos de texto CSS. Para más información, consulte "Aplicación de formato al texto con hojas de estilos en cascada" en la página 451. Admite el atributo siguiente:

- `class` Especifica una clase de estilos CSS definida por un objeto TextField.StyleSheet. Para más información sobre la creación de clases de estilos de texto, consulte "Utilización de las clases de estilos" en la página 458.

## Etiqueta de formato de texto

La etiqueta `<textformat>` permite utilizar un subconjunto de las propiedades de formato de párrafo de la clase TextFormat en los campos de texto HTML, incluidos el interlineado, la sangría, los márgenes y las tabulaciones. Puede combinar las etiquetas `<textformat>` con las etiquetas HTML incorporadas.

La etiqueta `<textformat>` tiene los atributos siguientes:

- `blockindent` Especifica la sangría de bloque en puntos; corresponde a `TextFormat.blockIndent`. (Consulte %{blockIndent (propiedad TextFormat.blockIndent)}% en *Referencia del lenguaje ActionScript 2.0*.)
- `indent` Especifica la sangría desde el margen izquierdo hasta el primer carácter del párrafo; corresponde a `TextFormat.indent`. Permite utilizar enteros negativos. (Consulte %{indent (propiedad TextFormat.indent)}% en *Referencia del lenguaje ActionScript 2.0*.)

- `leading` Especifica la cantidad de espacio vertical entre las líneas (interlineado); corresponde a `TextFormat.leading`. Permite utilizar enteros negativos. (Consulte %{leading (propiedad TextFormat.leading)}% en *Referencia del lenguaje ActionScript 2.0*.)
- `leftmargin` Especifica el margen izquierdo del párrafo en puntos; corresponde a `TextFormat.leftMargin`. (Consulte %{leftMargin (propiedad TextFormat.leftMargin)}% en *Referencia del lenguaje ActionScript 2.0*.)
- `rightmargin` Especifica el margen derecho del párrafo en puntos; corresponde a `TextFormat.rightMargin`. (Consulte %{rightMargin (propiedad TextFormat.rightMargin)}% en *Referencia del lenguaje ActionScript 2.0*.)
- `tabstops` Especifica las tabulaciones personalizadas como una matriz de enteros no negativos; corresponde a `TextFormat.tabStops`. (Consulte %{tabStops (propiedad TextFormat.tabStops)}% en *Referencia del lenguaje ActionScript 2.0*.)

La siguiente tabla de datos con encabezados de fila en negrita es el resultado del código de ejemplo del procedimiento que aparece debajo:

| Name | Age | Occupation |
|---|---|---|
| Rick | 33 | Detective |
| AJ | 34 | Detective |

### Para crear una tabla de datos con formato utilizando tabulaciones:

1. Cree un nuevo documento de Flash y guárdelo como **tabstops.fla**.
2. En la línea de tiempo, seleccione el primer fotograma de la capa 1.
3. Abra el panel Acciones (Ventana > Acciones) e introduzca el siguiente código en el mismo:

```
// Crear nuevo campo de texto.
this.createTextField("table_txt", 99, 50, 50, 450, 100);
table_txt.multiline = true;
table_txt.html = true;
// Crea encabezados de columna en negrita y separados por tabulaciones.
var rowHeaders:String = "<b>Name\tAge\tOccupation</b>";

// Crea filas con datos.
var row_1:String = "Rick\t33\tDetective";
var row_2:String = "AJ\t34\tDetective";

// Establece dos puntos de tabulación (tabstops), uno en 50 y otro en 100
  puntos.
table_txt.htmlText = "<textformat tabstops='[50,100]'>";
table_txt.htmlText += rowHeaders;
table_txt.htmlText += row_1;
table_txt.htmlText += row_2 ;
table_txt.htmlText += "</textformat>";
```

   La utilización de la secuencia de escape del carácter de tabulación (`\t`) añade tabulaciones entre cada columna de la tabla. El texto se añade empleando el operador `+=`.

4. Seleccione Control > Probar película para ver la tabla con formato.

### Etiqueta de subrayado

La etiqueta <u> subraya el texto al que se aplica, como se muestra en el siguiente código:

```
This is <u>underlined</u> text.
```

Este código se mostraría de la siguiente forma:

This is underlined text.

## Entidades HTML admitidas

Las entidades HTML permiten mostrar ciertos caracteres en campos de texto con formato HTML de forma que no se interpreten como HTML. Por ejemplo, se utilizan los caracteres menor que (<) y mayor que (>) para encerrar las etiquetas HTML, como <img> y . Para mostrar los caracteres menor que y mayor que en campos de texto con formato HTML en Flash, es preciso sustituir las entidades HTML por dichos caracteres. El siguiente código ActionScript crea un campo de texto con formato HTML en el escenario y utiliza entidades HTML para mostrar la cadena "<b>" sin que el texto deba aparecer en negrita:

```
this.createTextField("my_txt", 10, 100, 100, 100, 19);
my_txt.autoSize = "left";
my_txt.html = true;
my_txt.htmlText = "The &lt;b&gt; tag makes text appear <b>bold</b>.";
```

Durante la ejecución, el ejemplo de código muestra en Flash el siguiente texto en el escenario:

The <b> tag makes text appear **bold**.

Además de los símbolos menor que y mayor que, Flash también reconoce otras entidades HTML que se enumeran en la siguiente tabla.

| Entidad | Descripción |
|---|---|
| `<` | < (menor que) |
| `>` | > (mayor que) |
| `&` | & (ampersand) |
| `&quot;` | " (comillas dobles) |
| `&apos;` | ' (apóstrofe, comilla simple) |

Flash también admite códigos de caracteres explícitos como `&#39;` (ampersand - ASCII) y `&#x0026;` (ampersand - Unicode).

En el siguiente código ActionScript se muestra cómo utilizar códigos de caracteres ASCII o Unicode para incorporar un carácter de tilde (~):

```
this.createTextField("my_txt", 10, 100, 100, 100, 19);
my_txt.autoSize = "left";
my_txt.html = true;
my_txt.htmlText = "&#126;"; // tilde (ASCII)
my_txt.htmlText += "\t"
my_txt.htmlText += "&#x007E;"; // tilde (Unicode)
```

# Incorporación de imágenes, archivos SWF y clips de película en campos de texto

En Flash Player 7 y versiones posteriores, puede utilizar la etiqueta `<img>` para incorporar archivos e imagen (JPEG, GIF, PNG), archivos SWF y clips de película en campos de texto dinámicos y de introducción de texto, así como en instancias de componentes TextArea. (Para consultar la lista completa de los atributos de la etiqueta `<img>`, consulte "Etiqueta de imagen" en la página 469.)

Flash muestra los medios incorporados en un campo de texto con su tamaño completo. Para especificar las dimensiones de los medios que está incorporando, utilice los atributos `height` y `width` de la etiqueta `<img>`. (Consulte "Especificación de los valores de altura y anchura" en la página 476.)

En general, las imágenes incorporadas a un campo de texto aparecen en la línea que sigue a la etiqueta `<img>`. Sin embargo, si la etiqueta `<img>` es el primer carácter de un campo de texto, la imagen aparece en la primera línea del campo de texto.

## Incorporación de archivos SWF y de imagen

Para incorporar un archivo de imagen o SWF en un campo de texto, especifique la ruta absoluta o relativa del archivo de imagen (JPEG, GIF, PNG) o SWF en el atributo `src` de la etiqueta `<img>`. Por ejemplo, el código de muestra siguiente inserta un archivo GIF que se encuentra en el mismo directorio que el archivo SWF (una dirección relativa, en línea y sin conexión).

### Incorporación de una imagen en un campo de texto:

1. Cree un nuevo documento de Flash y guárdelo como **embedding.fla**.
2. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
this.createTextField("image1_txt", 10, 50, 50, 450, 150);
image1_txt.html = true;
image1_txt.htmlText = "<p>Here's a picture from my vacation:<img
  src='beach.gif'>";
```

El código anterior crea un nuevo campo de texto dinámico en el escenario, activa el formato HTML y añade texto y una imagen local al campo de texto.

3. Añada el siguiente código ActionScript debajo del código añadido en el paso anterior:

```
this.createTextField("image2_txt", 20, 50, 200, 400, 150);
image2_txt.html = true;
image2_txt.htmlText = "<p>Here's a picture from my garden:<img
  src='http://www.helpexamples.com/flash/images/image2.jpg'>";
```

También puede insertar una imagen utilizando una dirección absoluta. El código anterior inserta un archivo JPEG ubicado en un directorio que se encuentra en un servidor. El archivo SWF que contiene este código podría encontrarse en la unidad de disco duro o en un servidor.

4. Guarde el documento de Flash y seleccione Control > Probar película para probar el documento.

El campo de texto superior debería incluir una frase y probablemente un mensaje de texto en el panel Salida en el que se indicara que Flash no pudo encontrar el archivo beach.gif en el directorio actual. El campo de texto inferior debería incluir una frase y una imagen con una flor cargada desde el servidor remoto.

Copie una imagen GIF en el mismo directorio que el archivo FLA, cámbiele el nombre a beach.gif y seleccione Control > Probar película para volver a probar el documento de Flash.

| NOTA | Al utilizar URL absolutas, deberá asegurarse de que la URL lleva el prefijo `http://`. |
|---|---|

## Incorporación de símbolos de clip de película

Para incorporar un símbolo de clip de película a un campo de texto, especifique el identificador de vínculo del símbolo para el atributo `src` de la etiqueta `<img>`. (Para obtener información sobre la definición de un identificador de vinculación, consulte "Asociación de un símbolo de clip de película al escenario" en la página 386.)

Por ejemplo, el código siguiente inserta un símbolo de clip de película con el identificador de vínculo `symbol_ID` en un campo de texto dinámico con el nombre de instancia `textField_txt`.

### Para incorporar un clip de película en un campo de texto:

1. Cree un nuevo documento de Flash y guárdelo como **embeddedmc.fla**.
2. Dibuje una nueva forma en el escenario o seleccione Archivo > Importar > Importar a escenario y elija una imagen que tenga aproximadamente 100 píxeles de anchura por 100 de altura.

3. Convierta la forma o imagen importada en el paso anterior seleccionándola en el escenario y presionando F8 para abrir el cuadro de diálogo Convertir en símbolo.
4. Establezca el comportamiento como Clip de película e introduzca un nombre de símbolo descriptivo. Seleccione el cuadrado superior izquierdo de la cuadrícula de punto de registro y haga clic en Avanzado para cambiar al modo avanzado, si aún no lo ha hecho.
5. Seleccione las casillas de verificación Exportar para ActionScript y Exportar en primer fotograma.
6. Introduzca el identificador de vinculación **img_id** en el cuadro de texto Identificador y haga clic en Aceptar.
7. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
this.createTextField("textField_txt", 10, 0, 0, 300, 200);
textField_txt.html = true;
textField_txt.htmlText = "<p>Here's a movie clip symbol:<img
  src='img_id'>";
```

   Para que un clip de película incorporado se muestre correctamente y en su totalidad, el punto de registro del símbolo correspondiente debe estar en el punto (0,0).
8. Guarde los cambios en el documento de Flash.
9. Seleccione Control > Probar película para probar el documento de Flash.

## Especificación de los valores de altura y anchura

Si especifica los atributos `width` y `height` para una etiqueta `<img>`, en el campo de texto se reserva espacio para el archivo de imagen, el archivo SWF o el clip de película. Una vez que un archivo de imagen o SWF se ha descargado completamente, se mostrará en el espacio reservado. Flash aumenta o reduce la escala de los medios de acuerdo con los valores especificados para `height` y `width`. Debe introducir valores tanto para el atributo `height` como para el atributo `width` para poder aplicar la escala a la imagen.

Si no especifica los valores para `height` y `width`, no se reserva espacio para los medios incorporados. Una vez que un archivo de imagen o SWF se ha descargado completamente, Flash lo inserta en el campo de texto con su tamaño completo y redistribuye el texto alrededor.

| NOTA | Si va a cargar las imágenes dinámicamente en un campo de texto que contiene texto, es recomendable especificar la anchura y la altura de la imagen original de manera que el texto se ajuste correctamente alrededor del espacio que reserve para la imagen. |
|---|---|

## Control de los medios incorporados con ActionScript

Flash crea un clip de película nuevo para cada etiqueta `<img>` y lo incorpora al objeto TextField. El atributo `id` de la etiqueta `<img>` permite asignar un nombre de instancia al clip de película creado. Esto permite controlar ese clip de película con ActionScript.

El clip de película creado por Flash se añade como clip de película secundario en el campo de texto que contiene la imagen.

En el siguiente ejemplo se incorpora un archivo SWF en un campo de texto.

### Para incorporar un archivo SWF en un campo de texto:

1. Cree un nuevo documento de Flash.
2. Cambie el tamaño del documento en el escenario a 100 por 100 píxeles.
3. Utilice la herramienta Rectángulo para dibujar un cuadrado rojo en el escenario.
4. Cambie el tamaño del cuadrado a 80 por 80 píxeles con el inspector de propiedades y mueva la forma al centro del escenario.
5. Seleccione el fotograma 20 en la línea de tiempo y presione F7 (Windows o Macintosh) para insertar un fotograma clave en blanco.
6. Dibuje un círculo azul en el fotograma 20 del escenario con la herramienta Óvalo.
7. Cambie el tamaño del círculo a 80 por 80 píxeles con el inspector de propiedades y muévalo al centro del escenario.
8. Haga clic en un fotograma en blanco entre el fotograma 1 y el 20 y establezca el tipo de interpolación en Forma en el inspector de propiedades.
9. Guarde el documento como **animation.fla**.
10. Seleccione Control > Probar película para previsualizar la animación.

    El archivo SWF se crea en el mismo directorio que el archivo FLA. Para que este ejercicio funcione correctamente, el archivo SWF se debe generar de forma que se pueda cargar en un archivo FLA independiente.
11. Cree un archivo FLA nuevo y guárdelo como **animationholder.fla**.

    Guarde el archivo en la misma carpeta que el archivo animation.xml creado anteriormente.
12. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
this.createTextField("textField_txt", 10, 0, 0, 300, 200);
textField_txt.html = true;
textField_txt.htmlText = "Here's an interesting animation: <img
  src='animation.swf' id='animation_mc'>";
```

    En este caso, la ruta completa del clip de película recién creado es `textField_txt.animation_mc`.

13. Guarde los cambios en el documento de Flash y seleccione Control > Probar película para previsualizar la animación en el campo de texto.

Para controlar la reproducción del archivo SWF en el campo de texto, complete el siguiente ejercicio.

### Para controlar la reproducción de un archivo SWF en un campo de texto:

1. Siga los pasos del primer procedimiento de "Control de los medios incorporados con ActionScript" en la página 477.
2. Cree una instancia de botón en el escenario y asígnele el nombre de instancia **stop_btn** en el inspector de propiedades.
3. Añada el siguiente código ActionScript debajo del código existente en el fotograma 1 de la línea de tiempo principal:

```
stop_btn.onRelease = function() {
  textField_txt.animation_mc.stop();
};
```

4. Seleccione Control > Probar película para probar la aplicación.

   Siempre que haga clic en la instancia de botón `stop_btn`, la línea de tiempo de la animación anidada dentro del campo de texto se detendrá.

Para obtener información sobre cómo incorporar medios en un hipervínculo, consulte "Creación de vínculos de hipertexto a partir de medios incorporados" en la página 478.

## Creación de vínculos de hipertexto a partir de medios incorporados

Para crear un vínculo de hipertexto a partir de un archivo de imagen, un archivo SWF o un clip de película incorporado, coloque la etiqueta `<img>` dentro de una etiqueta `<a>`:

```
textField_txt.htmlText = "Click the image to return home<a
  href='home.htm'><img src='home.jpg'></a>";
```

Cuando se coloca el puntero del ratón sobre una imagen, un archivo SWF o un clip de película situado entre etiquetas `<a>`, el puntero del ratón se convierte en un icono de "mano que señala", el mismo que se muestra para vínculos de hipertexto estándar. Las acciones interactivas, como hacer clic con el ratón y presionar teclas, no se registran en los archivos SWF y clips de película que están entre etiquetas `<a>`.

Para obtener información sobre la incorporación de medios, consulte "Creación de vínculos de hipertexto a partir de medios incorporados" en la página 478.

# Ejemplo: Creación de texto desplazable

Existen varios métodos para crear texto desplazable en Flash. Puede permitir el desplazamiento en los campos de texto dinámico y de introducción de texto seleccionando la opción Desplazamiento permitido del menú Texto o del menú contextual o haciendo doble clic en el selector del campo con la tecla Mayús presionada.

Puede utilizar las propiedades `scroll` y `maxscroll` del objeto TextField para controlar el desplazamiento vertical, y las propiedades `hscroll` y `maxhscroll` para controlar el desplazamiento horizontal en un campo de texto. Las propiedades `scroll` y `hscroll` especifican las posiciones de desplazamiento vertical y horizontal actuales respectivamente; puede leer y escribir estas propiedades. Las propiedades `maxscroll` y `maxhscroll` especifican respectivamente las posiciones de desplazamiento vertical y horizontal máximas; estas propiedades son de sólo lectura.

El componente TextArea proporciona un método sencillo para crear campos de texto desplazable sin apenas crear scripts. Para más información, consulte "Componente TextArea" en la *Referencia del lenguaje de componentes*.

### Para crear un campo de texto dinámico desplazable:

Realice uno de los siguientes pasos:

- Haga doble clic con la tecla Mayús presionada en el selector del campo de texto dinámico.
- Seleccione el campo de texto dinámico con la herramienta Selección y seleccione Texto > Desplazamiento permitido.
- Seleccione el campo de texto dinámico con la herramienta Selección. Haga clic con el botón derecho del ratón (Windows) o con la tecla Control presionada (Macintosh) en el campo de texto dinámico y seleccione Texto > Desplazamiento permitido.

### Para utilizar la propiedad scroll para crear texto desplazable:

1. Realice uno de los siguientes pasos:
   - Con la herramienta Texto, arrastre un campo de texto en el escenario. Asigne al campo de texto el nombre de instancia **textField_txt** en el inspector de propiedades.
   - Utilice ActionScript para crear un campo de texto de forma dinámica con el método `MovieClip.createTextField()`. Asigne al campo de texto el nombre de instancia **textField_txt** como parámetro del método.

   **NOTA** Si *no* va a cargar texto de forma dinámica en el archivo SWF, seleccione Texto > Desplazamiento permitido del menú principal.

2. Cree un botón Arriba y un botón Abajo o seleccione Ventana > Bibliotecas comunes > Botones y arrastre los botones al escenario.

   Estos botones servirán para desplazar el texto hacia arriba y hacia abajo.
3. Seleccione el botón Abajo en el escenario y escriba **down_btn** en el cuadro de texto Nombre de instancia.
4. Seleccione el botón Arriba en el escenario y escriba **up_btn** en el cuadro de texto Nombre de instancia.
5. Seleccione el fotograma 1 en la línea de tiempo y, en el panel Acciones (Ventana > Acciones), introduzca el código siguiente para desplazar el texto hacia abajo en el campo de texto:

```
down_btn.onPress = function() {
  textField_txt.scroll += 1;
};
```

6. Tras el código ActionScript del paso 5, introduzca el código siguiente para desplazar el texto hacia arriba:

```
up_btn.onPress = function() {
  textField_txt.scroll -= 1;
};
```

   El texto que se cargue en el campo de texto `textField_txt` se podrá desplazar empleando los botones arriba y abajo.

# Cadenas y la clase String

En programación, una cadena es una serie de caracteres ordenados. Las cadenas se utilizan con frecuencia en documentos y archivos de clases de Flash para mostrar texto en las aplicaciones, como, por ejemplo, en campos de texto. Asimismo, puede almacenar valores como cadenas que puede emplear en una aplicación con diversos fines. Puede colocar cadenas directamente en el código ActionScript encerrando los caracteres de datos entre comillas. Para más información sobre la creación de cadenas, consulte "Creación de cadenas" en la página 489. Para obtener información sobre la utilización de campos de texto, consulte "Utilización de la clase TextField" en la página 410.

Puede asociar cada carácter con un código de carácter específico, que también puede utilizar opcionalmente para mostrar texto. Por ejemplo, el carácter "A" se representa mediante el código de carácter Unicode 0041 o 65 en ASCII (código americano estándar para intercambio de información). Para más información sobre códigos de caracteres y tablas de códigos, consulte www.unicode.org/charts. Como puede observar, la forma en que se representan las cadenas en un documento de Flash depende en gran medida del conjunto de caracteres que elija y de la forma en que codifique los caracteres.

La *codificación de caracteres* se refiere al código o método de representación del conjunto de caracteres de un idioma en códigos representativos, como, por ejemplo, valores numéricos. El *código de carácter* (como se menciona en el párrafo anterior) es la tabla de valores asignados (como la tabla ASCII, en la que A equivale a 65). El método de codificación lo descifra en un programa informático.

Por ejemplo, cada letra del idioma inglés tendría su código numérico representativo en una codificación de caracteres. ASCII codifica todas las letras, números y algunos símbolos en versiones binarias de 7 bits de cada entero. ASCII es un conjunto de caracteres que consta de 95 caracteres imprimibles y numerosos caracteres de control que utilizan los equipos informáticos para representar texto.

Al igual que ASCII, *Unicode* es otra forma de asociar un código a una letra del alfabeto. Dado que ASCII no admite conjuntos de caracteres grandes, como el del chino, Unicode constituye un valioso estándar para codificar idiomas. Unicode es el estándar para conjuntos de caracteres que pueden representar cualquier idioma. Se trata de un estándar cuyo objetivo es contribuir al desarrollo en múltiples idiomas. El código de carácter designa el carácter al que representa y el estándar intenta ofrecer una forma universal de codificar los caracteres que forman parte de cualquier idioma. Las cadenas pueden mostrarse en cualquier equipo, plataforma o software que se utilice. A partir de ahí, es responsabilidad del programa en cuestión (como Flash o un navegador Web) mostrar el glifo del carácter (su aspecto visual).

Con el paso de los años, el número de caracteres que admite Unicode se ha ampliado para dar cabida a más idiomas (e idiomas con más caracteres). Las codificaciones de caracteres se denominan UTF (Formato de transformación de Unicode, Unicode Transformation Format) y UCS (Conjunto de caracteres universal, Universal Character Set), que incluye UTF-8, UTF-16 y UTF-32. Los números de la codificación UTF representan el número de bits de una unidad, mientras que los números de una codificación UCS representan los bytes.

- UTF-8 es la codificación estándar para el intercambio de texto, como, por ejemplo, el que llevan a cabo los sistemas de correo en línea. UTF es un sistema de 8 bits.
- UTF-16 se utiliza habitualmente para procesamiento interno.

Las cadenas pueden tener diversas longitudes en sus aplicaciones. Puede determinar la longitud de una cadena, aunque ésta puede variar, dependiendo del idioma que utilice. Asimismo, podría encontrar un carácter de terminación al final de una cadena, y este carácter nulo no tiene ningún valor. Este carácter de terminación no es un carácter real, aunque puede utilizarlo para determinar cuándo termina una cadena. Por ejemplo, si trabaja con conexiones de socket, podría observar el carácter de terminación para conocer el final de una cadena (como, por ejemplo, en un programa de chat).

Puede encontrar un archivo de origen de muestra, strings.fla, en la carpeta Samples del disco duro. En él se muestra cómo crear un procesador de textos sencillo que compara y recupera selecciones de cadenas y subcadenas.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Strings.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Strings.

Para más información sobre las cadenas y la clase String, consulte los siguientes temas:


- "Panel Cadenas" en la página 482
- "Utilización de la clase Locale" en la página 483
- "Utilización de un editor de método de entrada" en la página 485
- "La clase String" en la página 488
- "Creación de cadenas" en la página 489
- "El carácter de escape" en la página 490
- "Análisis y comparación de caracteres en las cadenas" en la página 491
- "Conversión y concatenación de cadenas" en la página 495
- "Devolución de subcadenas" en la página 498


## Panel Cadenas

Este panel le permite crear y mantener contenido multilingüe. Puede especificar diferente contenido para los campos de texto que abarcan varios idiomas y que Flash determine automáticamente el contenido que debe mostrar en función del idioma del equipo en el que se ejecuta Flash Player.

Para obtener información general sobre el panel Cadenas y su utilización en las aplicaciones, consulte los siguientes temas en *Utilización de Flash*:


- "Creación de un texto con varios idiomas con el panel Cadenas" en la página 409
- "Edición de texto en el panel Cadenas" en la página 413
- "Traducción de texto en el panel Cadenas o en un archivo XML" en la página 419

■ "Importación de un archivo XML al panel Cadenas" en la página 420

Puede utilizar la clase Locale para controlar cómo se muestra el texto en varios idiomas. Para más información, consulte "Utilización de la clase Locale" en la página 483 y `%{Locale (mx.lang.Locale)}%` en *Referencia del lenguaje ActionScript 2.0*.

## Utilización de la clase Locale

La clase Locale (mx.lang.Locale) permite controlar cómo aparece el texto en varios idiomas en una aplicación de Flash durante la ejecución. Con el panel Cadenas se pueden emplear ID de cadena en lugar de literales de cadena en los campos de texto dinámicos, lo que permite crear un archivo SWF que muestra el texto cargado desde un archivo XML específico de idioma. Con los métodos siguientes se pueden mostrar las cadenas de un idioma específico incluidas en los archivos XLIFF (Formato de archivo de intercambio de localización XML, XML Localization Interchange File Format).

**automáticamente en tiempo de ejecución** Flash Player sustituye los ID con cadenas del archivo XML que coincidan con el código de idioma de sistema predeterminado devuelto por `%{language (propiedad capabilities.language)}%`.

**manualmente utilizando idioma del escenario** Se sustituyen los ID con cadenas en tiempo de compilación y Flash Player no los puede modificar.

**mediante ActionScript en tiempo de ejecución** Con ActionScript se controla la sustitución de ID de cadena durante la ejecución. Esta opción permite controlar la sincronización y el idioma de la sustitución de ID de cadena.

Puede utilizar las propiedades y los métodos de la clase Locale cuando quiera sustituir los ID de cadena mediante ActionScript para controlar la aplicación cuando se reproducen en Flash Player. Para ver una demostración de cómo utilizar la clase Locale, consulte el procedimiento siguiente.

### Para utilizar la clase Locale para crear sitios en varios idiomas:

1. Cree un nuevo documento de Flash y guárdelo como **locale.fla**.
2. Abra el panel Cadenas (Ventana > Otros paneles > Cadenas) y haga clic en Configuración.
3. Seleccione dos idiomas, en (inglés) y fr (francés) y haga clic en Añadir para añadirlos al panel Idiomas activos.
4. Seleccione la opción mediante ActionScript en tiempo de ejecución, establezca el idioma de tiempo de ejecución predeterminado como francés y haga clic en Aceptar.
5. Arrastre un componente ComboBox desde la carpeta User Interface del panel Componentes (Ventana > Componentes) al escenario y asígnele el nombre de instancia **lang_cb**.

6. Cree un campo de texto dinámico en el escenario mediante la herramienta Texto y asígnele el nombre de instancia **greeting_txt**.
7. Con el texto seleccionado en el escenario, escriba un identificador de cadena **greeting** en el cuadro de texto ID del panel Cadenas y haga clic en Aplicar.

   Observe que Flash convierte la cadena greeting en IDS_GREETING.
8. En la cuadrícula del panel Cadenas, escriba la cadena **hello** en la columna en.
9. Escriba la cadena **bonjour** en la columna fr.

   Estas cadenas se utilizan cuando se emplea el cuadro combinado `lang_cb` para cambiar el idioma en el escenario.
10. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
import mx.lang.Locale;
Locale.setLoadCallback(localeListener);
lang_cb.dataProvider = Locale.languageCodeArray.sort();
lang_cb.addEventListener("change", langListener);
greeting_txt.autoSize = "left";
Locale.loadLanguageXML(lang_cb.value);

function langListener(eventObj:Object):Void {
  Locale.loadLanguageXML(eventObj.target.value);
}
function localeListener(success:Boolean):Void {
  if (success) {
    greeting_txt.text = Locale.loadString("IDS_GREETING");
  } else {
    greeting_txt.text = "unable to load language XML file.";
  }
}
```

El código ActionScript anterior se divide en dos secciones. La primera importa la clase Locale y especifica una función callback a la que se llama siempre que se termina de cargar un archivo XML de idioma. A continuación, el cuadro combinado `lang_cb` se rellena con una matriz ordenada de idiomas disponibles. Siempre que cambia el valor de `lang_cb`, el distribuidor de eventos de Flash activa la función `langListener()`, que carga el archivo XML del idioma especificado. La segunda sección del código define dos funciones, `langListener()` y `localeListener()`. Se llama a la primera función, `langListener()`, siempre que el usuario cambia el valor del cuadro combinado `lang_cb`. Se llama a la segunda, `localeListener()`, siempre que se termina de cargar un archivo XML de idioma. La función comprueba si la carga se ha realizado correctamente y, si es así, establece la propiedad `text` de la instancia `greeting_txt` en el saludo en el idioma seleccionado.

11. Seleccione Control > Probar película para probar el documento de Flash.

| SUGERENCIA | El archivo XML utilizado debe tener el formato XLIFF. |
|---|---|

| ATENCIÓN | La clase Locale es diferente a otras clases en Referencia del lenguaje ActionScript 2.0, ya que no forma parte de Flash Player. Dado que esta clase se instala en la ruta de clases de edición de Flash, se compila automáticamente en los archivos SWF. El uso de la clase Locale aumenta ligeramente el tamaño del archivo SWF, ya que la clase se debe compilar en el mismo. |
|---|---|

Para más información, consulte `%{Locale (mx.lang.Locale)}%` en *Referencia del lenguaje ActionScript 2.0*.

## Utilización de un editor de método de entrada

Los editores de método de entrada (IME) permiten a los usuarios introducir caracteres de texto no ASCII en idiomas asiáticos, como chino, japonés o coreano. La clase IME en ActionScript permite manipular directamente el IME del sistema operativo en la aplicación Flash Player que se ejecuta en un equipo cliente.

También puede utilizar código ActionScript para determinar lo siguiente:

- Si hay instalado un IME en el equipo del usuario.
- Si el IME está activado o desactivado en el equipo del usuario.
- Qué modo de conversión utiliza el IME actual.

La clase IME puede determinar qué modo de conversión utiliza el IME actual: por ejemplo, si está activo el IME de japonés, puede determinar si el modo de conversión es Hiragana, Katakana (etc.) mediante el método `System.IME.getConversionMode()`. Puede establecerlo con el método `System.IME.setConversionMode()`.

| NOTA | En la actualidad, no se puede saber *qué* IME está activo (en caso de haber alguno disponible) ni cambiar de un IME a otro (por ejemplo, de inglés a japonés o de coreano a chino) |
|---|---|

Asimismo, puede desactivar o activar el IME utilizando la aplicación durante la ejecución y realizar otras funciones, dependiendo del sistema operativo del usuario. Puede comprobar si un sistema cuenta con un IME con la propiedad `System.capabilities.hasIME`. El ejemplo siguiente muestra cómo determinar si el usuario tiene un IME instalado y activo.

### Para determinar si el usuario tiene un IME instalado y activo:

1. Cree un nuevo documento de Flash y guárdelo como **ime.fla**.
2. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
if (System.capabilities.hasIME) {
  if (System.IME.getEnabled()) {
    trace("You have an IME installed and enabled.");
  } else {
    trace("You have an IME installed but not enabled.");
  }
} else {
  trace("Please install an IME and try again.");
}
```

   El código anterior primero comprueba si el sistema actual cuenta con un IME instalado. Si es así, Flash comprueba si está activado.

3. Seleccione Control > Probar película para probar el documento.

   Aparecerá un mensaje en el panel Salida indicando si se cuenta o no con el IME y si está activo en ese momento.

La clase IME también puede utilizarse para activar y desactivar el IME en Flash durante la ejecución. El siguiente ejemplo requiere que se cuente con un IME instalado en el sistema. Para más información sobre la instalación de un IME en su plataforma, consulte los siguientes vínculos:

- www.microsoft.com/globaldev/default.mspx
- http://developer.apple.com/documentation/
- http://java.sun.com

Puede activar o desactivar un IME mientras se reproduce el archivo SWF, como se muestra en este ejemplo.

### Para activar o desactivar un editor de método de entrada durante la ejecución:

1. Cree un nuevo documento de Flash y guárdelo como **ime2.fla**.
2. Cree dos instancias de símbolo de botón en el escenario y asígneles los nombres de instancia **enable_btn** y **disable_btn**.

3. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
checkIME();

var my_fmt:TextFormat = new TextFormat();
my_fmt.font = "_sans";

this.createTextField("ime_txt", 10, 100, 10, 320, 240);
ime_txt.border = true;
ime_txt.multiline = true;
ime_txt.setNewTextFormat(my_fmt);
ime_txt.type = "input";
ime_txt.wordWrap = true;

enable_btn.onRelease = function() {
  System.IME.setEnabled(true);
};
disable_btn.onRelease = function() {
  System.IME.setEnabled(false);
};

function checkIME():Boolean {
  if (System.capabilities.hasIME) {
    if (System.IME.getEnabled()) {
      trace("You have an IME installed and enabled.");
      return true;
    } else {
      trace("You have an IME installed but not enabled.");
      return false;
    }
  } else {
    trace("Please install an IME and try again.");
    return false;
  }
}
```

El código anterior se divide en cinco secciones independientes. La primera sección llama al método `checkIME()`, que muestra un mensaje en el panel Salida si el sistema cuenta con un IME instalado o activo. La segunda define un objeto de formato de texto personalizado, que establece la fuente como `_sans`. La tercera crea un campo de introducción de texto y aplica el formato de texto personalizado. La cuarta crea algunos controladores de eventos para las instancias `enable_btn` y `disable_btn` creadas en el paso anterior. La quinta y última sección del código define la función `checkIME()` personalizada, que comprueba si el sistema actual cuenta con un IME instalado y, si es así, si está o no activo.

4. Guarde el archivo FLA y seleccione Control > Probar película para probar el documento.

| NOTA | Este ejemplo requiere que se cuente con un IME instalado en el sistema. Para obtener información sobre la instalación de un IME, consulte los vínculos que preceden a este ejemplo. |
|---|---|

Escriba texto en el campo de introducción de texto en el escenario. Cambie el IME a otro idioma y vuelva a escribir en el campo de introducción de texto. Flash Player introduce los caracteres con el nuevo IME. Si hace clic en el botón `disable_btn` en el escenario, Flash vuelve a utilizar el idioma anterior y omite la configuración del IME actual.

Para obtener información sobre `System.capabilities.hasIME`, consulte `%{hasIME (propiedad capabilities.hasIME)}%` en *Referencia del lenguaje ActionScript 2.0*.

# La clase String

Una cadena (string) es también una clase y un tipo de datos en el lenguaje ActionScript básico. El tipo de datos String representa una secuencia de caracteres de 16 bits que puede incluir letras, números y signos de puntuación. Las cadenas se almacenan como caracteres Unicode empleando el formato UTF-16. Una operación sobre un valor de cadena (String) devuelve una nueva instancia de la cadena. El valor predeterminado de una variable declarada con el tipo de datos String es `null`.

Para más información sobre cadenas, datos y valores, consulte el Capítulo 10, “Datos y tipos de datos”.

La clase String contiene métodos que le permiten trabajar con cadenas de texto. Las cadenas son importantes al utilizar numerosos objetos; los métodos descritos en este capítulo son útiles al trabajar con cadenas utilizadas en muchos objetos, como, por ejemplo, instancias de TextField, XML, ContextMenu y FileReference.

La clase String es un envolvente para el tipo de datos primitivo de cadena y proporciona métodos y propiedades que le permiten manipular valores de cadena primitivos. Puede convertir el valor de cualquier objeto en una cadena utilizando la función `String()`. Todos los métodos de la clase String, salvo `concat()`, `fromCharCode()`, `slice()` y `substr()`, son genéricos, lo que significa que los métodos llaman a la función `toString()` antes de realizar sus operaciones y puede llamar a estos métodos con otros objetos que no sean String.

Dado que todos los índices de cadenas están basados en cero, el índice del último carácter de cualquier cadena `myStr` es `myStr.length - 1`.

Puede encontrar un archivo de origen de muestra, strings.fla, en la carpeta Samples del disco duro. En él se muestra cómo crear un procesador de textos sencillo que compara y recupera selecciones de cadenas y subcadenas.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Strings.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Strings.

## Creación de cadenas

Puede llamar a cualquiera de los métodos de la clase String utilizando el método constructor `new String()` o un nuevo valor de literal de cadena. Si especifica un literal de cadena, el intérprete de ActionScript lo convierte automáticamente en un objeto String temporal, llama al método y descarta el objeto String temporal. También puede utilizar la propiedad `String.length` con un literal de cadena.

No debe confundir un *literal* de cadena con un *objeto* String. Para más información sobre literales de cadenas y el objeto String, consulte el Capítulo 4, “Literales”, en la página 94.

En el siguiente ejemplo, la línea de código crea el literal de cadena `firstStr`. Para declarar un literal de cadena, utilice delimitadores de comilla simple recta (`'`) o de comilla doble recta (`"`).

### Para crear y utilizar cadenas:

1. Cree un nuevo documento de Flash y guárdelo como **strings.fla**.
2. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
var firstStr:String = "foo";
var secondStr:String = new String("foo");
trace(firstStr == secondStr); // true
var thirdStr:String;
trace(thirdStr); // no definido
```

   Este código define tres objetos String, uno que emplea un literal de cadena, otro que emplea el operador `new` y otro que no tiene valor inicial. Las cadenas se pueden comparar utilizando el operador de igualdad (==), como se muestra en la tercera línea de código. En el caso de las variables, se especifica el tipo de datos sólo cuando se define la variable.
3. Seleccione Control > Probar película para probar el documento.

Utilice siempre literales de cadena a no ser que necesite utilizar un objeto String específicamente. Para más información sobre literales de cadenas y el objeto String, consulte el Capítulo 4, “Literales”, en la página 94.

Para utilizar los delimitadores de comilla simple recta (') y de comilla doble recta (") dentro de un literal de cadena, puede utilizar el carácter de barra invertida (\) como escape del carácter. Las dos cadenas siguientes son equivalentes:

```
var firstStr:String = "That's \"fine\"";
var secondStr:String = 'That\'s "fine"';
```

Para más información sobre el uso del carácter de barra invertida en cadenas, consulte "El carácter de escape" en la página 490.

Recuerde que los caracteres de "comilla curva" o "comilla especial" no se pueden utilizar en el código ActionScript, a diferencia de las comillas rectas (') y ("), que sí pueden utilizarse en el código. Al pegar texto de otra fuente en el código ActionScript, por ejemplo, de la Web o de un documento de Word, asegúrese de que utiliza delimitadores de comillas rectas.

Puede encontrar un archivo de origen de muestra, strings.fla, en la carpeta Samples del disco duro. En él se muestra cómo crear un procesador de textos sencillo que compara y recupera selecciones de cadenas y subcadenas.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Strings.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Strings.

## El carácter de escape

Puede utilizar el carácter de escape de barra invertida (\) para definir otros caracteres en los literales de cadena.

| Secuencia de escape | Descripción |
|---|---|
| `\b` | El carácter de retroceso. |
| `\f` | El carácter de salto de página. |
| `\n` | El carácter de nueva línea. |
| `\r` | El retorno de carro. |
| `\t` | El carácter de tabulación. |
| `\u`*nnnn* | El carácter Unicode con el código de carácter especificado por el número hexadecimal *nnnn*. Por ejemplo, `\u263a` es el carácter de sonrisa. |
| `\x`*nn* | El carácter ASCII con el código de carácter especificado por el número hexadecimal *nn*. |

| Secuencia de escape | Descripción |
|---|---|
| \' | Una comilla simple. |
| \" | Una comilla doble. |
| \\ | Un carácter de barra invertida simple. |

Para más información sobre literales de cadena, consulte el Capítulo 4, “Literales”, en la página 94 y “Creación de cadenas” en la página 489.

## Análisis y comparación de caracteres en las cadenas

Cada carácter de una cadena tiene una posición de índice en la cadena (un entero). La posición de índice del primer carácter es 0. Por ejemplo, en la cadena `yellow`, el carácter `y` está en la posición 0 y el carácter `w`, en la posición 5.

Cada cadena tiene una propiedad `length` que es igual al número de caracteres de la cadena:

```
var companyStr:String = "macromedia";
trace(companyStr.length); // 10
```

Una cadena vacía o nula tienen una longitud cero:

```
var firstStr:String = new String();
trace(firstStr.length); // 0

var secondStr:String = "";
trace(secondStr.length); // 0
```

Si una cadena no contiene un valor, su longitud se establece en undefined:

```
var thirdStr:String;
trace(thirdStr.length); // no definido
```

**ADVERTENCIA**

Si la cadena contiene un carácter de byte nulo (\0), el valor de la cadena se trunca.

También puede utilizar los códigos de caracteres para definir una cadena. Para más información sobre códigos de caracteres y codificación de caracteres, consulte “Cadenas y la clase String” en la página 480.

El siguiente ejemplo crea una variable denominada `myStr` y establece el valor de la cadena en función de los valores ASCII pasados al método `String.fromCharCode()`:

```
var myStr:String =
  String.fromCharCode(104,101,108,108,111,32,119,111,114,108,100,33);
trace(myStr); // hello world!
```

Cada número incluido en el método `fromCharCode()` del código anterior representa un único carácter. Por ejemplo, el valor ASCII 104 representa una *h* minúscula y el valor ASCII 32 el carácter de espacio.

Puede utilizar el método `String.fromCharCode()` para convertir valores Unicode, aunque el valor Unicode debe convertirse de hexadecimal a decimal, como se muestra en el siguiente código ActionScript:

```
// Unicode 0068 == "h"
var letter:Number = Number(new Number(0x0068).toString(10));
trace(String.fromCharCode(letter)); // h
```

Puede examinar los caracteres situados en diversas posiciones de una cadena, como en este ejemplo.

### Para reproducir indefinidamente una cadena:

1. Cree un nuevo documento de Flash.
2. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

   ```
   var myStr:String = "hello world!";
   for (var i:Number = 0; i < myStr.length; i++) {
     trace(myStr.charAt(i));
   }
   ```

3. Seleccione Control > Probar película para previsualizar el documento de Flash. Debería ver cada carácter trazado en el panel Salida en una línea independiente.
4. Modifique el código ActionScript existente de forma que trace el valor ASCII de cada carácter:

   ```
   var myStr:String = "hello world!";
   for (var i:Number = 0; i < myStr.length; i++) {
     trace(myStr.charAt(i) + " - ASCII=" + myStr.charCodeAt(i));
   }
   ```

5. Guarde el documento de Flash y seleccione Control > Probar película para previsualizar el archivo SWF.

   Al ejecutar este código, aparece lo siguiente en el panel Salida:

```
h - ASCII=104
e - ASCII=101
l - ASCII=108
l - ASCII=108
o - ASCII=111
  - ASCII=32
w - ASCII=119
o - ASCII=111
r - ASCII=114
l - ASCII=108
d - ASCII=100
! - ASCII=33
```

**SUGERENCIA**

También puede dividir una cadena en una matriz de caracteres con el método String.split() e introduciendo una `cadena vacía ("")` como delimitador; por ejemplo,

```
var charArray:Array = myStr.split("");
```

Puede utilizar operadores para comparar cadenas. Para obtener información sobre la utilización de operadores con cadenas, consulte "Utilización de operadores con cadenas" en la página 150.

Puede utilizar estos operadores con sentencias condicionales, como `if` y `while`. En el siguiente ejemplo se utilizan operadores y cadenas para realizar una comparación.

### Para comparar dos cadenas:

1. Cree un nuevo documento de Flash y guárdelo como **comparestr.fla**.
2. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
var str1:String = "Apple";
var str2:String = "apple";
if (str1 < str2) {
  trace("Uppercase letters sort first.");
}
```

3. Guarde el documento de Flash y seleccione Control > Probar película para probar el archivo SWF.

Puede utilizar los operadores de igualdad (==) y desigualdad (!=) para comparar cadenas con otros tipos de objetos, como se muestra en el siguiente ejemplo.

### Para comparar cadenas con otros tipos de datos:

1. Cree un nuevo documento de Flash y guárdelo como **comparenum.fla**.
2. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
var myStr:String = "4";
var total:Number = 4;
if (myStr == total) {
  trace("Types are converted.");
}
```

3. Guarde el documento de Flash y seleccione Control > Probar película para probar el archivo SWF.

   Cuando compara dos tipos de datos distintos (como cadenas y números), Flash intenta convertirlos de forma que se pueda realizar la comparación.

Utilice los operadores de igualdad estricta (===) y desigualdad estricta (!==) para comprobar que los objetos comparados son del mismo tipo. En el siguiente ejemplo se utilizan operadores de comparación estrictos para garantizar que Flash no intenta convertir los tipos de datos mientras trata de comparar los valores.

### Para forzar comparaciones estrictas de tipos de datos:

1. Cree un nuevo documento de Flash y guárdelo como **comparestrict.fla**.
2. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
var str1:String = "4";
var str2:String = "5";
var total:Number = 4;
if (str1 !== total) {
  trace("Types are not converted.");
}
if (str1 !== str2) {
  trace("Same type, but the strings don't match.");
}
```

3. Guarde el documento de Flash y seleccione Control > Probar película.

Para más información sobre la utilización de operadores con cadenas, consulte "Utilización de operadores con cadenas" en la página 150.

Puede encontrar un archivo de origen de muestra, strings.fla, en la carpeta Samples del disco duro. En él se muestra cómo crear un procesador de textos sencillo que compara y recupera selecciones de cadenas y subcadenas.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Strings.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Strings.

## Conversión y concatenación de cadenas

Puede convertir numerosos objetos en cadenas empleando el método `toString()`. La mayoría de los objetos incorporados tienen un método `toString()` para este fin:

```
var n:Number = 0.470;
trace(typeof(n.toString())); // cadena
```

Al utilizar el operador de suma (+) con una combinación de instancias de String y distintas de String, no es necesario utilizar el método `toString()`. Para obtener detalles sobre concatenación, consulte el segundo procedimiento en esta sección.

El método `toLowerCase()` y el método `toUpperCase()` convierten los caracteres alfabéticos de la cadena a minúsculas y mayúsculas respectivamente. En el siguiente ejemplo se muestra cómo convertir una cadena de minúsculas a mayúsculas.

### Para convertir una cadena de minúsculas a mayúsculas:

1. Cree un nuevo documento de Flash y guárdelo como **convert.fla**.
2. Escriba el código siguiente en el fotograma 1 de la línea de tiempo:

```
var myStr:String = "Dr. Bob Roberts, #9.";
trace(myStr.toLowerCase()); // dr. bob roberts, #9.
trace(myStr.toUpperCase()); // DR. BOB ROBERTS, #9.
trace(myStr); // Dr. Bob Roberts, #9.
```

3. Guarde el documento de Flash y seleccione Control > Probar película.

| NOTA | Tras ejecutar estos métodos, la cadena original permanece intacta. Para transformar la cadena original, utilice lo siguiente: |
|---|---|

```
myStr = myStr.toUpperCase();
```

Al concatenar cadenas, se toman dos cadenas y se unen secuencialmente para formar una sola cadena. Por ejemplo, puede utilizar el operador de suma (+) para concatenar dos cadenas. En el siguiente ejemplo se muestra cómo concatenar dos cadenas.

### Para concatenar dos cadenas:

1. Cree un nuevo documento de Flash y guárdelo como **concat.fla**.
2. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
var str1:String = "green";
var str2:String = str1 + "ish";
trace(str2); // greenish
//
var str3:String = "yellow";
str3 += "ish";
trace(str3); // yellowish
```

   El código anterior muestra dos métodos de concatenación de cadenas. El primer método utilizar el operador de suma (+) para unir la cadena `str1` con la cadena `"ish"`. El segundo emplea el operador de suma y asignación (+=) para concatenar la cadena `"ish"` con el valor actual de `str3`.
3. Guarde el documento de Flash y seleccione Control > Probar película.

Además, puede utilizar el método `concat()` de la clase String para concatenar cadenas. Este método se muestra en el siguiente ejemplo.

### Para concatenar dos cadenas con el método concat():

1. Cree un nuevo documento de Flash y guárdelo como **concat2.fla**.
2. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
var str1:String = "Bonjour";
var str2:String = "from";
var str3:String = "Paris";
var str4:String = str1.concat(" ", str2, " ", str3);
trace(str4); // Bonjour from Paris
```

3. Seleccione Control > Probar película para probar el documento de Flash.

Si utiliza el operador de suma (+) (o el operador de suma y asignación [+=]) con una cadena y un objeto que no es una cadena, ActionScript convierte automáticamente en una cadena el objeto que no es una cadena para evaluar la expresión. Esta conversión se muestra en el siguiente ejemplo de código:

```
var version:String = "Flash Player ";
var rel:Number = 8;
version = version + rel;
trace(version); // Flash Player 8
```

No obstante, puede utilizar paréntesis para forzar al operador de suma (+) a que evalúe aritméticamente, como se muestra en el siguiente código ActionScript:

```
trace("Total: $" + 4.55 + 1.46); // Total: $4.551.46
trace("Total: $" + (4.55 + 1.46)); // Total: $6.01
```

Puede utilizar el método `split()` para crear una matriz de subcadenas de una cadena que se divide en función de un carácter delimitador. Por ejemplo, podría segmentar en varias cadenas una cadena delimitada por comas o tabulaciones.

Por ejemplo, el siguiente código muestra cómo puede dividirse una matriz en subcadenas empleando el carácter ampersand (`&`) como delimitador.

### Para crear una matriz de subcadenas segmentadas por un delimitador:

1. Cree un nuevo documento de Flash y guárdelo como **strsplit.fla**.
2. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
var queryStr:String = "first=joe&last=cheng&title=manager&startDate=3/6/
  65";
var params:Array = queryStr.split("&", 2);
trace(params); // first=joe,last=cheng
    /* params is set to an array with two elements:
       params[0] == "first=joe"
       params[1] == "last=cheng"
    */
```

3. Seleccione Control > Probar película para probar el documento de Flash.

**SUGERENCIA**

El segundo parámetro del método `split()` define el tamaño máximo de la matriz. Si no desea limitar el tamaño de la matriz creada por el método `split()`, puede omitir el segundo parámetro.

**SUGERENCIA**

La forma más sencilla de analizar una cadena de consulta (una cadena delimitada con caracteres & y =) es utilizar el método `LoadVars.decode()`, como se muestra en el siguiente código ActionScript:

```
var queryStr:String = "first=joe&last=cheng&title=manager&startDate=3/6/
  65";
var my_lv:LoadVars = new LoadVars();
my_lv.decode(queryStr);
trace(my_lv.first); // joe
```

Para más información sobre la utilización de operadores con cadenas, consulte "Utilización de operadores con cadenas" en la página 150.

Puede encontrar un archivo de origen de muestra, strings.fla, en la carpeta Samples del disco duro. En él se muestra cómo crear un procesador de textos sencillo que compara y recupera selecciones de cadenas y subcadenas.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Strings.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Strings.

## Devolución de subcadenas

Los métodos `substr()` y `substring()` de la clase String son similares. Ambos devuelven una subcadena de una cadena y ambos toman dos parámetros. En ambos métodos, el primer parámetro es la posición del carácter inicial de la cadena en cuestión. No obstante, en el método `substr()`, el segundo parámetro es la *longitud* de la subcadena que debe devolverse, mientras que en el método `substring()`, el segundo parámetro es la posición del carácter *final* de la subcadena (que no se incluye en la cadena devuelta). Este ejemplo muestra la diferencia entre estos dos métodos:

### Para encontrar una subcadena por la posición de caracteres:

1. Cree un nuevo documento de Flash y guárdelo como **substring.fla**.
2. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
var myStr:String = "Hello from Paris, Texas!!!";
trace(myStr.substr(11,15)); // Paris, Texas!!!
trace(myStr.substring(11,15)); // Pari
```

   El primer método, `substr()`, devuelve una cadena de 15 caracteres de longitud que comienza por el carácter 11. El segundo método, `substring()`, devuelve una cadena de 4 caracteres de longitud captando todos los caracteres entre el índice 11 y 15.

3. Añada el siguiente código ActionScript debajo del código creado en el paso anterior:

```
trace(myStr.slice(11, 15)); // Pari
trace(myStr.slice(-3, -1)); // !!
trace(myStr.slice(-3, 26)); // !!!
trace(myStr.slice(-3, myStr.length)); // !!!
trace(myStr.slice(-8, -3)); // Texas
```

   El método `slice()` funciona de forma muy similar al método `substring()`. Cuando se le facilitan como parámetros dos enteros no negativos, funciona exactamente de la misma forma. Sin embargo, `slice()` puede tomar enteros negativos como parámetros.

4. Seleccione Control > Probar película para probar el documento de Flash.

| NOTA | Puede combinar enteros positivos y negativos como parámetros del método `slice()`. |
|---|---|

Puede utilizar los métodos `indexOf()` y `lastIndexOf()` para localizar subcadenas coincidentes dentro de una cadena, como se muestra en el siguiente ejemplo.

### Para localizar la posición de carácter de una subcadena coincidente:

1. Cree un nuevo documento de Flash y guárdelo como **indexof.fla**.
2. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
var myStr:String = "The moon, the stars, the sea, the land";
trace(myStr.indexOf("the")); // 10
trace(myStr.indexOf("the", 11)); // 21
```

   El primer índice de la palabra *the* comienza en el carácter 10 dado que el método `indexOf()` distingue entre mayúsculas y minúsculas; por tanto, la primera instancia de *The* no se tiene en cuenta. También puede especificar un segundo parámetro al método `indexOf()` para indicar la posición de índice de la cadena desde la que debe iniciarse la búsqueda. En el código anterior, Flash busca el primer índice de la palabra *the* que aparece detrás del carácter 11.

3. Añada el siguiente código ActionScript debajo del código creado en el paso anterior:

```
trace(myStr.lastIndexOf("the")); // 30
trace(myStr.lastIndexOf("the", 29)); // 21
```

   El método `lastIndexOf()` localiza la última vez que aparece una subcadena en la cadena. Por ejemplo, en lugar de buscar un carácter o una subcadena desde el principio de una cadena, `lastIndexOf()` inicia la búsqueda desde el final de una cadena hacia atrás. De la misma forma que ocurre con el método `indexOf()`, si incluye un segundo parámetro con el método `lastIndexOf()`, la búsqueda se realiza desde dicha posición de índice aunque hacia atrás (de derecha a izquierda):

4. Seleccione Control > Probar película para probar el documento de Flash.

| SUGERENCIA | Los métodos `indexOf()` y `lastIndexOf()` distinguen entre mayúsculas y minúsculas. |
|---|---|

Puede encontrar un archivo de origen de muestra, strings.fla, en la carpeta Samples del disco duro. En él se muestra cómo crear un procesador de textos sencillo que compara y recupera selecciones de cadenas y subcadenas.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Strings.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Strings.

CAPÍTULO 13

# Animaciones, filtros y dibujos 13

En este capítulo se describe cómo añadir animación a las aplicaciones de Macromedia Flash Basic 8 y Macromedia Flash Professional 8 mediante código ActionScript en lugar de (o además de) animaciones basadas en la línea de tiempo que utilizan interpolaciones de movimiento y de forma. El uso de código para crear animaciones y efectos a menudo reduce el tamaño de archivo de la aplicación terminada y también puede mejorar el rendimiento y la coherencia de la propia animación. En ocasiones, las animaciones basadas en ActionScript incluso pueden llegar a reducir el volumen de trabajo: el código se escribe con mayor rapidez y resulta más sencillo de aplicarse a muchas instancias al mismo tiempo o de reutilizarse en otras aplicaciones. En este capítulo también se muestra cómo crear animaciones utilizando conceptos básicos fundamentales de ActionScript, las clases Tween y TransitionManager, la interfaz API de dibujo, las clases de filtro y los modos de mezcla.

Puede utilizar la interfaz API de dibujo, que consta de los métodos de dibujo de la clase MovieClip, para añadir animaciones y dibujos. Estos métodos permiten emplear código para crear líneas, rellenos y formas, en lugar de herramientas de dibujo en la herramienta de edición.

Los filtros y otros efectos expresivos también son importantes en muchas aplicaciones de Flash, ya que permiten aplicar rápidamente efectos y animarlos. Puede utilizar código para añadir y animar efectos de filtro, modos de mezcla e imágenes de mapa de bits.

Este capítulo contiene las siguientes secciones, que describen el uso de código ActionScript para crear animaciones y añadir efectos, así como la utilización de la interfaz API de dibujo para dibujar en ActionScript:

# Creación de scripts para animaciones con ActionScript 2.0

ActionScript 2.0 permite añadir animaciones a las aplicaciones de Flash, en lugar de utilizar interpolaciones de movimiento y de forma en una línea de tiempo. En las siguientes secciones se muestra cómo utilizar código para animar instancias, por ejemplo, para cambiar su transparencia y su aspecto o mover la instancia por el escenario.

Para obtener información sobre el uso de las clases Tween y TransitionManager para automatizar más las animaciones basadas en código, consulte "Clase TransitionManager" en la página 1281 y "Clase Tween" en la página 1357. Estas clases permiten añadir animaciones de transición y ecuaciones de suavizado (aceleración) avanzadas a las instancias de clips de película en la aplicación. Muchos de estos efectos son difíciles de recrear mediante código ActionScript si las clases no se han creado previamente, ya que el código necesario implica escribir ecuaciones matemáticas complejas para lograr el efecto deseado.

Para más información sobre cómo animar dibujos creados con código, consulte "Dibujo con código ActionScript" en la página 574.

En las secciones siguientes se describe cómo crear scripts de animaciones:

- "Animaciones y velocidad de fotogramas" en la página 503
- "Aparición o desaparición progresiva de objetos mediante código" en la página 504
- "Adición de efectos de color y brillo mediante código" en la página 507
- "Desplazamiento de objetos mediante código" en la página 510
- "Desplazamiento lateral de una imagen mediante código" en la página 511

Para ver un ejemplo de una animación creada mediante script en Flash, puede encontrar un archivo de origen de muestra, animation.fla, en la carpeta Samples del disco duro:

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Animation.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Animation.

También puede encontrar aplicaciones de galerías de fotos de muestra en el disco duro. Estos archivos proporcionan ejemplos de cómo emplear ActionScript para controlar los clips de película dinámicamente mientras se cargan archivos de imagen en un archivo SWF, que incluye la animación creada mediante script. Los archivos de origen de ejemplo se denominan gallery_tree.fla y gallery_tween.fla y se encuentran en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Galleries.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Galleries.

## Animaciones y velocidad de fotogramas

Cuando añada animaciones a una aplicación, tenga en cuenta la velocidad de fotogramas a la que establece el archivo FLA. Debe considerarla dado que puede afectar al rendimiento del archivo SWF y del equipo en el que se reproduce. Si la establece en un valor muy alto pueden darse problemas en el procesador, sobre todo cuando se emplean muchos elementos o se utiliza código ActionScript para crear la animación.

Sin embargo, también debe tener en cuenta la configuración de la velocidad de fotogramas, ya que afecta a la suavidad con la que se reproduce la animación. Por ejemplo, una animación establecida a 12 fotogramas por segundo (fps) en el inspector de propiedades reproduce 12 fotogramas cada segundo. Si la velocidad de fotogramas del documento se establece en 24 fps, la animación parece reproducirse con mayor suavidad que si se ejecuta a 12 fps. No obstante, la animación a 24 fps también se reproduce mucho más rápidamente que a 12 fps, por lo que la duración total (en segundos) es más corta. Por tanto, si precisa crear una animación de 5 segundos con una velocidad de fotogramas mayor, debe añadir fotogramas adicionales para rellenar esos cinco segundos en comparación con una velocidad de fotogramas menor (lo que aumenta el tamaño de archivo total de la animación). Una animación de 5 segundos a 24 fps normalmente presenta un tamaño de archivo mayor que una animación de 5 segundos a 12 fps.

**NOTA**

Cuando utiliza un controlador de eventos `onEnterFrame` para crear animaciones mediante scripts, la animación se ejecuta a la velocidad de fotogramas del documento, de forma similar a si hubiera creado una interpolación de movimiento en una línea de tiempo. Una alternativa al controlador de eventos `onEnterFrame` es `setInterval` (consulte %{setInterval function}% en Referencia del lenguaje ActionScript 2.0). En lugar de depender de la velocidad de fotogramas, se llama a las funciones a un intervalo especificado. Al igual que `onEnterFrame`, cuanto más frecuentemente se utilice `setInterval` para llamar a una función, mayor número de recursos del procesador utilizará la animación.

Utilice la menor velocidad de fotogramas posible que reproduzca la animación con suavidad durante la ejecución; esto contribuirá a reducir la carga en el procesador del usuario final. Intente no emplear una velocidad superior a 30 ó 40 fps; las velocidades mayores generan mucha carga en los procesadores y no cambian demasiado o en absoluto la apariencia de las animaciones durante la ejecución.

Asimismo, sobre todo si trabaja con animaciones basadas en la línea de tiempo, seleccione una velocidad de fotogramas lo antes posible en el proceso de desarrollo. Cuando pruebe el archivo SWF, compruebe la duración y el tamaño de archivo SWF de la animación. La velocidad de fotogramas afecta enormemente a la velocidad de la animación.

## Aparición o desaparición progresiva de objetos mediante código

Al trabajar con clips de película en el escenario, puede que desee que el clip de película aparezca o desaparezca progresivamente en lugar de activar o desactivar su propiedad `_visible`. En el siguiente procedimiento se muestra cómo utilizar un controlador de eventos `onEnterFrame` para animar un clip de película.

### Para hacer aparecer o desaparecer progresivamente un clip de película mediante código:

1. Cree un nuevo documento de Flash denominado **fade1.fla**.
2. Dibuje algunos gráficos en el escenario con las herramientas de dibujo o importe una imagen al escenario (Archivo > Importar > Importar a escenario).
3. Seleccione el contenido en el escenario y luego Modificar > Convertir en símbolo.
4. Seleccione la opción Clip de película y haga clic en Aceptar para crear el símbolo.
5. Seleccione la instancia de clip de película en el escenario y escriba **img1_mc** en el cuadro de texto Nombre de instancia del inspector de propiedades.

6. Seleccione el fotograma 1 de la línea de tiempo y añada el siguiente código ActionScript en el panel Acciones:

```
img1_mc.onEnterFrame = function() {
  img1_mc._alpha -= 5;
  if (img1_mc._alpha <= 0) {
    img1_mc._visible = false;
    delete img1_mc.onEnterFrame;
  }
};
```

   Este código utiliza un controlador de eventos `onEnterFrame` que se invoca repetidamente a la velocidad de fotogramas del archivo SWF. El número de veces por segundo que se llama al controlador de eventos depende de la velocidad de fotogramas establecida para el documento de Flash. Si la velocidad de fotogramas es de 12 fotogramas por segundo (fps), se llamará al controlador de eventos `onEnterFrame` 12 veces por segundo. Del mismo modo, si la velocidad de fotogramas del documento de Flash fuese de 30 fps, se invocaría el controlador de eventos 30 veces por segundo.

7. Seleccione Control > Probar película para probar el documento.

   El clip de película añadido al escenario desaparece lentamente.

También puede modificar la propiedad `_alpha` utilizando la función `setInterval()` en lugar de un controlador de eventos `onEnterFrame`, como se muestra en el siguiente procedimiento:

### Para hacer desaparecer progresivamente un objeto con la función setInterval():

1. Cree un nuevo documento de Flash denominado **fade2.fla**.
2. Dibuje algunos gráficos en el escenario o importe una imagen al escenario (Archivo > Importar > Importar a escenario).
3. Seleccione el contenido en el escenario y luego Modificar > Convertir en símbolo.
4. Seleccione la opción Clip de película y haga clic en Aceptar para crear el símbolo.
5. Seleccione la instancia de clip de película en el escenario y escriba **img1_mc** en el cuadro de texto Nombre de instancia del inspector de propiedades.

6. Seleccione el fotograma 1 de la línea de tiempo y añada el siguiente código en el panel Acciones:

```
var alpha_interval:Number = setInterval(fadeImage, 50, img1_mc);
function fadeImage(target_mc:MovieClip):Void {
  target_mc._alpha -= 5;
  if (target_mc._alpha <= 0) {
    target_mc._visible = false;
    clearInterval(alpha_interval);
  }
}
```

   La función `setInterval()` se comporta de forma ligeramente distinta al controlador de eventos `onEnterFrame`, ya que `setInterval()` indica a Flash la frecuencia con la que el código debe llamar a una función concreta. En este ejemplo de código, se llama a la función `fadeImage()` definida por el usuario cada 50 milisegundos (20 veces por segundo). La función `fadeImage()` reduce el valor de la propiedad `_alpha` del clip de película actual. Cuando el valor de `_alpha` es igual o menor que 0, el intervalo se borra, lo que ocasiona que la función `fadeImage()` deje de ejecutarse.

7. Seleccione Control > Probar película para probar el documento.

   El clip de película añadido al escenario desaparece lentamente.

Para más información sobre funciones definidas por el usuario, consulte "Definición de funciones globales y de línea de tiempo" en la página 180. Para más información sobre el controlador de eventos `onEnterFrame`, consulte `%{onEnterFrame (MovieClip.onEnterFrame handler)}%` en *Referencia del lenguaje ActionScript de Flash 2.0*. Para más información sobre la función `setInterval()`, consulte `%{setInterval function}%` en *Referencia del lenguaje ActionScript 2.0*.

Para ver un ejemplo de una animación creada mediante script en Flash, puede encontrar un archivo de origen de muestra, animation.fla, en la carpeta Samples del disco duro:

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Animation.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Animation.

# Adición de efectos de color y brillo mediante código

Además de utilizar ActionScript para establecer y aplicar efectos de desvanecimiento alfa (consulte "Aparición o desaparición progresiva de objetos mediante código" en la página 504), puede crear animaciones con distintos efectos de color y brillo mediante código, en lugar de emplear el panel Filtros del inspector de propiedades.

El siguiente procedimiento carga una imagen JPEG y aplica un filtro de transformación de color que modifica los canales rojo y verde a medida que el puntero del ratón se mueve por los ejes *x* e *y*.

### Para cambiar los canales de color de un objeto mediante código ActionScript:

1. Cree un nuevo documento de Flash denominado **colorTrans.fla**.
2. Seleccione el fotograma 1 de la línea de tiempo y añada el siguiente código ActionScript en el panel Acciones:

```
import flash.geom.Transform;
import flash.geom.ColorTransform;

var imageClip:MovieClip = this.createEmptyMovieClip("imageClip", 1);
var clipLoader:MovieClipLoader = new MovieClipLoader();
clipLoader.loadClip("http://www.helpexamples.com/flash/images/
  image1.jpg", imageClip);

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  var transformer:Transform = new Transform(imageClip);
  var colorTransformer:ColorTransform = transformer.colorTransform;
  colorTransformer.redMultiplier = (_xmouse / Stage.width) * 1;
  colorTransformer.greenMultiplier = (_ymouse / Stage.height) * 1;
  transformer.colorTransform = colorTransformer;
}
Mouse.addListener(mouseListener);
```

3. Seleccione Control > Probar película para probar el documento y después mueva el puntero del ratón por el escenario.

   El archivo de imagen que se carga transforma los colores a medida que se mueve el ratón.

También puede utilizar la clase ColorMatrixFilter para convertir una imagen en color a blanco y negro, como muestra el siguiente procedimiento:

### Para utilizar la clase ColorMatrixFilter para cambiar una imagen a imagen en escala de grises:

1. Cree un nuevo documento de Flash denominado **grayscale.fla**.
2. Seleccione el fotograma 1 de la línea de tiempo y añada el siguiente código ActionScript en el panel Acciones:

```
import flash.filters.ColorMatrixFilter;
System.security.allowDomain("http://www.helpexamples.com");
var mcl_obj:Object = new Object();
mcl_obj.onLoadInit = function(target_mc:MovieClip):Void {
  var myElements_array:Array = [0.3, 0.59, 0.11, 0, 0,
       0.3, 0.59, 0.11, 0, 0,
       0.3, 0.59, 0.11, 0, 0,
       0, 0, 0, 1, 0];
  var myColorMatrix_filter:ColorMatrixFilter = new
  ColorMatrixFilter(myElements_array);
  target_mc.filters = [myColorMatrix_filter];
}
this.createEmptyMovieClip("img_mc", this.getNextHighestDepth());
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mcl_obj);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img_mc);
```

   El código anterior importa primero la clase ColorMatrixFilter y crea un objeto detector que se empleará con la nueva instancia de MovieClipLoader creada en código posterior. A continuación, se crea una nueva instancia de clip de película denominada `img_mc`, así como una instancia de cargador de clip de película denominada `img_mcl`. Por último, el clip de película de origen se carga en el clip `img_mc` en el escenario. Cuando la imagen se ha cargado correctamente, se llama al controlador de eventos onLoadInit, que asocia un filtro ColorMatrixFilter a la imagen cargada.

3. Seleccione Control > Probar película para probar el documento.

   La imagen cargada en el escenario cambia a imagen en escala de grises. Compruebe la imagen en línea (`http://www.helpexamples.com/flash/images/image1.jpg`) para ver su color original.

También puede establecer el brillo de una imagen mediante código ActionScript en el siguiente procedimiento.

### Para cambiar el brillo de una imagen:

1. Cree un nuevo documento de Flash denominado **brightness.fla**.
2. Seleccione el fotograma 1 de la línea de tiempo y añada el siguiente código en el panel Acciones:

```
import flash.filters.ColorMatrixFilter;
System.security.allowDomain("http://www.helpexamples.com/");
var mcl_obj:Object = new Object();
mcl_obj.onLoadInit = function(target_mc:MovieClip):Void {
  var myElements_array:Array = [1, 0, 0, 0, 100,
       0, 1, 0, 0, 100,
       0, 0, 1, 0, 100,
       0, 0, 0, 1, 0];
  var myColorMatrix_filter:ColorMatrixFilter = new
  ColorMatrixFilter(myElements_array);
  target_mc.filters = [myColorMatrix_filter];
}
this.createEmptyMovieClip("img_mc", this.getNextHighestDepth());
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mcl_obj);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image2.jpg",
  img_mc);
```

   Este bloque de código emplea la clase MovieClipLoader para cargar un archivo JPEG externo. Cuando la imagen se ha cargado correctamente, se llama al controlador de eventos onLoadInit de la clase MovieClipLoader y se modifica el brillo de la imagen a 100 mediante el filtro ColorMatrixFilter.

3. Seleccione Control > Probar película para probar el documento.

   La imagen cargada en el archivo SWF cambia su brillo al probar el archivo SWF. Compruebe la imagen en línea (`http://www.helpexamples.com/flash/images/image2.jpg`) para ver su apariencia original.

Para ver un ejemplo de una animación creada mediante script en Flash, puede encontrar un archivo de origen de muestra, animation.fla, en la carpeta Samples del disco duro:

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Animation.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Animation.

También puede encontrar aplicaciones de galerías de fotos de muestra en el disco duro. Estos archivos proporcionan ejemplos de cómo emplear ActionScript para controlar los clips de película dinámicamente mientras se cargan archivos de imagen en un archivo SWF, que incluye la animación creada mediante script. Los archivos de origen de ejemplo se denominan gallery_tree.fla y gallery_tween.fla y se encuentran en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Galleries.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Galleries.

## Desplazamiento de objetos mediante código

El desplazamiento de un objeto mediante código ActionScript es similar a la modificación de la propiedad `_alpha` de un objeto, con la diferencia de que en este caso se modifica la propiedad `_x` o `_y` del mismo.

En el siguiente procedimiento, se aplica animación a una imagen JPEG cargada dinámicamente, lo que provoca que ésta se deslice horizontalmente por el escenario.

### Para mover una instancia en el escenario mediante código:

1. Cree un nuevo documento de Flash denominado **moveClip.fla**.
2. Cambie la velocidad de fotogramas del documento a **24** fps en el inspector de propiedades.

   La animación se ejecuta con mucha mayor suavidad si se emplea una velocidad de fotogramas superior, por ejemplo, 24 fps.
3. Seleccione el fotograma 1 de la línea de tiempo y añada el siguiente código ActionScript en el panel Acciones:

```
// Crear una instancia de clip de película.
this.createEmptyMovieClip("img1_mc", 10);
var mcl_obj:Object = new Object();
mcl_obj.onLoadInit = function (target_mc:MovieClip):Void {
  target_mc._x = Stage.width;
  target_mc.onEnterFrame = function() {
    target_mc._x -= 3; // disminuir posición _x actual en 3 píxeles
    if (target_mc._x <= 0) {
      target_mc._x = 0;
      delete target_mc.onEnterFrame;
    }
  };
};
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mcl_obj);
// Cargar una imagen en el clip de película
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img1_mc);
```

   Este ejemplo de código carga una imagen externa de un servidor Web remoto y, una vez que la imagen se ha cargado completamente, se le aplica animación para que se desplace horizontalmente por el escenario. En lugar de utilizar el controlador de eventos `onEnterFrame`, podría utilizarse la función `setInterval()` para aplicar animación a la imagen.

4. Seleccione Control > Probar película para probar el documento.

   La imagen se carga y se anima desde la derecha del escenario a su esquina superior izquierda.

Para obtener información sobre el uso del controlador de eventos `onEnterFrame` o la función `setInterval()` para aplicar animación a una imagen, consulte "Aparición o desaparición progresiva de objetos mediante código" en la página 504.

Para ver un ejemplo de una animación creada mediante script en Flash, puede encontrar un archivo de origen de muestra, animation.fla, en la carpeta Samples del disco duro:

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Animation.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Animation.

También puede encontrar aplicaciones de galerías de fotos de muestra en el disco duro. Estos archivos proporcionan ejemplos de cómo emplear ActionScript para controlar los clips de película dinámicamente mientras se cargan archivos de imagen en un archivo SWF, que incluye la animación creada mediante script. Los archivos de origen de ejemplo se denominan gallery_tree.fla y gallery_tween.fla y se encuentran en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Galleries.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Galleries.

## Desplazamiento lateral de una imagen mediante código

Con código ActionScript, puede desplazar lateralmente con facilidad imágenes de gran tamaño en documentos de Flash. Resulta útil cuando la imagen no cabe en el escenario o si desea crear un efecto de animación en el que se desplace lateralmente un clip de película de un lado del escenario al otro. Por ejemplo, si tiene una imagen panorámica mayor que el escenario y no desea reducir las dimensiones de la imagen ni aumentar las del escenario, puede crear un clip de película que actúe a modo de máscara para la imagen de mayor tamaño.

El siguiente procedimiento muestra cómo aplicar dinámicamente una máscara a un clip de película y utilizar un controlador de eventos `onEnterFrame` para animar una imagen situada detrás de la máscara.

### Para desplazar lateralmente una instancia en el escenario mediante código:

1. Cree un nuevo documento de Flash denominado **pan.fla**.
2. Cambie la velocidad de fotogramas del documento a **24** fps en el inspector de propiedades.

   La animación se ejecuta con mucha mayor suavidad si se emplea una velocidad de fotogramas superior, por ejemplo, 24 fps.
3. Seleccione el fotograma 1 de la línea de tiempo y añada el siguiente código ActionScript en el panel Acciones:

```
System.security.allowDomain("http://www.helpexamples.com/");
// inicializar variables
var direction:Number = -1;
var speed:Number = 5;
// crear clip para cargar la imagen
this.createEmptyMovieClip("img_mc", 10);
// crear un clip para utilizarlo como máscara
this.createEmptyMovieClip("mask_mc", 20);
// utilizar la interfaz API de dibujo para dibujar/crear una máscara
with (mask_mc) {
  beginFill(0xFF0000, 0);
  moveTo(0, 0);
  lineTo(300, 0);
  lineTo(300, 100);
  lineTo(0, 100);
  lineTo(0, 0);
  endFill();
}

var mcl_obj:Object = new Object();
mcl_obj.onLoadInit = function(target_mc:MovieClip) {
  // establecer la máscara del clip de película de destino como mask_mc
  target_mc.setMask(mask_mc);
  target_mc.onEnterFrame = function() {
    target_mc._x += speed * direction;
    // si target_mc está en un borde, invertir la dirección de animación
    if ((target_mc._x <= -(target_mc._width-mask_mc._width)) ||
  (target_mc._x >= 0)) {
      direction *= -1;
    }
  };
};

var my_mcl:MovieClipLoader = new MovieClipLoader();
my_mcl.addListener(mcl_obj);
my_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img_mc);
```

La primera sección del código en este ejemplo define dos variables: `direction` y `speed`. La variable `direction` controla si la imagen enmascarada debe desplazarse de izquierda a derecha (`1`) o de derecha a izquierda (`-1`). La variable `speed` controla el número de píxeles que deben desplazarse cada vez que se llama al controlador de eventos `onEnterFrame`. Cuanto mayor es el número, mayor es la velocidad de desplazamiento de la animación, aunque ello conlleva que la animación pierda suavidad.

En la siguiente sección del código se crean dos clips de película vacíos: `img_mc` y `mask_mc`. Se dibuja un rectángulo de 300 por 100 píxeles dentro del clip de película `mark_mc` empleando la interfaz API de dibujo. A continuación, se crea un nuevo objeto (`mcl_obj`) que se empleará como detector de una instancia de MovieClipLoader creada en el bloque de código final. Este objeto define un detector para el evento `onLoadInit`, que enmascara la imagen cargada dinámicamente y establece la animación de desplazamiento. Una vez que la imagen se encuentra en el borde izquierdo o derecho de la máscara, se invierte la animación.

El bloque de código final define una instancia de MovieClipLoader, especifica el objeto detector creado anteriormente y comienza a cargar el archivo de imagen JPEG en el clip de película `img_mc`.

4. Seleccione Control > Probar película para probar el documento.

   La imagen se carga y se anima hacia atrás y hacia adelante con un movimiento de desplazamiento lateral (de lado a lado). La imagen se enmascara durante la ejecución. Para ver la imagen original, compruébela en línea (`http://www.helpexamples.com/flash/images/image1.jpg`).

# Caché de mapa de bits, desplazamiento y rendimiento

Flash Player 8 introduce la caché de mapa de bits, que permite mejorar el rendimiento de los clips de película sin cambios en las aplicaciones. Cuando se establecen las propiedades `MovieClip.cacheAsBitmap` o `Button.cacheAsBitmap` en `true`, Flash Player almacena en caché una representación de mapa de bits interno de la instancia de clip de película o de botón. Esto puede mejorar el rendimiento de los clips de película que tienen contenido vectorial complejo. Todos los datos vectoriales de un clip de película con un mapa de bits en caché se dibujan en el mapa de bits, no en el escenario principal.

| NOTA | El mapa de bits se copia en el escenario principal como píxeles no expandidos ni rotados, pero ajustados a los límites del píxel más cercano. Los píxeles se asignan de 1 a 1 con el objeto principal. Si los límites del mapa de bits cambian, el mapa de bits se vuelve a crear en lugar de expandirse. |
|---|---|

Para obtener información detallada del almacenamiento en caché de instancias de botón o clip de película, consulte las secciones siguientes en el Capítulo 11, "Trabajo con clips de película":

- "Asignación de caché y desplazamiento de clips de película con ActionScript" en la página 393
- "Asignación de caché para un clip de película" en la página 398
- "Definición del fondo de un clip de película" en la página 401

La propiedad `cacheAsBitmap` es idónea con clips de película que tienen sobre todo contenido estático y no cambian de escala ni giran con frecuencia. Con estos clips, la propiedad `cacheAsBitmap` puede mejorar el rendimiento cuando se convierte el clip de película (cuando se modifican sus posiciones *x* e *y*). Para obtener información detallada sobre el uso de esta función, consulte "Cuándo es conveniente activar la caché" en la página 396.

Puede encontrar un archivo de origen de ejemplo que muestra cómo aplicar la caché de mapa de bits a una instancia. El archivo se denomina cacheBitmap.fla y se encuentra en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\CacheBitmap.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/CacheBitmap.

También puede encontrar un archivo de origen de ejemplo que muestra cómo aplicar la caché de mapa de bits al texto desplazable. El archivo de origen de ejemplo se denomina flashtype.fla y se encuentra en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\FlashType.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/FlashType.

# Las clases Tween y TransitionManager

Al instalar Flash Basic 8 o Flash Professional 8, también se instalan dos clases muy potentes: las clases Tween y TransitionManager. En esta sección se describe cómo utilizar estas clases con clips de película y componentes de la versión 2 de Macromedia (incluida con Flash MX 2004 y Flash 8) para añadir animación fácilmente a los archivos SWF.

Si crea una presentación de diapositivas o una aplicación de formulario con Flash Professional 8 (sólo ActionScript 2.0), podrá seleccionar comportamientos que añaden diferentes tipos de transiciones entre diapositivas, algo similar a la creación de una presentación de PowerPoint. Esta funcionalidad puede añadirse a una aplicación de pantallas mediante las clases Tween y TransitionManager, que generan código ActionScript que aplica animación a las pantallas en función del comportamiento que elija.

También puede utilizar las clases Tween y TransitionManager fuera de un documento basado en pantalla, ya sea en Flash Basic 8 o Flash Professional 8. Por ejemplo, puede utilizar las clases con el conjunto de componentes de la versión 2 de la arquitectura de componentes de Macromedia o con clips de película. Si desea cambiar la forma en que se aplica animación a un componente ComboBox, puede utilizar la clase TransitionManager para añadir *suavizado* al abrirse el menú. El suavizado es la aceleración o desaceleración gradual durante una animación, que hace que las animaciones parezcan más realistas. También puede utilizar las clases Tween y TransitionManager en lugar de crear interpolaciones de movimiento en la línea de tiempo o escribir código personalizado para crear su propio sistema de menús con animación.

| NOTA | Las clases Tween y TransitionManager están disponibles sólo en ActionScript 2.0, pero lo están tanto en Flash Basic 8 como Flash Professional 8. |
|---|---|

Para obtener información sobre cada método y propiedad de la clase Tween, consulte el Capítulo 51, “Clase Tween” en *Referencia del lenguaje de componentes*. Para obtener información sobre cada método y propiedad de la clase TransitionManager, consulte el Capítulo 48, “Clase TransitionManager” en *Referencia del lenguaje de componentes*. Para obtener información sobre cómo trabajar con paquetes, consulte “Utilización de paquetes de filtros” en la página 534.

Puede encontrar un archivo de origen de ejemplo que utiliza estas clases para añadir animación mediante scripts. Se denomina tweenProgress.fla y se encuentra en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Tween ProgressBar.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Tween ProgressBar.

Para más información sobre las clases Tween y TransitionManager, consulte los siguientes temas:



# Adición de interpolaciones y transiciones a un archivo en Flash Professional 8 (Sólo en Flash Professional 8)

| NOTA | En esta sección se describe cómo añadir interpolaciones y transiciones a una presentación de diapositivas de Flash Professional para demostrar su apariencia para los usuarios de esta aplicación. No obstante, puede añadir interpolaciones y transiciones a las aplicaciones de Flash Basic 8 (o Flash Professional 8) mediante código. En las siguientes secciones se incluyen ejemplos que muestran cómo hacerlo. |
|---|---|

Las clases Tween y TransitionManager están diseñadas para permitirle añadir animaciones en partes del archivo SWF empleando código ActionScript sencillo. El entorno de edición de Flash contiene comportamientos que le permiten utilizar estas clases predefinidas para transiciones en aplicaciones basadas en pantallas. Para crear una presentación de diapositivas o una aplicación de formulario, puede seleccionar comportamientos que añadan distintos tipos de transiciones entre diapositivas.

Antes de comenzar a utilizar estas transiciones con clips de película en Flash, deberá comprobar lo que hacen cuando se utilizan en una aplicación basada en pantallas.

### Para ver el código ActionScript que crea una transición en una presentación de diapositivas:

1. Seleccione Archivo > Nuevo para crear una nueva presentación de diapositivas en Flash Professional 8.
2. Seleccione Presentación de Flash de la ficha General y haga clic en Aceptar.
3. Seleccione Ventana > Comportamientos para abrir el panel Comportamientos.
4. Haga clic en Añadir comportamiento (+).
5. Seleccione Pantalla > Transición del menú emergente para abrir el cuadro de diálogo Transiciones.

6. Seleccione la transición Zoom.
7. Escriba **1** en el cuadro de texto Duración.
8. Seleccione Con rebote del menú emergente Aceleración.
9. Haga clic en Aceptar para guardar la configuración y cerrar el cuadro de diálogo.

   Esto añade alrededor de 15 líneas de código ActionScript a la diapositiva. El siguiente fragmento de código muestra el código de transición relevante:

```
mx.transitions.TransitionManager.start(eventObj.target,
  {type:mx.transitions.Zoom, direction:0, duration:1,
  easing:mx.transitions.easing.Bounce.easeOut, param1:empty,
  param2:empty});
```

   Este código llama a la clase TransitionManager y luego aplica la transición Zoom con el método de suavizado `mx.transitions.easing.Bounce.easeOut` especificado. En este caso, la transición se aplica a la diapositiva seleccionada. Para aplicar este efecto a un clip de película, puede modificar el código ActionScript que debe utilizarse en las animaciones de Flash. La modificación del código para que funcione con un símbolo de clip de película es sencilla: cambie el primer parámetro de `eventObj.target` al nombre de instancia de clip de película deseado.

Flash se suministra con diez transiciones que puede personalizar utilizando los métodos de suavizado y diversos parámetros opcionales. Recuerde que el suavizado es la aceleración o desaceleración gradual durante una animación, que hace que las animaciones parezcan más realistas. Por ejemplo, una bola puede aumentar gradualmente su velocidad hacia el comienzo de una animación e ir reduciendo la velocidad antes de detenerse por completo al finalizar la animación. Existen numerosas ecuaciones para esta aceleración y desaceleración que cambian el suavizado de la animación.

En la tabla siguiente se describen las transiciones incluidas en Flash Basic 8 (mediante código) y Flash Professional 8 (mediante código o comportamientos):

| Transición | Descripción |
|---|---|
| Iris | Muestra la pantalla o el clip de película mediante una máscara animada que se acerca. |
| Barrido | Muestra la pantalla o el clip de película mediante una máscara animada que se mueve horizontalmente. |
| Disolución de píxeles | Enmascara la pantalla o el clip de película empleando rectángulos que desaparecen o aparecen. |
| Secciones | Muestra la siguiente pantalla o clip de película empleando rectángulos que desaparecen o aparecen. |

| Transición | Descripción |
|---|---|
| Desvanecimiento | Hace aparecer o desaparecer progresivamente la pantalla o el clip de película. |
| Deslizamiento | Desliza la pantalla o el clip de película desde una dirección en particular. |
| Zoom | Acerca o aleja la pantalla o el clip de película. |
| Compresión | Aplica escala horizontal o vertical a la pantalla o el clip de película actual. |
| Giro | Gira la pantalla o el clip de película actual. |
| Foto | Hace que la pantalla o el clip de película aparezca como un flash fotográfico. |

Cada transición cuenta con personalizaciones ligeramente distintas que puede aplicar a la animación. El cuadro de diálogo Transiciones ofrece una vista previa de muestra de la animación antes de aplicar el efecto a la diapositiva o el formulario.

SUGERENCIA

Para previsualizar el funcionamiento de cada transición con los distintos métodos en las clases de suavizado, puede hacer doble clic en Transition.swf, que se encuentra en la carpeta *unidad de inicio*\Archivos de programa\Macromedia\Flash 8*idioma*\First Run\Behaviors\ o en *Disco duro de Macintosh*:Applications:Macromedia Flash 8:First Run:Behaviors: para abrir el archivo SWF en el reproductor autónomo.

## Animación con las clases TransitionManager y Tween

Puede utilizar las clases TransitionManager y Tween en Flash Basic 8 y Flash Professional 8 para añadir animaciones a clips de película, componentes y fotogramas mediante código ActionScript. Si no utiliza las clases TransitionManager o Tween, tendrá que escribir código personalizado para animar los clips de película o modificar su nivel de transparencia (alfa) y sus coordenadas (ubicación). Si añade suavizado a la animación, el código ActionScript (y las matemáticas) pueden alcanzar rápidamente gran complejidad. No obstante, si desea cambiar el suavizado de una determinada animación y utiliza estas clases predefinidas, podrá seleccionar una clase diferente en lugar de intentar averiguar las nuevas ecuaciones matemáticas que necesita para crear una animación suave.

En el siguiente procedimiento se aplica animación a un clip de película para que se acerque en el escenario empleando la clase TransitionManager.

### Para animar un clip de película con la clase TransitionManager:

1. Seleccione Archivo > Nuevo y seleccione Documento de Flash.
2. Haga clic en Aceptar para crear el nuevo archivo FLA.
3. Guarde el archivo FLA como **zoom.fla**.
4. Seleccione Archivo > Importar > Importar a escenario y seleccione una imagen de la unidad de disco duro para importarla al archivo FLA.

   La imagen se importa al archivo como imagen de mapa de bits, por lo que tendrá que convertirla manualmente en un símbolo de clip de película.
5. Haga clic en Abrir para importar la imagen.
6. Seleccione la imagen importada al escenario y luego elija Modificar > Convertir en símbolo.
7. Asigne al símbolo el nombre **img1** y asegúrese de que establece el comportamiento como Clip de película.

   De manera predeterminada, el punto de registro del símbolo es la esquina superior izquierda del símbolo.
8. Haga clic en Aceptar para convertir la imagen de mapa de bits en un clip de película.
9. Con la imagen aún seleccionada, abra el inspector de propiedades (Ventana > Propiedades > Propiedades) y asigne al clip de película el nombre de instancia **img1_mc**.
10. Seleccione el fotograma 1 de la línea de tiempo principal y añada el siguiente código ActionScript al panel Acciones:

```
mx.transitions.TransitionManager.start(img1_mc,
  {type:mx.transitions.Zoom, direction:0, duration:1,
  easing:mx.transitions.easing.Bounce.easeOut, param1:empty,
  param2:empty});
```

> **NOTA** Para obtener información sobre cómo trabajar con paquetes, consulte "Utilización de paquetes de filtros" en la página 534.

11. Seleccione Control > Probar película para probar la animación.

    La imagen crece y rebota ligeramente antes de recuperar su tamaño original. Si la animación se mueve demasiado rápido, aumente la duración de la animación (en el fragmento de código anterior) de un segundo a dos o tres segundos (por ejemplo, `duration:3`).

    Puede que observe que la imagen está anclada en la esquina superior izquierda y que crece hacia la esquina inferior derecha. Este efecto es diferente a la vista previa que se observa en el cuadro de diálogo Transiciones.

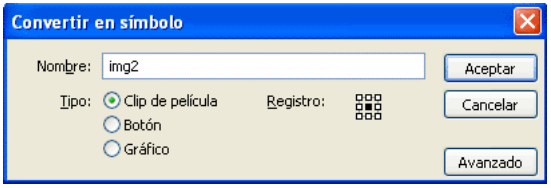
La creación de animaciones complejas es sencilla mediante las clases Tween y TransitionManager y no requiere la creación de interpolaciones de movimiento o figuras en la línea de tiempo. Y lo que es más importante aún, no tendrá que escribir ecuaciones matemáticas complejas para crear métodos de suavizado. Si desea que las imágenes se acerquen al centro en lugar de anclarse a una esquina, modifique el punto de registro del símbolo cuando convierta la imagen de un mapa de bits a un símbolo.

### Para acercar las imágenes desde el centro de la imagen:

1. Realice los pasos del procedimiento anterior.
2. Abra el archivo zoom.fla y seleccione Archivo > Guardar como para guardar una nueva copia del documento.

   Guarde el archivo como **zoom2.fla**.
3. Arrastre una copia del símbolo de mapa de bits desde el panel Biblioteca hasta el escenario, junto al símbolo de clip de película actual.
4. Con la imagen de mapa de bits todavía seleccionada en el escenario, presione F8 para convertir el símbolo en un clip de película.

   Asigne al símbolo el nombre **img2**.
5. En cuadro de diálogo Convertir en símbolo, haga clic en el centro de la cuadrícula de 3x3 para establecer el punto de registro en el centro del mapa de bits y haga clic en Aceptar.




6. Seleccione el nuevo clip de película en el escenario y asígnele el nombre de instancia **img2_mc** empleando el inspector de propiedades.
7. Seleccione el fotograma 1 de la línea de tiempo principal y añada el siguiente código ActionScript al código existente:

```
mx.transitions.TransitionManager.start(img2_mc,
  {type:mx.transitions.Zoom, direction:mx.transitions.Transition.IN,
  duration:1, easing:mx.transitions.easing.Bounce.easeOut});
```

8. Seleccione Control > Probar película para probar la animación.

El segundo clip de película crece desde el centro del símbolo en lugar de hacerlo desde la esquina.

| NOTA | Algunas transiciones se ven afectadas por el lugar en el que se sitúa el punto de registro. El cambio del punto de registro puede cambiar radicalmente el aspecto de una animación en un archivo SWF. Por ejemplo, si el punto de registro se encuentra en la esquina superior izquierda (predeterminado) cuando se emplea la transición Zoom, la transición comienza a aplicarse desde esa posición. |
|---|---|

Para obtener información sobre cada método y propiedad de la clase Tween, consulte el Capítulo 51, "Clase Tween" en *Referencia del lenguaje de componentes*. Para obtener información sobre cada método y propiedad de la clase TransitionManager, consulte el Capítulo 48, "Clase TransitionManager" en *Referencia del lenguaje de componentes*.

Puede encontrar un archivo de origen de ejemplo que utiliza estas clases para añadir animación mediante scripts. Se denomina tweenProgress.fla y se encuentra en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Tween ProgressBar.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Tween ProgressBar.

## Métodos y clases de suavizado

En "Adición de interpolaciones y transiciones a un archivo en Flash Professional 8 (Sólo en Flash Professional 8)" en la página 516 se describe cómo utilizar la clase de suavizado (aceleración) Bounce para añadir un efecto de rebote al clip de película. Además de Bounce, Flash 8 ofrece otras cinco clases de suavizado adicionales que se describen en la siguiente tabla:

| Transición | Descripción |
|---|---|
| Back | Amplía la animación más allá del rango de la transición en uno o ambos extremos una vez para asemejar un efecto de desbordamiento. |
| Bounce | Añade un efecto de rebote dentro del rango de la transición en uno o ambos extremos. El número de rebotes está en relación con la duración (a mayor duración, mayor número de rebotes). |
| Elastic | Añade un efecto elástico que está fuera del rango de la transición en uno o ambos extremos. La duración no afecta al grado de elasticidad. |
| Regular | Añade un movimiento más lento en uno o en ambos extremos. Esta función le permite añadir un efecto de aceleración, de ralentización o ambos a la vez. |

| Transición | Descripción |
|---|---|
| Strong | Añade un movimiento más lento en uno o en ambos extremos. Este efecto es similar al suavizado Regular (normal) pero más pronunciado. |
| None | Añade un movimiento uniforme de principio a fin sin efectos, ralentización ni aceleración. Esta transición también se denomina transición lineal. |

Estas seis clases de suavizado (aceleración) tienen tres métodos de suavizado que se describen en la siguiente tabla:

| Método | Descripción |
|---|---|
| `easeIn` | Proporciona el efecto de suavizado al principio de la transición. |
| `easeOut` | Proporciona el efecto de suavizado al final de la transición. |
| `easeInOut` | Proporciona el efecto de suavizado al principio y al final de la transición. |

Para abrir estas clases en Flash o en el editor de ActionScript, desplácese a la carpeta *Disco duro*\Archivos de programa\Macromedia\Flash 8*idioma*\First Run\Classes\mx\transitions\easing\ en Windows (se presupone una instalación predeterminada) o *Disco duro de Macintosh*:Applications:Macromedia Flash 8:First Run:Classes:mx:transitions:easing.

En el procedimiento de aplicar zoom a las imágenes descrito en "Animación con las clases TransitionManager y Tween" en la página 518 se utilizaban el método y la clase de suavizado mx.transitions.easing.Bounce.easeOut. En la carpeta de su disco duro, el código ActionScript hace referencia al método `easeOut()` dentro de la clase Bounce.as. Este archivo de ActionScript se encuentra en la carpeta easing.

Para obtener información sobre cada método y propiedad de la clase Tween, consulte el Capítulo 51, "Clase Tween" en *Referencia del lenguaje de componentes*. Para obtener información sobre cada método y propiedad de la clase TransitionManager, consulte el Capítulo 48, "Clase TransitionManager" en *Referencia del lenguaje de componentes*.

**SUGERENCIA**

Para previsualizar el funcionamiento de cada transición con los distintos métodos en las clases de suavizado, puede hacer doble clic en Transition.swf, que se encuentra en la carpeta *unidad de inicio*\Archivos de programa\Macromedia\Flash 8*idioma*\First Run\Behaviors\ o en *Disco duro de Macintosh*:Applications:Macromedia Flash 8:First Run:Behaviors: para abrir el archivo SWF en el reproductor autónomo.

Puede encontrar un archivo de origen de ejemplo que utiliza estas clases para añadir animación mediante scripts. Se denomina tweenProgress.fla y se encuentra en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Tween ProgressBar.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Tween ProgressBar.

## La clase Tween

La clase Tween le permite mover, cambiar el tamaño y desvanecer clip de película fácilmente en el escenario. El constructor de la clase mx.transitions.Tween tiene el siguiente nombre y tipos de parámetro:

```
function Tween(obj, prop, func, begin, finish, duration, useSeconds) {
    // código ...
}
```

`obj` El objeto de clip de película al que se refiere la instancia de Tween.

`prop` Un nombre de cadena de una propiedad de `obj` cuyos valores se interpolarán.

`func` El método de suavizado que calcula un efecto de suavizado para los valores de propiedad del objeto interpolado.

`begin` Un número que indica el valor inicial de `prop` (propiedad del objeto de destino que se interpolará).

`finish` Un número que indica el valor final de `prop` (propiedad del objeto de destino que se interpolará).

`duration` Un número que indica la duración del movimiento de interpolación. Si se omite, se establece la duración en `infinity` de forma predeterminada.

`useSeconds` Un valor booleano relacionado con el valor especificado en el parámetro `duration`, que indica que se deben utilizar segundos si se establece como `true` o fotogramas si se establece como `false`.

Supongamos, por ejemplo, que desea mover un clip de película por el escenario. Puede añadir fotogramas clave a una línea de tiempo e insertar una interpolación de movimiento o forma entre ellos, escribir código en un controlador de eventos `onEnterFrame` o utilizar la función `setInterval()` parar llamar a la función a intervalos regulares. Si utiliza la clase Tween, dispondrá de otra opción para modificar las propiedades `_x` e `_y` de un clip de película. También puede añadir los métodos de suavizado descritos anteriormente. Para aprovechar las ventanas de la clase Tween, puede utilizar el siguiente código ActionScript:

```
new mx.transitions.Tween(ball_mc, "_x",
  mx.transitions.easing.Elastic.easeOut, 0, 300, 3, true);
```

Este fragmento de código ActionScript crea una nueva instancia de la clase Tween que aplica animación al clip de película `ball_mc` en el escenario a lo largo del eje *x* (de izquierda a derecha). El clip de película se desplaza desde los 0 píxeles hasta los 300 píxeles en tres segundos y el código ActionScript aplica un método de suavizado elástico. Esto significa que la pelota llega más allá de los 300 píxeles del eje *x* antes de regresar empleando un efecto de movimiento fluido.

Puede encontrar un archivo de origen de ejemplo que utiliza estas clases para añadir animación mediante scripts. Se denomina tweenProgress.fla y se encuentra en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Tween ProgressBar.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Tween ProgressBar.

## Utilización de la clase Tween

Si utiliza la clase Tween en más de un lugar del documento de Flash, puede optar por utilizar una sentencia `import`. Esto le permite importar la clase Tween y los métodos de suavizado en lugar de proporcionar los nombres de clase completos cada vez que los utilice, como muestra el siguiente procedimiento:

### Para importar y utilizar la clase Tween:

1. Cree un documento nuevo y asígnele el nombre **easeTween.fla**.
2. Cree un clip de película en el escenario.
3. Seleccione la instancia de clip de película y, en el inspector de propiedades, escriba **ball_mc** en el cuadro de texto Nombre de instancia.

4. Seleccione el fotograma 1 de la línea de tiempo y añada el siguiente código en el panel Acciones:

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
new Tween(ball_mc, "_x", Elastic.easeOut, Stage.width, 0, 3, true);
```

Este código de ejemplo utiliza dos sentencias `import`. La primera sentencia importa sólo la clase mx.transitions.Tween, mientras que la segunda sentencia `import` utiliza el comodín (*) como atajo para importar las seis clases de suavizado empleando una única línea de código. La segunda sentencia importa un paquete de clases completo.

| NOTA | Para obtener información sobre cómo trabajar con paquetes, consulte "Utilización de paquetes de filtros" en la página 534. |
|---|---|

5. Seleccione Control > Probar película para ver la animación.

En la documentación de Flash, los *paquetes* son directorios que contienen uno o más archivos de clases y que residen en un directorio de ruta de clases determinado. En este caso, el paquete reside en la carpeta C:\Archivos de programa\Macromedia\Flash 8*idioma*\First Run\Classes\mx\transitions\easing (Windows) o en Disco duro:Applications:Macromedia Flash 8:First Run:Classes:mx:transitions:easing (Macintosh). Obviamente, importar un paquete entero es preferible a tener que importar seis clases por separado. En lugar de hacer referencia a la clase mx.transitions.Tween, el código ActionScript hace referencia directamente a la clase Tween. De la misma forma, en lugar de utilizar el nombre de clase completo para las clases de suavizado, por ejemplo, mx.transitions.easing.Elastic.easeOut, puede escribir **Elastic.easeOut** en el código ActionScript. Para más información, consulte "Utilización de paquetes de filtros" en la página 534.

Empleando un código similar, puede establecer la propiedad `_alpha` para que las instancias aparezcan o desaparezcan (se desvanezcan) progresivamente, en lugar de la propiedad `_x`, como se muestra en el siguiente procedimiento:

### Para hacer aparecer o desaparecer progresivamente instancias con la clase Tween:

1. Cree un nuevo documento y asígnele el nombre **fadeTween.fla**.
2. Cree un clip de película en el escenario.
3. Seleccione la instancia de clip de película y, en el inspector de propiedades, escriba **ball_mc** en el cuadro de texto Nombre de instancia.

4. Seleccione el fotograma 1 de la línea de tiempo y añada el siguiente código en el panel Acciones:

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
new Tween(ball_mc, "_alpha", Strong.easeIn, 100, 0, 3, true);
```

   En lugar de moverse por el escenario, `ball_mc` ahora desaparece progresivamente desde una visibilidad del 100% hasta alcanzar la transparencia total en tres segundos. Para que el símbolo desaparezca más rápidamente, cambie el parámetro de duración de `3` a `1` o `2`.

5. Seleccione Control > Probar película para ver la animación.

   Si cambia la velocidad de fotogramas del documento, la animación parece reproducirse con mayor suavidad. Para obtener información sobre animaciones y velocidad de fotogramas, consulte "Animaciones y velocidad de fotogramas" en la página 503.

En lugar de utilizar segundos, puede hacer que el símbolo desaparezca progresivamente en unos pocos fotogramas. Para establecer la duración en fotogramas en lugar de segundos en la clase Tween, deberá cambiar el parámetro final, `useSeconds`, de `true` a `false`. Al establecer el parámetro con el valor `true`, estará indicando a Flash que la duración especificada está expresada en segundos. Si establece el parámetro con el valor `false`, la duración será el número de fotogramas que desea utilizar para la interpolación. En el siguiente procedimiento se muestra cómo establecer una interpolación en fotogramas en lugar de en segundos.

### Para establecer una duración de fotogramas en lugar de segundos:

1. Cree un documento nuevo y asígnele el nombre **framesTween.fla**.
2. Cree un clip de película en el escenario.
3. Seleccione la instancia de clip de película y, en el inspector de propiedades, escriba **ball_mc** en el cuadro de texto Nombre de instancia.
4. Seleccione el fotograma 1 de la línea de tiempo y añada el siguiente código en el panel Acciones:

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
new Tween(ball_mc, "_alpha", Strong.easeIn, 100, 0, 24, false);
```

   Este código hace que la instancia `ball_mc` desaparezca progresivamente utilizando el método de suavizado `Strong.easeIn`. En lugar de hacer que la instancia desaparezca en tres segundos, ésta desaparece en 24 fotogramas.

5. Seleccione Control > Probar película para ver la animación.

   Espere un momento y comprobará que la instancia desaparece progresivamente en 24 fotogramas.

6. Regrese al entorno de edición y abra el inspector de propiedades.

7. Cambie la velocidad de fotogramas del documento a **24** fps.

   Si aumenta la velocidad de fotogramas del archivo FLA, verá cómo la instancia desaparece antes. Para obtener información sobre animaciones y velocidad de fotogramas, consulte "Animaciones y velocidad de fotogramas" en la página 503.

El uso de fotogramas en lugar de segundos ofrece mayor flexibilidad, pero recuerde que la duración está en relación con la velocidad de fotogramas del documento de Flash actual. Si el documento de Flash utiliza una velocidad de fotogramas de 12 fotogramas por segundo (fps), el fragmento de código anterior hará que la instancia desaparezca progresivamente en dos segundos (24 fotogramas/12 fps = 2 segundos). Sin embargo, si la velocidad de fotogramas es de 24 fps, el mismo código hará que la instancia se desvanezca en un segundo (24 fotogramas/24 fps = 1 segundo). Si utiliza fotogramas para medir la duración, podrá cambiar de forma significativa la velocidad de la animación al cambiar la velocidad de fotogramas del documento sin necesidad de modificar el código ActionScript.

La clase Tween también cuenta con otras funciones muy útiles. Por ejemplo, puede escribir un controlador de eventos que se active cuando finalice la animación, como muestra el siguiente procedimiento:

### Para activar el código cuando una animación se ha completado:

1. Cree un nuevo documento y asígnele el nombre **triggerTween.fla**.
2. Cree un clip de película en el escenario.
3. Seleccione la instancia de clip de película y, en el inspector de propiedades, escriba **ball_mc** en el cuadro de texto Nombre de instancia.
4. Seleccione el fotograma 1 de la línea de tiempo y añada el siguiente código en el panel Acciones:

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
var tween_handler:Object = new Tween(ball_mc, "_alpha", Strong.easeIn,
  100, 0, 3, true);
tween_handler.onMotionFinished = function() {
  trace("onMotionFinished triggered");
};
```

   Si prueba este código ActionScript en el archivo FLA, observará que el mensaje "onMotionFinished triggered" aparece en el panel Salida una vez que `ball_mc` termine de desaparecer en el escenario.

5. Seleccione Control > Probar película para ver la animación.

   Espere un momento y comprobará que la instancia desaparece progresivamente. Cuando haya terminado de realizarse la interpolación, verá un mensaje en el panel Salida.

Para más información sobre funciones, consulte el Capítulo 6, "Clases"

## Animaciones continuas con el método continueTo()

En "Utilización de la clase Tween" en la página 524 se muestra cómo utilizar la clase Tween en las aplicaciones. Sin embargo, si desea mover la pelota después de que se haya completado la animación inicial, existen al menos dos métodos que lo permiten. Una solución sería volver a animar la pelota utilizando el controlador de eventos `onMotionFinished`. Sin embargo, la clase Tween ofrece una solución más simple: el método `continueTo()`. El método `continueTo()` indica a la animación interpolada que continúe desde su valor actual hasta un nuevo valor, como muestra el siguiente código ActionScript:

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
var ball_tween:Object = new Tween(ball_mc, "_x", Regular.easeIn, 0, 300, 3,
  true);
ball_tween.onMotionFinished = function() {
    ball_tween.continueTo(0, 3);
};
```

Una vez finalizada la interpolación inicial, el clip de película `ball_mc` se interpola de nuevo en su posición original de 0 píxeles. El siguiente fragmento de código (que se ha modificado para mayor brevedad) muestra el prototipo de función para el método `continueTo()`:

```
function continueTo(finish:Number, duration:Number):Void {
    /* se omite para ahorrar espacio. */
}
```

Sólo se pasan dos argumentos al método `continueTo()`, en lugar de los siete argumentos del método constructor de Tween, como muestra el siguiente fragmento:

```
function Tween (obj, prop, func, begin, finish, duration, useSeconds) {
    /* se omite para ahorrar espacio. */
}
```

Los cinco parámetros que no son necesarios para el método `continueTo()` (`obj`, `prop`, `func`, `begin` y `useSeconds`) utilizan los argumentos que definió anteriormente en la llamada a la clase Tween. Al llamar al método `continueTo()`, se da por hecho que los argumentos `obj`, `prop`, `func` (tipo de suavizado) y `useSeconds` son los mismos que los de la anterior llamada a la clase Tween. El método `continueTo()` utiliza el valor `finish` de la llamada a la clase Tween, en lugar de especificar un valor para el argumento `begin`, como muestra el siguiente código ActionScript:

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
var ball_tween:Object = new Tween(ball_mc, "_x", Regular.easeIn, 0, 300, 3,
  true);
ball_tween.onMotionFinished = function() {
     ball_tween.continueTo(0, 3);
};
```

Este código mueve la instancia `ball_mc` a lo largo del eje *x* desde 0 hasta 300 píxeles en tres segundos. Una vez que finaliza la animación, se activa el controlador de eventos `onMotionFinished` y se llama al método `continueTo()`. El método `continueTo()` indica al objeto de destino (`ball_mc`) que continúe desde su posición actual y prosiga la animación durante tres segundos a lo largo del eje *x* hasta llegar a los 0 píxeles utilizando el mismo método de suavizado. Se utilizarán los valores especificados en la llamada al método constructor de Tween para los parámetros que no defina en el método `continueTo()`. Si no especifica una duración para el método `continueTo()`, se utiliza la duración especificada en la llamada al constructor de Tween.

## Creación de animaciones que se ejecutan continuamente

Puede hacer que una animación avance y retroceda a lo largo del eje *x* sin detenerse. La clase Tween permite este tipo de animación a través de un método que, con gran acierto, recibe el nombre de `yoyo()`. El método `yoyo()` espera a que se ejecute el controlador de eventos `onMotionFinished` y, seguidamente, invierte los parámetros `begin` y `finish`. La animación comienza de nuevo, como muestra el siguiente procedimiento.

### Para crear una animación que continúe indefinidamente:

1. Cree un nuevo documento de Flash denominado **yoyo.fla**.
2. Abra el panel Acciones e introduzca el siguiente código ActionScript en el fotograma 1 de la línea de tiempo:

```
import mx.transitions.Tween;
import mx.transitions.easing.*;

this.createEmptyMovieClip("box_mc", this.getNextHighestDepth());
with (box_mc) {
  beginFill(0xFF0000, 60);
  moveTo(0, 0);
  lineTo(20, 0);
  lineTo(20, Stage.height);
  lineTo(0, Stage.height);
  lineTo(0, 0);
  endFill();
}
```

   La primera sección del código comienza importando la clase Tween y todas las clases del paquete de suavizado. En la siguiente sección del código, se crea un nuevo clip de película con el nombre de instancia `box_mc` y se dibuja un rectángulo de 20 píxeles de ancho y la misma altura en el escenario.

3. Añada el siguiente código ActionScript detrás del código creado en el paso anterior:

```
var box_tween:Tween = new Tween(box_mc, "_x", Regular.easeInOut, 0,
  Stage.width, 3, true);
box_tween.onMotionFinished = function() {
     box_tween.yoyo();
};
```

   Este código crea una nueva interpolación para animar el clip de película `box_mc` por el escenario a lo largo del eje *x*- durante 3 segundos.

4. Seleccione Control > Probar película para probar la animación.

   El cuadrado se desplaza de izquierda a derecha y a la inversa. Si la animación no es suave, puede aumentar la velocidad de fotogramas del documento de 12 fps a 24 fps.

   Al aproximarse al borde derecho del escenario, el cuadrado se desplaza fuera de los límites del escenario. Aunque esto no parezca un problema importante, lo más probable es que no desee que el rectángulo desaparezca de la vista por un lado del escenario y luego reaparezca un segundo más tarde desplazándose en la otra dirección.

   Para realizar ajustes, aplique animación al rectángulo desde los 0 píxeles hasta la anchura total del escenario menos la anchura del clip de película `box_mc`.

5. Para impedir que desaparezca el rectángulo, revise las líneas correspondientes del código del paso 3 y compruebe que coincide con el siguiente:

```
var box_tween:Tween = new Tween(box_mc, "_x", Regular.easeInOut, 0,
  (Stage.width - box_mc._width), 3, true);
```

6. Vuelva a probar la animación (Control > Probar película).

   El suavizado del cuadrado se interrumpe antes de desaparecer por el borde del escenario.

## Combinación de las clases TransitionManager y Tween

Puede generar efectos interesantes al combinar las clases TransitionManager y Tween. Puede utilizar la clase TransitionManager para desplazar el clip de película a lo largo del eje *x* mientras ajusta la propiedad `_alpha` del mismo clip empleando la clase Tween. Cada clase puede utilizar un método de suavizado diferente, lo que ofrece numerosas posibilidades de animación para los objetos de archivos SWF. Puede aprovechar las ventajas de los métodos `continueTo()` y `yoyo()` de la clase Tween o el controlador de eventos `onMotionFinished` para crear un efecto único.

Puede combinar las clases TransitionManager y Tween para aplicar animación a un clip de película cargado dinámicamente y hacer que desaparezca progresivamente en el escenario tras cargarse completamente del servidor remoto, como muestra el siguiente procedimiento.

### Para utilizar las clases TransitionManager y Tween conjuntamente:

1. Cree un nuevo documento de Flash y guarde el archivo como **combination.fla**.
2. Añada el siguiente código ActionScript en el fotograma 1 de la línea de tiempo:

```
import mx.transitions.*;
import mx.transitions.easing.*;

var mcl_obj:Object = new Object();
mcl_obj.onLoadInit = function(target_mc:MovieClip) {
    new Tween(target_mc, "_alpha", Strong.easeIn, 0, 100, 2, true);
    TransitionManager.start(target_mc, {type:Fly,
  direction:Transition.IN, duration:3, easing:Elastic.easeInOut,
  startPoint:6});
};

var my_mcl:MovieClipLoader = new MovieClipLoader();
my_mcl.addListener(mcl_obj);
my_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  this.createEmptyMovieClip("img_mc", this.getNextHighestDepth()));
```

Este código se divide en tres secciones principales.

La primera sección del código importa las clases del paquete de transiciones, así como el paquete `transitions.easing`. En este ejemplo, se importa todo el paquete de transiciones para no tener que introducir el nombre completo de la clase Tween, la clase TransitionManager o la transición seleccionada (en este caso, *Fly*). Este proceso puede reducir la cantidad de código que es preciso escribir, con lo que se evitan también posibles errores tipográficos.

La segunda sección del código ActionScript crea un objeto detector para la instancia de la clase MovieClipLoader que se crea en la tercera sección del código. Cuando el clip de película de destino se carga en la instancia de MovieClipLoader, se activa el evento `onLoadInit` y se ejecuta un bloque de código, que llama tanto a la clase Tween como a la clase TransitionManager. Este controlador de eventos hace que el clip de película desaparezca progresivamente, ya que se ha modificado la propiedad `_alpha` de la clase Tween y el clip de película de destino "vuela" (*flies*) a lo largo del eje *x*.

La tercera sección del código ActionScript crea una instancia de MovieClipLoader y aplica el objeto detector que ha creado anteriormente (de forma que la instancia de cargador del clip de película de destino pueda detectar el evento `onLoadInit`). Luego se carga la imagen JPEG de destino en un clip de película que se crea dinámicamente llamando al método `createEmptyMovieClip()`.

3. Guarde el documento y seleccione Control > Probar película para ver la animación en el entorno de prueba.

   Una vez que la imagen JPEG externa termina de descargarse del servidor, se desvanece gradualmente y se desplaza de derecha a izquierda por el escenario.

Para obtener información sobre la utilización de la clase Tween, consulte "Utilización de la clase Tween" en la página 524.

Para obtener información sobre cada método y propiedad de la clase Tween, consulte el Capítulo 51, "Clase Tween" en *Referencia del lenguaje de componentes*. Para obtener información sobre cada método y propiedad de la clase TransitionManager, consulte el Capítulo 48, "Clase TransitionManager" en *Referencia del lenguaje de componentes*.

Puede encontrar un archivo de origen de ejemplo que utiliza estas clases para añadir animación mediante scripts. Se denomina tweenProgress.fla y se encuentra en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Tween ProgressBar.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Tween ProgressBar.

# Utilización de efectos de filtro

Los filtros son efectos visuales que puede aplicar a los objetos mostrados durante la ejecución por Flash Player, como instancias de clip de película. Entre los filtros disponibles se incluyen sombra, desenfoque, iluminado, bisel, iluminado degradado y bisel degradado. También puede emplear un filtro de ajuste de color para modificar el brillo, el contraste, la saturación y el matiz del clip de película. Los filtros se pueden aplicar con la interfaz de usuario de Flash en Flash Professional 8 o mediante código ActionScript en Flash Basic 8 o Flash Professional 8.

Cada uno de estos filtros se puede aplicar a clips de película, botones o campos de texto desde la ficha Filtros del inspector de propiedades o mediante código ActionScript. Si emplea ActionScript para aplicar filtros a una instancia, también puede utilizar un filtro de mapa de desplazamiento (consulte "Utilización del filtro de mapa de desplazamiento" en la página 563) o un filtro de convolución (consulte "Utilización del filtro de convolución" en la página 561). Estos filtros se aplican a las definiciones vectoriales para que no exista la sobrecarga de almacenar una imagen de mapa de bits en el archivo SWF. Asimismo, puede escribir código ActionScript para modificar un filtro existente que ha aplicado a un campo de texto, un clip de película o un botón.

En el siguiente procedimiento se muestra cómo utilizar un controlador de eventos `onEnterFrame` para animar un efecto de filtro de iluminado en un clip de película.

### Para animar un efecto de filtro aplicado a una instancia de clip de película:

1. Cree un nuevo documento de Flash y guárdelo como **animFilter.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
this.createEmptyMovieClip("box_mc", 10);
box_mc.lineStyle(20, 0x000000);
box_mc.beginFill(0x000000);
box_mc.moveTo(0, 0);
box_mc.lineTo(160, 0);
box_mc.lineTo(160, 120);
box_mc.lineTo(0, 120);
box_mc.lineTo(0, 0);
box_mc.endFill();
box_mc._x = 100;
box_mc._y = 100;

box_mc.filters = [new flash.filters.GlowFilter()];
var dir:Number = 1;
box_mc.blur = 10;
box_mc.onEnterFrame = function() {
  box_mc.blur += dir;
  if ((box_mc.blur >= 30) || (box_mc.blur <= 10)) {
    dir *= -1;
  }
  var filter_array:Array = box_mc.filters;
  filter_array[0].blurX = box_mc.blur;
  filter_array[0].blurY = box_mc.blur;
  box_mc.filters = filter_array;
};
```

   Este código completa dos funcionalidades distintas. La primera sección crea y coloca una instancia de clip de película y dibuja un rectángulo redondeado negro en el escenario. El segundo bloque de código aplica un filtro de iluminado al rectángulo en el escenario y define un controlador de eventos `onEnterFrame`, que es el responsable de animar el efecto de filtro. El controlador de eventos `onEnterFrame` anima el efecto de filtro entre un desenfoque de 10 y 30 píxeles y, cuando la animación es mayor o igual a 30, o menor o igual a 10, ésta cambia de dirección.
3. Guarde los cambios en el documento de Flash y seleccione Control > Probar película para probar el archivo SWF.

Para más información sobre el trabajo con filtros en una aplicación, consulte los temas siguientes:


- "Utilización de paquetes de filtros" en la página 534
- "Utilización de filtros, caché y la clase MovieClip" en la página 536
- "Filtros de detección de zona activa, rotación, sesgo y escalado" en la página 538
- "Aplicación de filtros a instancias de objetos y BitmapData" en la página 538
- "Gestión de errores, rendimiento y filtros" en la página 539


Para ver un ejemplo del uso de ActionScript para aplicar filtros, puede encontrar un archivo de origen de muestra, Filters.fla, en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Filters.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Filters.

## Utilización de paquetes de filtros

Los paquetes son directorios que contienen uno o más archivos de clase y que residen en un directorio de ruta de clases determinado. Por ejemplo, el paquete flash.filters es un directorio del disco duro que contiene varios archivos de clase para cada tipo de filtro (como BevelFilter, BlurFilter, DropShadowFilter, entre otros) en Flash 8. Cuando los archivos de clase se organizan de esta forma, debe acceder a las clases de una determinada manera. Puede *importar* la clase o hacer referencia a ella con un *nombre completo*.

| NOTA | Para utilizar la sentencia import, debe especificar ActionScript 2.0 y Flash Player 6 o posterior en la ficha Flash del cuadro de diálogo Configuración de publicación del archivo FLA. |
|---|---|

La sentencia `import` permite acceder a las clases sin especificar sus nombres completos. Por ejemplo, para utilizar BlurFilter en un script, debe hacer referencia a esta clase con el nombre completo (flash.filters.BlurFilter) o importarla; si la importa, puede hacer referencia a ella por su nombre de clase (BlurFilter) en el código. El siguiente código ActionScript muestra las diferencias entre utilizar la sentencia import y los nombres de clase completos.

Si no realiza la importación de la clase BlurFilter, el código necesita utilizar el nombre completo de la clase (nombre del paquete seguido del nombre de la clase) para poder utilizar el filtro:

```
// sin importación
var myBlur:flash.filters.BlurFilter = new flash.filters.BlurFilter(10, 10,
  3);
```

El mismo código, escrito con una sentencia `import`, le permite acceder a BlurFilter con el nombre de la clase en lugar de hacer referencia siempre a ésta utilizando el nombre completo. De este modo, se ahorra tener que escribir y se reducen las posibilidades de cometer errores:

```
// con la importación
import flash.filters.BlurFilter;
var myBlur:BlurFilter = new BlurFilter(10, 10, 3);
```

Para importar varias clases dentro de un paquete (como BlurFilter, DropShadowFilter y GlowFilter), puede utilizar uno de los dos métodos para importar cada clase. El primer método para importar varias clases consiste en importar cada clase utilizando una sentencia `import` independiente, como se ve en el siguiente fragmento:

```
import flash.filters.BlurFilter;
import flash.filters.DropShadowFilter;
import flash.filters.GlowFilter;
```

La utilización de sentencias import independientes para cada clase del paquete puede resultar una tarea lenta y con muchas posibilidades de escribir errores. Puede evitar tener que importar archivos de clase individuales utilizando una importación con *comodín*, que importa todas las clases de un cierto nivel de un paquete. El siguiente código ActionScript muestra un ejemplo de una importación con comodín:

```
import flash.filters.*; // importa cada clase dentro del paquete
  flash.filters
```

La sentencia `import` sólo se aplica al script actual (fotograma u objeto) en el que se llama. Por ejemplo, supongamos que importa todas las clases del paquete `macr.util` en el fotograma 1 de un documento de Flash. En dicho fotograma, puede hacer referencia a las clases del paquete por sus nombres de clase en lugar de sus nombres completos. Para utilizar el nombre de clase en otro script del fotograma, haga referencia a las clases del paquete por sus nombres completos o añada una sentencia import al otro fotograma que importa las clases en dicho paquete.

Cuando utilice sentencias import, recuerde que las clases sólo se importan para el nivel especificado. Por ejemplo, si importa todas las clases del paquete `mx.transitions`, sólo se importan las clases dentro del directorio /transitions/, no todas las clases dentro de los subdirectorios (como las clases del paquete `mx.transitions.easing` ).

**SUGERENCIA**

Si importa una clase pero no la utiliza en el script, la clase no se exporta como parte del archivo SWF. Eso significa que puede importar paquetes grandes sin preocuparse del tamaño del archivo SWF; el código de bytes asociado con una clase se incluye en un archivo SWF únicamente si dicha clase se utiliza realmente.

Para ver un ejemplo del uso de ActionScript para aplicar filtros, puede encontrar un archivo de origen de muestra, Filters.fla, en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Filters.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Filters.

## Utilización de filtros, caché y la clase MovieClip

Durante la carga de un archivo SWF, si un clip de película tiene un filtro asociado, se marca para guardarlo en caché como mapa de bits transparente. Siempre que el clip de película tenga al menos un filtro aplicado, Flash Player lo almacena en caché como mapa de bits durante la ejecución forzando la propiedad `cacheAsBitmap` como `true`. Este mapa de bits almacenado en caché se utiliza como imagen de origen para los efectos de filtro. Normalmente, un clip de película tiene dos mapas de bits: uno es el clip de película de origen sin filtrar (original) y otro la imagen final una vez filtrada. Si no cambia la apariencia del clip de película durante la ejecución, no es preciso actualizar la imagen final, lo que mejora el rendimiento.

Puede acceder a los filtros aplicados a una instancia llamando a la propiedad `MovieClip.filters`. Al hacerlo, se devuelve una matriz que contiene todos los objetos de filtro actualmente asociados a la instancia de clip de película. Un filtro cuenta con un grupo de propiedades exclusivas, como las siguientes:

```
trace(my_mc.filters[0].angle); // 45.0
trace(my_mc.filters[0].distance); // 4
```

Puede acceder y modificar los filtros de la misma forma que lo hace con cualquier objeto de matriz. Al establecer y obtener los filtros con la propiedad, se devuelve un *duplicado* del objeto de filtro, no una *referencia*.

Para modificar un filtro existente, puede emplear código como el que se muestra en el siguiente procedimiento.

**Para modificar las propiedades de un filtro cuando se aplica a una instancia de clip de película:**

1. Cree un nuevo documento Flash y guarde el archivo como **modifyFilter.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
this.createEmptyMovieClip("my_mc", 10);
// dibujar cuadrado
with (my_mc) {
  beginFill(0xFF0000, 100);
  moveTo(0, 0);
  lineTo(100, 0);
  lineTo(100, 100);
  lineTo(0, 100);
  lineTo(0, 0);
  endFill();
}
my_mc._x = 100;
my_mc._y = 100;

// utilizar valores de DropShadowFilter predeterminados
my_mc.filters = [new flash.filters.DropShadowFilter()];
trace(my_mc.filters[0].distance); // 4
var filter_array:Array = my_mc.filters;
filter_array[0].distance = 10;
my_mc.filters = filter_array;
trace(my_mc.filters[0].distance); // 10
```

   La primera sección de este código utiliza la interfaz API de dibujo para crear un cuadrado rojo y coloca la forma en el escenario. La segunda sección del código aplica un filtro de sombra al cuadrado. A continuación, el código crea una matriz temporal para mantener los filtros actuales que se aplican el cuadrado rojo en el escenario. La propiedad `distance` del primer filtro se establece en 10 píxeles y el filtro modificado se vuelve a aplicar a la instancia de clip de película `my_mc`.
3. Seleccione Control > Probar película para probar el documento.

**NOTA**

En la actualidad no se ofrece compatibilidad para filtros de rotación basada en la rotación del principal o en otro tipo. El filtro de desenfoque siempre desenfoca bien horizontal o verticalmente, con independencia de la rotación o el sesgo de cualquier elemento del árbol de objetos principal.

**SUGERENCIA**

El contenido filtrado presenta las mismas restricciones de tamaño con la propiedad `cacheAsBitmap` establecida en `true`. Si el autor acerca demasiado el contenido en el archivo SWF, los filtros no son visibles cuando la representación de mapa de bits es superior a 2.880 píxeles en cualquier dirección. Si publica archivos SWF con filtros, resulta recomendable desactivar las opciones de menú de zoom.

Para ver un ejemplo del uso de ActionScript para aplicar filtros, puede encontrar un archivo de origen de muestra, Filters.fla, en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Filters.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Filters.

## Filtros de detección de zona activa, rotación, sesgo y escalado

Ninguna región no filtrada (por ejemplo, con sombra) que se encuentre fuera del rectángulo del recuadro de delimitación de la instancia de clip de película se considera parte de la superficie para la detección de zona activa (que determina si una instancia se solapa o presenta un punto de intersección con otra). Dado que la detección de zona activa se basa en vectores, no se puede realizar en el resultado de mapa de bits. Por ejemplo, si aplica un filtro de bisel a una instancia de botón, la detección de zona activa no se encuentra disponible en la parte biselada de la instancia.

Los filtros no admiten el escalado, la rotación ni el sesgo; si se escala la propia instancia (`_xscale` y `_yscale` no son 100%), el efecto de filtro no se escalará con ella. Esto significa que la forma original de la instancia rota, se escala o se sesga, pero el filtro no lo hace con ella.

Puede animar una instancia con un filtro para crear efectos realistas o anidar instancias, y utilizar la clase BitmapData para animar filtros con la intención de obtener el mismo efecto.

## Aplicación de filtros a instancias de objetos y BitmapData

El uso de filtros depende de la instancia de objeto al que se aplique el filtro. Siga las siguientes indicaciones cuando aplique un filtro a un objeto o a una instancia de BitmapData:

- Para aplicar filtros a clips de película, campos de texto y botones durante la ejecución, utilice la propiedad `filters`. El establecimiento de la propiedad `filters` de un objeto no modifica el objeto y se puede deshacer borrándola.
- Para aplicar filtros a instancias BitmapData, utilice el método `BitmapData.applyFilter()`. La llamada a `applyFilter()` en un objeto BitmapData modifica ese objeto BitmapData y no se puede deshacer.

**NOTA** (Sólo en Flash Professional 8) También puede aplicar efectos de filtro a imágenes y vídeos durante la edición desde la ficha Filtros del inspector de propiedades.

## Gestión de errores, rendimiento y filtros

Uno de los problemas derivados de emplear demasiados filtros en una aplicación es la posibilidad de utilizar grandes cantidades de memoria y hacer que se resienta el rendimiento de Flash Player. Dado que un clip de película con filtros asociados cuenta con dos mapas de bits de 32 bits, si emplea muchos mapas de bits, puede hacer que la aplicación utilice gran cantidad de memoria. Podría recibir un error de memoria insuficiente generado por el sistema operativo del equipo. En los equipos modernos, este tipo de errores son poco comunes, a menos que se empleen muchos efectos de filtro en una aplicación (por ejemplo, si tiene cientos de mapas de bits en el escenario).

Sin embargo, si encuentra un error de memoria insuficiente ocurre lo siguiente:

- La matriz de filtros se omite.
- El clip de película se dibuja con el procesador de vectores normal.
- No se almacena en caché ningún mapa de bits para el clip de película.

Después de ver un error de memoria insuficiente, el clip de película ya no intenta utilizar la matriz de filtros o la caché de mapa de bits. Otro factor que afecta al rendimiento del reproductor es el valor que utilice para el parámetro `quality` de cada filtro que aplique. Los valores mayores requieren más memoria y CPU para que el efecto se muestre, mientras que si el parámetro `quality` se establece en un valor menor, se requieren muchos menos recursos del equipo. Por tanto, debe evitar utilizar un número excesivo de filtros y establecer el parámetro `quality` en un valor menor siempre que sea posible.

**ATENCIÓN**

Si se acerca una vez un objeto de 100 por 100 píxeles, éste utiliza cuatro veces más memoria, ya que las dimensiones del contenido son de 200 por 200 píxeles. Si lo acerca otras dos veces, la forma se dibuja como objeto de 800 por 800 píxeles que emplea 64 veces más memoria que el objeto de 100 por 100 píxeles original. Siempre que utilice filtros en un archivo SWF, resulta recomendable desactivar las opciones de menú de zoom del menú contextual del archivo SWF.

Asimismo, puede encontrar errores si emplea tipos de parámetros no válidos. Algunos parámetros de filtro tienen un rango válido concreto. Si establece un valor no incluido en el rango válido, el valor cambiará a un valor válido que se encuentre dentro del rango. Por ejemplo, el valor de `quality` debe estar entre 1 y 3 para una operación estándar y sólo puede establecerse de 0 a 15. Cualquier valor superior a 15 se establece en 15.

De la misma forma, algunos constructores presentan restricciones en la longitud de las matrices necesarias como parámetros de entrada. Si se crea un filtro de convolución o de matriz de colores con una matriz no válida (que no sea del tamaño correcto), se produce un error en el constructor y el filtro no se crea correctamente. A continuación, el objeto de filtro se omite si se utiliza como entrada en una matriz de filtros de un clip de película.

SUGERENCIA

Cuando se utiliza un filtro de desenfoque, los valores de blurX y blurY que sean potencias de 2 (como 2, 4, 8, 16 y 32) se procesan más rápidamente y ofrecen una mejora del rendimiento de entre el 20% y el 30%.

# Utilización de filtros con código ActionScript

El paquete flash.filters contiene clases para los efectos de filtro de mapa de bits que son nuevas en Flash Player 8. Los filtros permiten utilizar código ActionScript para aplicar efectos visuales muy diversos, como desenfoque, bisel, iluminado y sombra, a instancias de texto, clip de película y botón. También puede utilizar la herramienta de edición de Flash para aplicar efectos de filtro a objetos como texto, imágenes y vídeos. Flash ofrece nueve efectos de filtro, aunque sólo siete de ellos se encuentran disponibles desde la interfaz de usuario de Flash Professional 8. Los filtros ConvolutionFilter y DisplacementMapFilter sólo se pueden emplear con código ActionScript.

NOTA

Todos los filtros están disponibles desde ActionScript en Flash Basic 8 y Flash Professional 8.

En el siguiente procedimiento se carga una imagen PNG semitransparente y se aplica el efecto GlowFilter a la parte no transparente de la misma.

### Para aplicar filtros a imágenes semitransparentes:

1. Cree un nuevo documento de Flash y guárdelo como **transparentImg.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
import flash.filters.GlowFilter;
System.security.allowDomain("http://www.helpexamples.com");
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  target_mc._x = (Stage.width - target_mc._width) / 2;
  target_mc._y = (Stage.height - target_mc._height) / 2;
  var glow:GlowFilter = new GlowFilter();
  target_mc.filters = [glow];
};
this.createEmptyMovieClip("img_mc", 10);
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mclListener);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/logo.png",
  img_mc);
```

   Este código utiliza una instancia de cargador de clip de película para cargar una imagen PNG semitransparente. Una vez se ha cargado la imagen, ésta se mueve al centro del escenario y se le aplica el filtro de iluminado.

3. Seleccione Control > Probar película para probar el documento.

   El efecto de filtro de iluminado sólo se aplica al área opaca (no transparente) de la imagen PNG.

En las siguientes secciones se describe cómo utilizar los filtros:

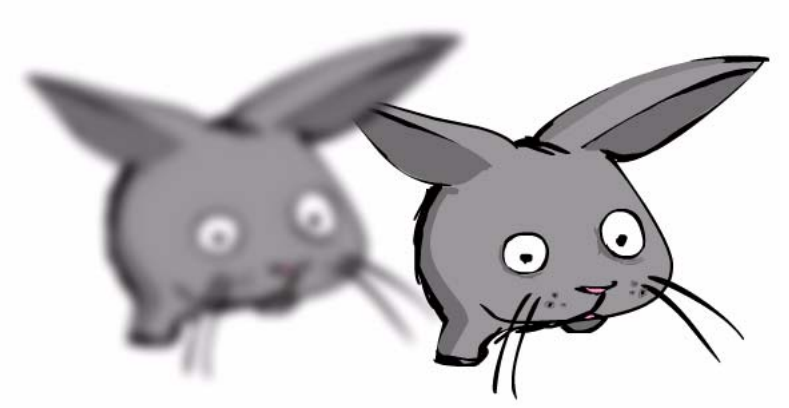

- "Utilización del filtro de desenfoque" en la página 542
- "Utilización del filtro de sombra" en la página 544
- "Utilización del filtro de iluminado" en la página 548
- "Creación de iluminados degradados" en la página 550
- "Utilización del filtro de bisel" en la página 552
- "Aplicación de un filtro de bisel degradado" en la página 558
- "Utilización del filtro de matriz de colores" en la página 559
- "Utilización del filtro de convolución" en la página 561
- "Utilización del filtro de mapa de desplazamiento" en la página 563


Para ver un ejemplo del uso de ActionScript para aplicar filtros, puede encontrar un archivo de origen de muestra, Filters.fla, en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Filters.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Filters.

## Utilización del filtro de desenfoque

La clase BlurFilter permite aplicar un efecto visual de desenfoque a diversos objetos en Flash. El efecto de desenfoque suaviza los detalles de una imagen. El desenfoque puede oscilar desde un ligero efecto hasta el desenfoque gaussiano, con un aspecto brumoso como el que se obtiene al mirar a través de un cristal semiopaco. El filtro de desenfoque se basa en un filtro de desenfoque de pasada de cuadro. El parámetro `quality` define cuántas veces se debe repetir el desenfoque (tres veces se aproxima a un filtro de desenfoque gaussiano).



**NOTA** El filtro de desenfoque sólo se escala cuando se aplica el zoom en el escenario.

Para más información sobre este filtro, consulte %{BlurFilter (flash.filters.BlurFilter)}% en *Referencia del lenguaje ActionScript 2.0*.

En el siguiente procedimiento se desenfoca una imagen cargada dinámicamente que se basa en la posición actual del puntero del ratón en el escenario. Cuanto más se aleja el puntero del centro del escenario, más se desenfoca la imagen.

### Para desenfocar una imagen basada en la posición del puntero del ratón:

1. Cree un nuevo documento de Flash y guárdelo como **dynamicblur.fla**.
2. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
import flash.filters.BlurFilter;
System.security.allowDomain("http://www.helpexamples.com");
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  // Centrar el clip de película target_mc en el escenario.
  target_mc._x = (Stage.width - target_mc._width) / 2;
  target_mc._y = (Stage.height - target_mc._height) / 2;
};
this.createEmptyMovieClip("img_mc", 10);
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mclListener);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img_mc);
var blur:BlurFilter = new BlurFilter(10, 10, 2);

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  /* Moving the pointer to the center of the Stage sets the blurX and
  blurY properties to 0%. */
  blur.blurX = Math.abs(_xmouse - (Stage.width / 2)) / Stage.width * 2 *
  255;
  blur.blurY = Math.abs(_ymouse - (Stage.height / 2)) / Stage.height * 2
  * 255;
  img_mc.filters = [blur];
};
Mouse.addListener(mouseListener);
```

La primera sección de este código carga y coloca en el escenario una imagen cargada dinámicamente. La segunda define un detector al que se llama siempre que se mueve el ratón. Puede calcular la cantidad de desenfoque horizontal y vertical en función de la posición actual del puntero del ratón en el escenario. Cuanto más se aleja el puntero del centro del escenario, más desenfoque se aplica a la instancia.

3. Seleccione Control > Probar película para probar el documento de Flash.

   Mueva el puntero a lo largo del eje *x* para modificar la cantidad de desenfoque horizontal. La instancia se desenfoca cuando el punteo se aleja del centro horizontal del escenario. Al mover el puntero a lo largo del eje *y*, el desenfoque vertical aumenta o disminuye, dependiendo de la distancia desde el centro vertical del escenario.

| SUGERENCIA | Cuando se utiliza un filtro de desenfoque, los valores de blurX y blurY que sean potencias de 2 (como 2, 4, 8, 16 y 32) se procesan más rápidamente y ofrecen una mejora del rendimiento de entre el 20% y el 30%. |
|---|---|

| ATENCIÓN | Si se establece un valor de desenfoque menor a 1.03125 se desactiva este efecto. |
|---|---|

## Utilización del filtro de sombra

La clase DropShadowFilter permite añadir una sombra a diversos objetos de Flash. El algoritmo de sombra se basa en el mismo filtro de cuadro que utiliza el filtro de desenfoque (consulte "Utilización del filtro de desenfoque" en la página 542). Hay varias opciones disponibles para el estilo de la sombra, incluida sombra interior o exterior y modo de extractor.

Para más información sobre el filtro de sombra, consulte %{DropShadowFilter (flash.filters.DropShadowFilter)}% en *Referencia del lenguaje ActionScript 2.0*.

En el siguiente procedimiento se emplea la interfaz API de dibujo para dibujar un cuadrado en el escenario. Cuando se mueve el puntero del ratón horizontalmente por el escenario, este código modifica la distancia desde el cuadrado en el que aparece la sombra, mientras que si se mueve verticalmente, se modifica cuánto se desenfoca la sombra.

### Para utilizar el filtro de sombra:

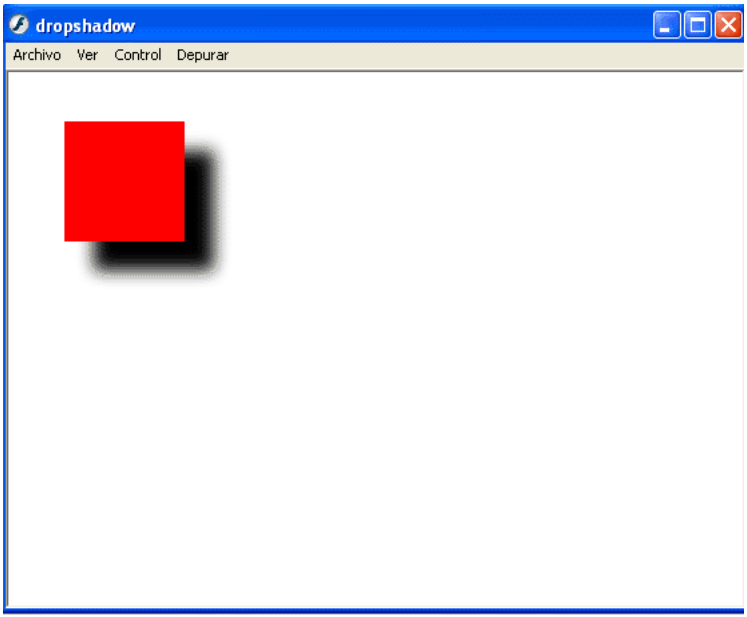
1. Cree un nuevo documento de Flash y guárdelo como **dropshadow.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
// import the filter classes
import flash.filters.DropShadowFilter;
// crear un clip de película denominado shapeClip
this.createEmptyMovieClip("shapeClip", 1);
// utilizar la interfaz API de dibujo para dibujar una forma
with (shapeClip) {
  beginFill(0xFF0000, 100);
  moveTo(0, 0);
  lineTo(100, 0);
  lineTo(100, 100);
  lineTo(0, 100);
  lineTo(0, 0);
  endFill();
}
// colocar la forma
shapeClip._x = 100;
shapeClip._y = 100;
// hacer clic en el cuadrado, aumentar la intensidad de la sombra
shapeClip.onPress = function():Void {
  dropShadow.strength++;
  shapeClip.filters = [dropShadow];
};
// crear un filtro
var dropShadow:DropShadowFilter = new DropShadowFilter(4, 45, 0x000000,
  0.4, 10, 10, 2, 3);

var mouseListener:Object = new Object();
// crear y aplicar un detector que controle el filtro cuando se mueva el
  ratón
mouseListener.onMouseMove = function():Void {
  dropShadow.distance = (_xmouse / Stage.width) * 50 - 20;
  dropShadow.blurX = (_ymouse / Stage.height) * 10;
  dropShadow.blurY = dropShadow.blurX;
  shapeClip.filters = [dropShadow];
};
Mouse.addListener(mouseListener);
```

La primera sección del código crea un nuevo clip de película y utiliza la interfaz API de dibujo para dibujar un cuadrado rojo. La segunda sección define un detector de ratón al que se llama siempre que éste se mueve. El detector de ratón calcula la distancia de sombra y el nivel de desenfoque en función de las posiciones *x* e *y* actuales del puntero y vuelve a aplicar el filtro de sombra. Si hace clic en el cuadrado rojo, aumenta la intensidad de la sombra.

3. Seleccione Control > Probar película para probar el documento de Flash.

   Mueva el puntero del ratón por el eje *x* para cambiar el valor de la distancia de sombra y, a continuación, a lo largo del eje *y* para cambiar la cantidad de desenfoque que se aplica a la instancia de clip de película.




También puede crear sombras y aplicarlas a imágenes cargadas dinámicamente. En el siguiente procedimiento se muestra cómo cargar una imagen externa y aplicar un filtro de sombra que siga al puntero del ratón. Cuanto más se aleja el puntero de la esquina superior izquierda de la imagen, más desenfoque horizontal y vertical se aplica a la misma.

### Para crear una sombra que siga al puntero del ratón:

1. Cree un nuevo documento de Flash y guárdelo como **dropshadowmouse.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
import flash.filters.DropShadowFilter;
System.security.allowDomain("http://www.helpexamples.com");
var dropShadow:DropShadowFilter = new DropShadowFilter(4, 45, 0x000000,
  0.8, 10, 10, 2, 2);
// Cargar y colocar la imagen en el escenario.
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip):Void {
  target_mc._x = (Stage.width - target_mc._width) / 2;
  target_mc._y = (Stage.height - target_mc._height) / 2;
};
```

```
this.createEmptyMovieClip("img_mc", 10);
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mclListener);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img_mc);

// Cuando se mueva el ratón, volver a calcular la posición de la sombra.
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  var p1:Number = img_mc._y - _ymouse;
  var p2:Number = img_mc._x - _xmouse;
  var degrees:Number = Math.atan2(p1, p2) / (Math.PI / 180);
  dropShadow.distance = Math.sqrt(Math.pow(p1, 2) + Math.pow(p2, 2)) *
  0.5;
  dropShadow.blurX = dropShadow.distance;
  dropShadow.blurY = dropShadow.blurX;
  dropShadow.angle = degrees - 180;
  img_mc.filters = [dropShadow];
};
Mouse.addListener(mouseListener);
```

La primera sección de este código define una instancia de sombra, carga una imagen externa y vuelve a colocarla en el centro del escenario. La segunda sección define un detector de ratón al que se llama siempre que se mueve el puntero del ratón por el escenario. Cuando se mueve el ratón, el controlador de eventos vuelve a calcular la distancia y el ángulo entre el puntero y la esquina superior izquierda de la imagen. En función de este cálculo, el filtro de sombra se vuelve a aplicar al clip de película.

3. Seleccione Control > Probar película para probar el documento de Flash.

   Cuando se ejecuta el archivo SWF, la sombra sigue al puntero del ratón. Cuanto más se acerca el puntero a la esquina superior izquierda de la imagen en el escenario, menos efecto de desenfoque se aplica a la misma. A medida que el puntero se aleja de la esquina superior izquierda, el efecto de sombra se hace más evidente.

También puede aplicar sombras a imágenes PNG semitransparentes cargadas dinámicamente. En el siguiente procedimiento, el filtro de sombra sólo se aplica al área sólida, no a la transparente, de la imagen PNG.

### Para aplicar un filtro de sombra a una imagen semitransparente:

1. Cree un nuevo documento de Flash y guárdelo como **dropshadowTransparent.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
import flash.filters.DropShadowFilter;
System.security.allowDomain("http://www.helpexamples.com");
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip):Void {
  target_mc._x = (Stage.width - target_mc._width) / 2;
  target_mc._y = (Stage.height - target_mc._height) / 2;
  var dropShadow:DropShadowFilter = new DropShadowFilter(4, 45,
  0x000000, 0.5, 10, 10, 2, 3);
  target_mc.filters = [dropShadow];
};
mclListener.onLoadError = function(target_mc:MovieClip):Void {
  trace("unable to load image.");
};
this.createEmptyMovieClip("logo_mc", 10);
var my_mcl:MovieClipLoader = new MovieClipLoader();
my_mcl.addListener(mclListener);
my_mcl.loadClip("http://www.helpexamples.com/flash/images/logo.png",
  logo_mc);
```

   Este código ActionScript utiliza la clase MovieClipLoader para cargar una imagen y aplicarle un filtro de sombra cuando se ha cargado completamente del servidor remoto.
3. Seleccione Control > Probar película para probar el documento de Flash.

   Flash carga una imagen PNG con fondo transparente. Cuando se aplica el filtro de sombra, sólo se hace a la parte opaca (no transparente) de la misma.

## Utilización del filtro de iluminado

La clase GlowFilter permite añadir un efecto de iluminado a distintos objetos en Flash. El algoritmo de iluminado se basa en el mismo filtro de cuadro que utiliza el filtro de desenfoque (consulte "Utilización del filtro de desenfoque" en la página 542). Cuenta con varias opciones para establecer el estilo del iluminado, incluidos iluminado interior o exterior y modo de extractor. El filtro de iluminado es muy similar al filtro de sombra con las propiedades `distance` y `angle` de la sombra establecidas en `0`.

Para más información sobre el filtro de iluminado, consulte %{GlowFilter (flash.filters.GlowFilter)}% en *Referencia del lenguaje ActionScript 2.0*.

En el siguiente procedimiento se muestra cómo aplicar un filtro de iluminado a un clip de película creado dinámicamente en el escenario. Al mover el puntero del ratón por el escenario cambia el desenfoque del clip de película y, al hacer clic en la forma creada dinámicamente, aumenta la intensidad del filtro.

### Para utilizar el filtro de iluminado:

1. Cree un nuevo documento de Flash y guárdelo como **glowfilter.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
import flash.filters.GlowFilter;

this.createEmptyMovieClip("shapeClip", 10);
with (shapeClip) {
  beginFill(0xFF0000, 100);
  moveTo(0, 0);
  lineTo(100, 0);
  lineTo(100, 100);
  lineTo(0, 100);
  lineTo(0, 0);
  endFill();
}
shapeClip._x = 100;
shapeClip._y = 100;
shapeClip.onPress = function():Void {
  glow.strength++;
  shapeClip.filters = [glow];
};
var glow:GlowFilter = new GlowFilter(0xCC0000, 0.5, 10, 10, 2, 3);
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  glow.blurX = (_xmouse / Stage.width) * 255;
  glow.blurY = (_ymouse / Stage.width) * 255;
  shapeClip.filters = [glow];
};
Mouse.addListener(mouseListener);
```

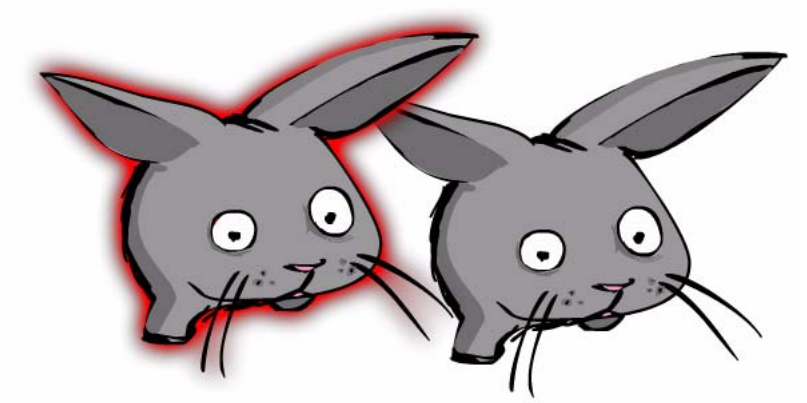
   Este código utiliza la interfaz API de dibujo para dibujar un cuadrado en el escenario y aplica un filtro de iluminado a la forma. Siempre que se mueve el puntero del ratón a lo largo del eje *x* o *y*, el desenfoque del filtro de iluminado se calcula y se aplica a la forma.

3. Seleccione Control > Probar película para probar el documento.

   La cantidad de desenfoque horizontal y vertical se calcula en función de la posición `_xmouse` e `_ymouse` del puntero del ratón. A medida que se mueve el puntero del ratón a la esquina superior izquierda del escenario, disminuye la cantidad de desenfoque horizontal y vertical. De manera inversa, a medida que se mueve el puntero del ratón a la esquina inferior derecha del escenario, aumenta la cantidad de desenfoque horizontal y vertical.

# Creación de iluminados degradados

La clase GradientGlowFilter permite crear un efecto de iluminado degradado en diversos objetos en Flash. Un iluminado degradado es un iluminado de aspecto realista con un degradado de color que se puede especificar. Puede aplicar un iluminado degradado alrededor del borde interior o exterior de un objeto o sobre un objeto.



Para más información sobre este filtro, consulte %{GradientBevelFilter (flash.filters.GradientBevelFilter)}% en *Referencia del lenguaje ActionScript 2.0*.

El siguiente procedimiento utiliza la interfaz API de dibujo para dibujar un cuadrado en el escenario y después aplica un filtro de iluminado degradado a la forma. Al hacer clic en el cuadrado en el escenario, aumenta la intensidad del filtro, mientras que si se mueve el puntero del ratón horizontal o verticalmente, se modifica la cantidad de desenfoque a lo largo del eje *x* o *y*.

### Para aplicar un filtro de iluminado degradado:

1. Cree un nuevo documento de Flash y guárdelo como **gradientglow.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
import flash.filters.GradientGlowFilter;
// crear una nueva instancia de shapeClip
var shapeClip:MovieClip = this.createEmptyMovieClip("shapeClip", 10);
// utilizar la interfaz API de dibujo para crear una forma
with (shapeClip) {
  beginFill(0xFF0000, 100);
  moveTo(0, 0);
  lineTo(100, 0);
  lineTo(100, 100);
  lineTo(0, 100);
  lineTo(0, 0);
  endFill();
}
```

```
// colocar la forma
shapeClip._x = 100;
shapeClip._y = 100;
// definir un iluminado degradado
var gradientGlow:GradientGlowFilter = new GradientGlowFilter(0, 45,
  [0x000000, 0xFF0000], [0, 1], [0, 255], 10, 10, 2, 3, "outer");

// definir un detector de ratón, detectar dos eventos
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function():Void {
  gradientGlow.strength++;
  shapeClip.filters = [gradientGlow];
};
mouseListener.onMouseMove = function():Void {
  gradientGlow.blurX = (_xmouse / Stage.width) * 255;
  gradientGlow.blurY = (_ymouse / Stage.height) * 255;
  shapeClip.filters = [gradientGlow];
};
Mouse.addListener(mouseListener);
```

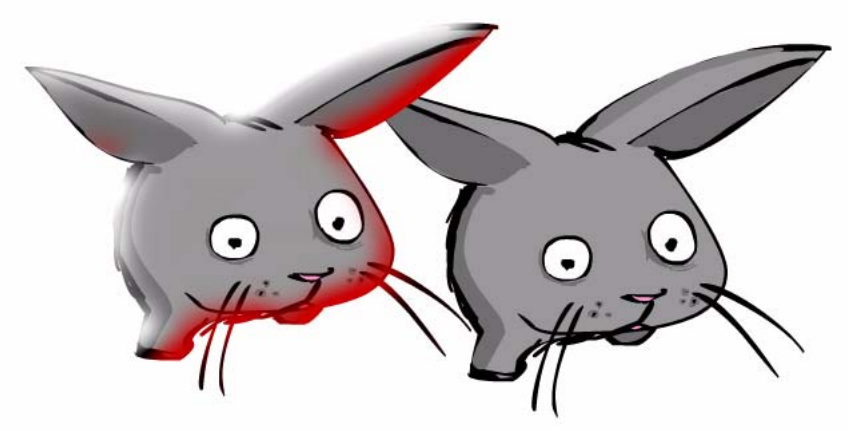
El código anterior se divide en tres secciones. La primera sección del código utiliza la interfaz API de dibujo para crear un cuadrado y coloca la forma en el escenario. La segunda define una nueva instancia de filtro de iluminado degradado que crea un iluminado de rojo a negro. La tercera define un detector de ratón que detecta dos controladores de eventos de ratón. El primer controlador es `onMouseDown`, que hace que aumente la intensidad del iluminado degradado. El segundo controlador es `onMouseMove`, al que se llama siempre que se mueve el puntero del ratón dentro del archivo SWF. Cuanto más se aleje el puntero de la esquina superior izquierda del documento de Flash, mayor será el efecto de iluminado que se le aplique.

3. Seleccione Control > Probar película para probar el documento.

   A medida que se mueve el puntero del ratón por el escenario, aumenta el desenfoque del filtro de iluminado degradado y disminuye su intensidad. Haga clic en el botón izquierdo del ratón para aumentar la intensidad del iluminado.

# Utilización del filtro de bisel

La clase BevelFilter permite añadir un efecto de biselado a diversos objetos en Flash. El efecto de biselado da aspecto tridimensional a los objetos. Puede personalizar el aspecto del biselado con distintos colores de resaltado y sombra, la cantidad de desenfoque del biselado, el ángulo del biselado, la ubicación del biselado y un efecto de extractor.



Para más información sobre este filtro, consulte %{BevelFilter (flash.filters.BevelFilter)}% en *Referencia del lenguaje ActionScript 2.0*.

El siguiente procedimiento utiliza la interfaz API de dibujo para crear un cuadrado y añade un bisel a la forma.

### Para utilizar el filtro de bisel:

1. Cree un nuevo documento de Flash y guárdelo como **bevel.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
import flash.filters.BevelFilter;
// definir un filtro de bisel
var bevel:BevelFilter = new BevelFilter(4, 45, 0xFFFFFF, 1, 0xCC0000, 1,
  10, 10, 2, 3);
// crear una nueva instancia de shapeClip
var shapeClip:MovieClip = this.createEmptyMovieClip("shapeClip", 1);
// utilizar la interfaz API de dibujo para crear una forma
with (shapeClip) {
  beginFill(0xFF0000, 100);
  moveTo(0, 0);
  lineTo(100, 0);
  lineTo(100, 100);
  lineTo(0, 100);
  lineTo(0, 0);
  endFill();
}
```

```
// colocar la forma en el escenario
shapeClip._x = 100;
shapeClip._y = 100;
// hacer clic con el ratón para aumentar la intensidad
shapeClip.onPress = function():Void {
  bevel.strength += 2;
  shapeClip.filters = [bevel];
};

// definir un detector para modificar el filtro cuando se mueva el ratón
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  bevel.distance = (_xmouse / Stage.width) * 10;
  bevel.blurX = (_ymouse / Stage.height) * 10;
  bevel.blurY = bevel.blurX;
  shapeClip.filters = [bevel];
};
Mouse.addListener(mouseListener);
```

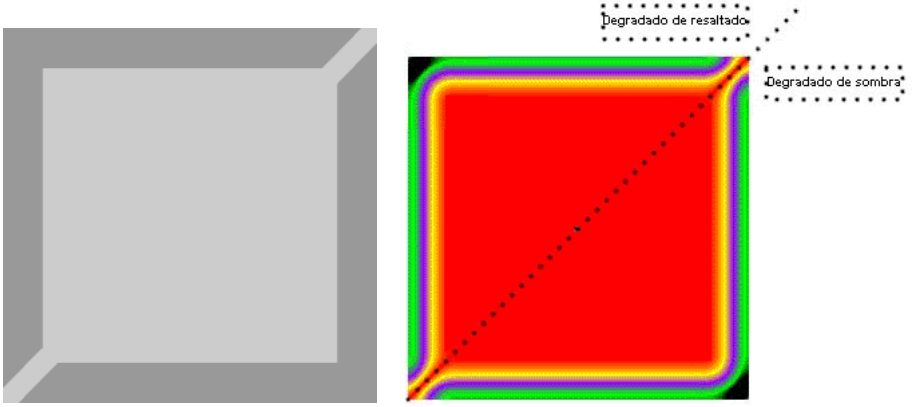
La primera sección del código define una instancia de BevelFilter y utiliza la interfaz API de dibujo para dibujar un cuadrado en el escenario. Cuando se hace clic en el cuadrado en el escenario, aumenta el valor de intensidad actual del bisel y se le da una apariencia más alta y nítida. La segunda sección del código define un detector de ratón, que modifica el desenfoque y la distancia del bisel en función de la posición actual del puntero.

3. Seleccione Control > Probar película para probar el documento de Flash.

   Cuando se mueve el puntero del ratón a lo largo del eje *x*, aumenta o disminuye la distancia de desplazamiento del bisel. Cuando se mueve a lo largo del eje *y*, las coordenadas actuales del puntero modifican la cantidad de desenfoque horizontal y vertical.

# El filtro de bisel degradado

El filtro de bisel degradado se aplica a objetos como un rectángulo, con los colores del degradado distribuidos en tres partes del rectángulo: dos bordes biselados (un *resaltado* y una *sombra*) y un área que llamaremos *relleno de base*. En los siguientes diagramas se muestra el rectángulo con el tipo de bisel establecido como interior. En el rectángulo de la izquierda, las áreas en gris oscuro son los bordes biselados y el área en gris claro es el relleno de base. En el rectángulo de la derecha, se ha aplicado un bisel degradado de arco iris con un bisel de cuatro colores en cada borde.




Las distintas propiedades del filtro de bisel degradado controlan la forma en la que se aplica el filtro. Los colores del bisel degradado se establecen en la matriz de colores. La distribución de los colores en cada parte del rectángulo la determina la matriz de proporciones. La propiedad distance determina la distancia de desplazamiento o a qué número de píxeles del objeto se aplica el filtro de bisel degradado. Las propiedades `blurX` y `blurY` controlan la nitidez de los colores del bisel; unos valores mayores hacen que aparezca más ancho y suave, mientras que unos valores menores lo muestran más delgado y nítido. La propiedad angle es la fuente de luz teórica que cae sobre el objeto, lo que produce un efecto de resaltado y sombra en sus bordes. La propiedad `strength` controla la extensión de los colores: un valor de intensidad menor apaga los colores, como ocurre en el ejemplo; por el contrario, un valor mayor hace que los números exteriores de la matriz sean más intensos, lo que hace que los colores del centro de la matriz estén menos marcados. Por último, con las propiedades `knockout` y `type` se determina cómo y dónde se aplica el filtro de bisel degradado al objeto completo: si el filtro extrae el objeto y dónde se coloca este último.

Uno de los conceptos más complicados de aplicar en el filtro de bisel degradado es la distribución de colores. Para comprender cómo se distribuyen en el filtro, piense primero qué colores desea incluir en él. Dado que un bisel simple presenta los conceptos ya conocidos de color de resaltado y color de sombra, puede aplicar los mismos conceptos para comprender el filtro de bisel degradado: dispone de un degradado de resaltado y un degradado de sombra. El resaltado aparece en la esquina superior izquierda y la sombra en la inferior derecha. Hay cuatro colores en el resaltado y otros cuatro en la sombra. Sin embargo, debe añadir otro color (el del relleno de base), que aparece donde se encuentran los bordes del resaltado y la sombra. Por tanto, son nueve los colores de la matriz, como se puede ver en el diagrama anterior.

El número de colores de la matriz determina el número de elementos de la matriz de proporciones y alfas. El primer elemento de la matriz de colores corresponde al primer elemento de la matriz de alfas y de la de proporciones y así sucesivamente. Como dispone de nueve colores, también dispone de nueve valores en la matriz de alfas y de otros nueve en la de proporciones. Los valores alfa establecen el valor de transparencia alfa de los colores.

Los valores de proporción en la matriz de proporciones pueden ir de 0 a 255 píxeles. El valor medio es 128, que corresponde al del relleno de base. En la mayoría de los casos, para obtener el efecto de bisel deseado, debe asignar los valores de proporción como se muestra a continuación, utilizando el ejemplo de los nueve colores:

- Los primeros cuatro colores van de 0 a 127, aumentando de valor de forma que cada valor sea superior o igual al anterior. Este es el primer borde del bisel, es decir, el resaltado en nuestro ejemplo.
- El quinto color (el central) es el relleno de base, que se establece en 128. El valor de píxel 128 define el relleno de base, que aparece bien en el exterior de la forma (o alrededor de los bordes del bisel) si el tipo establece como exterior, bien dentro de la forma, cubriendo el propio relleno del objeto, si el tipo se establece como interior.
- Los últimos cuatro colores van de 129 a 255, aumentando de valor de forma que cada valor sea superior o igual al anterior. Este es el segundo borde del bisel, la sombra en nuestro ejemplo.

Si piensa en un degradado como compuesto por rayas de varios colores que se mezclan, cada valor de proporción establece el número de píxeles del color asociado, por lo que establece la anchura de la raya de color en el degradado. Si desea una distribución igualada de los colores de cada borde:

- Utilice un número impar de colores, donde el color central sea el del relleno de base.
- Distribuya los valores entre 0 y 127 y entre 129 y 255 de manera uniforme entre los colores.

- Ajuste el valor de forma que cambie la anchura de cada raya de color en el degradado.

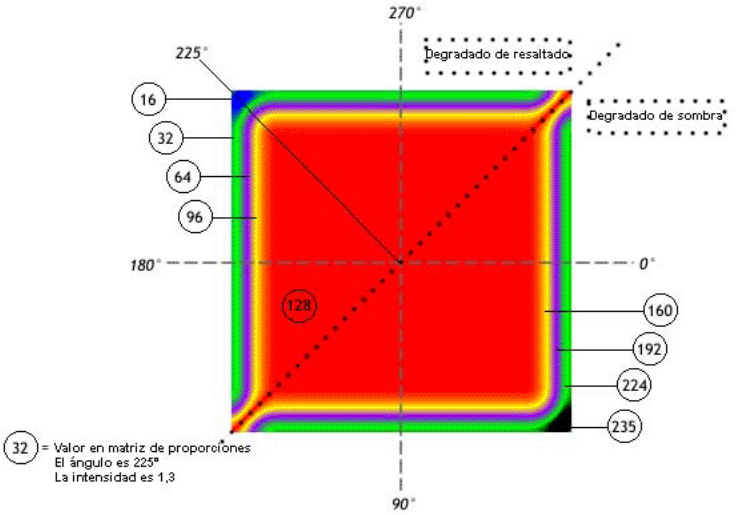
| NOTA | El valor de ángulo determina qué borde es el resaltado y cuál la sombra. |
|---|---|

El valor de ángulo determina el ángulo en el que se aplican los colores del degradado al objeto, es decir, dónde aparecen el resaltado y la sombra en el objeto. Los colores se aplican en el mismo orden que la matriz.

En el siguiente código se toma un cuadrado rosa (dibujado con la interfaz API de dibujo) y se le aplica un filtro de degradado de arco iris. Los colores, en el orden en el que aparecen en la matriz, son los siguientes: azul, verde, púrpura y amarillo (resaltado); rojo (relleno de base); amarillo, púrpura, verde, negro (sombra). Para determinar los valores de proporción, se asignan cuatro valores de color de resaltado de 0 a 127, lo que los hace aproximadamente iguales, y colores de sombra de 129 a 255. Los colores de los bordes externos son azul (16) y negro (235).

```
var colors:Array = [0x0000FF, 0x00FF00, 0x9900FF, 0xFFFF00, 0xFF0000,
  0xFFFF00, 0x9900FF, 0x00FF00,0x000000];
var alphas:Array = [1, 1, 1, 1, 1, 1, 1, 1, 1];
var ratios:Array = [16, 32, 64, 96, 128, 160, 192, 224, 235];
var gradientBevel:GradientBevelFilter = new GradientBevelFilter(8, 225,
  colors, alphas, ratios, 16, 16, 1.3, 2, "inner", false);
```

En la siguiente ilustración se muestra el filtro de bisel degradado creado por el código anterior, un bisel de arco iris de nueve colores aplicado a un clip de película en forma de rectángulo rojo:



La línea discontinua muestra cómo se determinan los ángulos. La ilustración muestra cómo se realiza el ángulo de 225° en el filtro y también el valor de proporción de cada color. Establecer el ángulo en 225° indica que el primer color de la matriz comienza en 225°, en la esquina superior izquierda (el resaltado). La línea punteada muestra dónde se aplica el degradado de resaltado y dónde el de sombra.

El color del clip de película original es el rosa, pero al establecer el valor 128 en rojo supone que el valor de 128 píxeles es el relleno de base que cubre el relleno original. Sin embargo, cuando se establece la propiedad filters, el objeto original no se altera; basta con borrar la propiedad `filters` para restaurar el relleno original del clip de película.

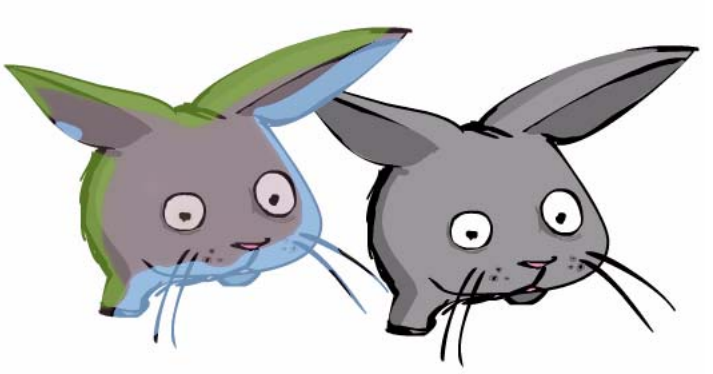
Las propiedades de todos los filtros se afectan mutuamente, por lo que si ajusta una propiedad para que cambie el efecto que se aplica, es posible que deba ajustar también alguna otra.

El código ActionScript completo utilizado para crear la ilustración anterior es el siguiente:

```
import flash.filters.GradientBevelFilter;

// dibuja una forma de cuadrado con relleno
this.createEmptyMovieClip("square_mc", this.getNextHighestDepth());
    square_mc.beginFill(0xFF99CC);
    square_mc.moveTo(40, 40);
    square_mc.lineTo(200, 40);
    square_mc.lineTo(200, 200);
    square_mc.lineTo(40, 200);
    square_mc.lineTo(40, 40);
    square_mc.endFill();

/* GradientBevelFilter(distance:Number, angle:Number, colors:Array,
  alphas:Array, ratios:Array, blurX:Number, blurY:Number, strength:Number,
  quality:Number, type:String, knockout:Boolean) */

// crear colores, alfas y matrices de proporciones
var colors:Array = [0x0000FF, 0x00FF00, 0x9900FF, 0xFFFF00, 0xFF0000,
  0xFFFF00, 0x9900FF, 0x00FF00,0x000000];//blue, green, purple, yellow,
  red, yellow, purple, green, black
var alphas:Array = [1, 1, 1, 1, 1, 1, 1, 1, 1];
var ratios:Array = [16, 32, 64, 96, 128, 160, 192, 224, 235];

// crear el objeto de filtro
var gradientBevel:GradientBevelFilter = new GradientBevelFilter(8, 225,
  colors, alphas, ratios, 16, 16, 1.3, 2, "inner", false);

// aplicar el filtro al clip de película en forma de cuadrado
square_mc.filters = [gradientBevel];
```

## Aplicación de un filtro de bisel degradado

La clase GradientBevelFilter permite aplicar un efecto de bisel degradado a objetos de Flash. Un bisel degradado es un borde biselado, realzado con color degradado en el exterior, interior o la parte superior de un objeto. Los bordes biselados aportan una apariencia tridimensional a los objetos y pueden ofrecer resultados muy coloristas, como se muestra en la siguiente ilustración.



Para más información sobre este filtro, consulte %{GradientBevelFilter (flash.filters.GradientBevelFilter)}% en *Referencia del lenguaje ActionScript 2.0*.

El siguiente procedimiento utiliza la interfaz API de dibujo para dibujar un cuadrado en el escenario y aplica un filtro de bisel degradado a la forma.

### Para utilizar el filtro de bisel degradado:

1. Cree un nuevo documento de Flash y guárdelo como **gradientbevel.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
import flash.filters.GradientBevelFilter;
var shapeClip:MovieClip = this.createEmptyMovieClip("shape_mc", 1);
with (shapeClip) {
  beginFill(0xFF0000, 100);
  moveTo(0, 0);
  lineTo(200, 0);
  lineTo(200, 200);
  lineTo(0, 200);
  lineTo(0, 0);
  endFill();
}
shapeClip._x = (Stage.width - shapeClip._width) / 2;
shapeClip._y = (Stage.height - shapeClip._height) / 2;
var colors:Array = new Array(0xFFFFFF, 0xCCCCCC, 0x000000);
var alphas:Array = new Array(1, 0, 1);
var ratios:Array = new Array(0, 128, 255);
```

```
var gradientBevel:GradientBevelFilter = new GradientBevelFilter(10, 45,
  colors, alphas, ratios, 4, 4, 5, 3);
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  gradientBevel.strength++;
  shapeClip.filters = [gradientBevel];
};
mouseListener.onMouseMove = function() {
  gradientBevel.blurX = (_xmouse / Stage.width) * 255;
  gradientBevel.blurY = (_ymouse / Stage.height) * 255;
  shapeClip.filters = [gradientBevel];
};
Mouse.addListener(mouseListener);
```

Este código utiliza la interfaz API de dibujo para dibujar y colocar un cuadrado en el centro del escenario. Cuando se mueve el puntero de ratón por el escenario, la cantidad de desenfoque a lo largo de los ejes *x* e *y* aumenta o disminuye. Cuando se mueve hacia la izquierda del escenario, disminuye la cantidad de desenfoque horizontal. Cuando se mueve hacia la derecha, aumenta el desenfoque. De la misma forma, cuanto más arriba esté el puntero en el escenario, menor será la cantidad de desenfoque que se aplica a lo largo del eje *y*.

3. Seleccione Control > Probar película para probar el documento y ver los resultados.

## Utilización del filtro de matriz de colores

La clase ColorMatrixFilter permite aplicar una transformación de matriz 4 x 5 en los valores de color RVA alfa y alfa de cada píxel de la imagen de entrada para producir un resultado con un nuevo conjunto de valores de color RVA alfa y alfa. Este filtro permite la rotación de matiz (diferente color o sombra), los cambios de saturación (intensidad de un matiz específico), la luminancia a alfa (brillo o intensidad de un color), entre otros efectos. Asimismo, puede animar estos filtros para crear efectos en las aplicaciones.

| NOTA | El filtro de matriz de colores se puede aplicar a mapas de bits e instancias de clip de película. |
|---|---|

Para más información sobre el filtro de matriz de colores, consulte %{ColorMatrixFilter (flash.filters.ColorMatrixFilter)}% en *Referencia del lenguaje ActionScript 2.0*.

Puede utilizar el filtro de matriz de colores para modificar el brillo de una instancia, como muestra el siguiente ejemplo.

### Para aumentar el brillo de un clip de película:

1. Cree un nuevo documento de Flash y guárdelo como **brightness.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
import flash.filters.ColorMatrixFilter;
System.security.allowDomain("http://www.helpexamples.com/");
var mcl_obj:Object = new Object();
mcl_obj.onLoadInit = function(target_mc:MovieClip):Void {
  var myElements_array:Array = [1, 0, 0, 0, 100,
       0, 1, 0, 0, 100,
       0, 0, 1, 0, 100,
       0, 0, 0, 1, 0];
  var myColorMatrix_filter:ColorMatrixFilter = new
  ColorMatrixFilter(myElements_array);
  target_mc.filters = [myColorMatrix_filter];
}
this.createEmptyMovieClip("img_mc", this.getNextHighestDepth());
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mcl_obj);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image2.jpg",
  img_mc);
```

   Este código carga dinámicamente una imagen JPEG mediante una instancia de MovieClipLoader. Una vez que la imagen se ha cargado completamente y se ha colocado en el escenario, el brillo de la instancia se establece en 100% mediante un filtro de matriz de colores.

3. Seleccione Control > Probar película para probar el documento.

También podría crear un efecto de brillo animado combinando la clase Tween y la clase ColorMatrixFilter, como muestra el siguiente procedimiento.

### Para animar el nivel de brillo de una instancia con la clase Tween:

1. Cree un nuevo documento de Flash y guárdelo como **brightnesstween.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
import flash.filters.ColorMatrixFilter;
import mx.transitions.Tween;
import mx.transitions.easing.*;
System.security.allowDomain("http://www.helpexamples.com");
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip):Void {
  // centrar instancia de clip de película en el escenario
  target_mc._x = (Stage.width - target_mc._width) / 2;
  target_mc._y = (Stage.height - target_mc._height) / 2;
  target_mc.watch("brightness", brightnessWatcher, target_mc);
  // animar el clip de película target_mc con un brillo entre -100 y +100
  var t:Object = new Tween(target_mc, "brightness", Elastic.easeOut,
  100, -100, 3, true);
  t.onMotionFinished = function() {
    this.yoyo();
  };
};
```

```
this.createEmptyMovieClip("img_mc", 10);
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mclListener);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img_mc);

function brightnessWatcher(prop:String, oldVal:Number, newVal:Number,
  target_mc:MovieClip):Number {
  var brightness_array:Array = [1, 0, 0, 0, newVal,
       0, 1, 0, 0, newVal,
       0, 0, 1, 0, newVal,
       0, 0, 0, 1, 0];
  target_mc.filters = [new ColorMatrixFilter(brightness_array)];
  return newVal;
};
```

La primera sección del código utiliza la clase MovieClipLoader para cargar una imagen JPEG en el escenario. Una vez que la imagen se ha cargado completamente, se vuelve a colocar en el centro del mismo. A continuación, se utiliza la clase Tween para animar el nivel de brillo de la imagen. Para ello se emplea el método `Object.watch()`, que registra un controlador de eventos que se inicia cuando cambia una propiedad especificada de un objeto de ActionScript. Siempre que el código ActionScript intenta establecer la propiedad de brillo personalizada de la instancia `target_mc`, se llama a la función `brightnessWatcher`. La función `brightnessWatcher` personalizada crea una nueva matriz que utiliza el filtro de matriz de colores para establecer el brillo de la imagen en una cantidad determinada.

3. Seleccione Control > Probar película para probar el documento.

   Una vez que la imagen se ha cargado y colocado en el escenario, el brillo se anima con un valor entre -100 y 100. Cuando la interpolación de brillo se ha completado, la animación se invierte mediante el método `Tween.yoyo()`, que hace que la interpolación tenga efecto de animación constantemente.

## Utilización del filtro de convolución

La clase ConvolutionFilter aplica un efecto de filtro de convolución de matrices. Una convolución combina píxeles en la imagen de origen especificada con píxeles colindantes para producir una imagen. El filtro de convolución permite conseguir una amplia variedad de operaciones de imagen, entre las que se incluyen efectos de desenfoque, detección de bordes, aumento de la nitidez, relieve y biselado.

| NOTA | Este filtro se puede aplicar a mapas de bits e instancias de clip de película. |
|---|---|

Una convolución de matriz se basa en una matriz *n* x *m*, que describe cómo el valor de un píxel determinado en la imagen de entrada se combina con los valores de los píxeles colindantes para producir un valor de píxel resultante. Cada píxel resultante se determina aplicando la matriz al píxel de origen correspondiente y a los píxeles colindantes.

Este filtro sólo está disponible mediante código ActionScript. Para más información sobre este filtro, consulte %{ConvolutionFilter (flash.filters.ConvolutionFilter)}% en *Referencia del lenguaje ActionScript 2.0*.

En el siguiente procedimiento se aplica el filtro de convolución a una imagen JPEG cargada dinámicamente.

### Para utilizar el filtro de convolución para modificar el color de una imagen:

1. Cree un nuevo documento de Flash y guárdelo como **convolution.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
import flash.filters.ConvolutionFilter;
import flash.display.BitmapData;

this.createEmptyMovieClip("shape_mc", 1);
shape_mc.createEmptyMovieClip("holder_mc", 1);
var imageLoader:MovieClipLoader = new MovieClipLoader();
imageLoader.loadClip("http://www.helpexamples.com/flash/images/
  image1.jpg", shape_mc.holder_mc);
var matrixArr:Array = [1, 4, 6, 4, 1, 4, 16, 24, 16, 4, 16, 6, 24, 36,
  24, 6, 4, 16, 24, 16, 4, 1, 4, 6, 4, 1];
var convolution:ConvolutionFilter = new ConvolutionFilter(5, 5,
  matrixArr);
shape_mc.filters = [convolution];

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  convolution.divisor = (_xmouse / Stage.width) * 271;
  convolution.bias = (_ymouse / Stage.height) * 100;
  shape_mc.filters = [convolution];
};
Mouse.addListener(mouseListener);
```

   El código anterior se divide en tres secciones independientes. La primera sección importa dos clases: ConvolutionFilter y BitmapData. La segunda crea un clip de película anidado y utiliza el objeto cargador de clips de película para cargar en él una imagen. Se crea entonces un objeto de filtro de convolución y se aplica al clip de película shape_mc. La sección final del código define un objeto detector de ratón que modifica las propiedades de divisor y sesgo del filtro de convolución en función de la posición actual del puntero y que vuelve a aplicar el filtro al clip de película `shape_mc`.

3. Seleccione Control > Probar película para probar el documento de Flash.

   Al mover el puntero del ratón a lo largo del eje *x* del escenario, se modifica el divisor del filtro, mientras que si se mueve por el eje *y* se modifica su sesgo.

# Utilización del filtro de mapa de desplazamiento

La clase DisplacementMapFilter utiliza los valores de píxel del objeto BitmapData especificado (llamado *imagen del mapa de desplazamiento*) para realizar un desplazamiento de una instancia en el escenario, como una instancia de clip de película o de mapa de bits. Puede utilizar este filtro para conseguir un efecto combado o moteado en una instancia especificada.

Este filtro sólo está disponible mediante código ActionScript. Para más información sobre este filtro, consulte %{DisplacementMapFilter (flash.filters.DisplacementMapFilter)}% en *Referencia del lenguaje ActionScript 2.0*.

En el siguiente procedimiento se carga una imagen JPEG y se aplica un filtro de mapa de desplazamiento que hace que la imagen aparezca distorsionada. Siempre que se mueve el puntero del ratón, se regenera el mapa de desplazamiento.

### Para distorsionar una imagen con el filtro de mapa de desplazamiento:

1. Cree un nuevo documento de Flash y guárdelo como **displacement.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
import flash.filters.DisplacementMapFilter;
import flash.geom.Point;
import flash.display.BitmapData;

var perlinBmp:BitmapData;
var displacementMap:DisplacementMapFilter;
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip):Void  {
     target_mc._x = (Stage.width - target_mc._width) / 2;
     target_mc._y = (Stage.height - target_mc._height) / 2;
     perlinBmp = new BitmapData(target_mc._width, target_mc._height);
     perlinBmp.perlinNoise(target_mc._width, target_mc._height, 10,
  Math.round(Math.random() * 100000), false, true, 1, false);
    displacementMap = new DisplacementMapFilter(perlinBmp, new Point(0,
  0), 1, 1, 100, 100, "color");
     shapeClip.filters = [displacementMap];
};
var shapeClip:MovieClip = this.createEmptyMovieClip("shapeClip", 1);
shapeClip.createEmptyMovieClip("holderClip", 1);
var imageLoader:MovieClipLoader = new MovieClipLoader();
imageLoader.addListener(mclListener);
imageLoader.loadClip("http://www.helpexamples.com/flash/images/
  image1.jpg", shapeClip.holderClip);

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  perlinBmp.perlinNoise(shapeClip._width, shapeClip._height, 10,
  Math.round(Math.random() * 100000), false, true, 1, false);
  shapeClip.filters = [displacementMap];
};
Mouse.addListener(mouseListener);
```

Este código carga una imagen JPEG y la coloca en el escenario. Una vez que la imagen se ha cargado completamente, este código crea una instancia de BitmapData y utiliza el método `perlinNoise()` para rellenarla con píxeles colocados aleatoriamente. La instancia de BitmapData se pasa al filtro de mapa de desplazamiento, que se aplica a la imagen y hace que aparezca distorsionada.

3. Seleccione Control > Probar película para probar el documento.

Mueva el puntero del ratón por el escenario para volver a crear un mapa de desplazamiento llamando al método `perlinNoise()`, que cambia la apariencia de la imagen JPEG.

# Manipulación de efectos de filtro mediante código

Flash Basic 8 y Flash Professional 8 permiten añadir dinámicamente distintos filtros a los clips de película, campos de texto y botones en el escenario, en lugar de tener que añadirlos en el entorno de edición de Flash Professional 8 (desde la ficha Filtros del inspector de propiedades). Cuando se añaden y manipulan filtros durante la reproducción, se pueden añadir sombras, desenfoques e iluminados de gran realismo que reaccionan a los movimientos del ratón o los eventos que realiza el usuario.

Para ver ejemplos de cómo manipular filtros mediante código, consulte las siguientes secciones:


- "Ajuste de propiedades de filtro" en la página 565
- "Animación de un filtro mediante código ActionScript" en la página 566
- "Utilización del método clone()" en la página 567

## Ajuste de propiedades de filtro

La matriz de filtros se aplica a un objeto al que se puede acceder a través de llamadas ActionScript estándar mediante la propiedad `MovieClip.filters`. Este proceso devuelve una matriz que contiene todos los objetos de filtro actualmente asociados con la instancia de MovieClip. Cada filtro cuenta con un grupo exclusivo de propiedades. Se puede acceder y modificar los filtros del mismo modo que se hace con un objeto de matriz, aunque obtener y establecer los filtros mediante la propiedad `filters` devuelve un duplicado del objeto de filtro en lugar de una referencia.

Al establecer la propiedad `filters`, la matriz de filtros que se pasa se duplica y no se almacena como referencia. Cuando se obtiene la propiedad filters, ésta devuelve una nueva copia de la matriz. Una implicación negativa de este enfoque es que no funciona el siguiente código:

```
// no funciona
my_mc.filters[0].blurX = 20;
```

Dado que el fragmento de código anterior devuelve una copia de la matriz de filtros, el código modifica la copia en vez de la matriz original. Para modificar la propiedad blurX, debería utilizar el siguiente código ActionScript:

```
// funciona
var filterArray:Array = my_mc.filters;
filterArray[0].blurX = 20;
my_mc.filters = filterArray;
```

El procedimiento siguiente desenfoca una imagen en función de la posición actual del puntero del ratón en el escenario. Cuando se mueve horizontal o verticalmente, las propiedades `blurX` y `blurY` del filtro de desenfoque se modifican en consecuencia.

### Para ajustar las propiedades del filtro de un clip de película:

1. Cree un nuevo documento de Flash y guárdelo como **adjustfilter.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
import flash.filters.BlurFilter;

this.createEmptyMovieClip("holder_mc", 10);
holder_mc.createEmptyMovieClip("img_mc", 20);
holder_mc.img_mc.loadMovie("http://www.helpexamples.com/flash/images/
  image2.jpg");
holder_mc.filters = [new BlurFilter(10, 10, 2)];
holder_mc._x = 75;
holder_mc._y = 75;

holder_mc.onMouseMove = function() {
  var tempFilter:BlurFilter = holder_mc.filters[0];
  tempFilter.blurX = Math.floor((_xmouse / Stage.width) * 255);
  tempFilter.blurY = Math.floor((_ymouse / Stage.height) * 255);
  holder_mc.filters = [tempFilter];
};
```

El código anterior se divide en tres secciones. La primera sección importa la clase flash.filters.BlurFilter para que no tenga que utilizar el nombre de clase completo al hacer referencia a la clase BlurFilter. La segunda crea dos clips de película y carga una imagen en uno de los clips anidados. La tercera sección responde al movimiento del ratón en el escenario y ajusta el desenfoque en consecuencia.

3. Seleccione Control > Probar película para probar el documento de Flash.

   Al mover el puntero del ratón a lo largo del eje *x*, se modifica la propiedad `blurX` del filtro de desenfoque. Al moverlo a lo largo del eje *y*, se modifica su propiedad `blurY`. Cuanto más cerca esté el puntero de la esquina superior izquierda del escenario, menos desenfoque se aplicará al clip de película.

## Animación de un filtro mediante código ActionScript

Puede utilizar código ActionScript, como la clase Tween, para animar los filtros durante la ejecución, lo que le permite incluir interesantes efectos de animación en las aplicaciones de Flash.

En el siguiente ejemplo se muestra cómo combinar BlurFilter con la clase Tween para crear un desenfoque animado que modifica el filtro Desenfocar entre un valor de 0 y 10 durante la ejecución.

### Para animar el desenfoque con la clase Tween:

1. Cree un nuevo documento de Flash y guárdelo como **animatedfilter.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
import flash.filters.BlurFilter;
import mx.transitions.Tween;
import mx.transitions.easing.*;

this.createEmptyMovieClip("holder_mc", 10);
holder_mc.createEmptyMovieClip("img_mc", 20);

var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  target_mc._x = (Stage.width - target_mc._width) / 2;
  target_mc._y = (Stage.height - target_mc._height) / 2;
  var myTween:Tween = new Tween(target_mc, "blur", Strong.easeInOut, 0,
  20, 3, true);
  myTween.onMotionChanged = function() {
    target_mc._parent.filters = [new BlurFilter(target_mc.blur,
  target_mc.blur, 1)];
  };
  myTween.onMotionFinished = function() {
    myTween.yoyo();
  }
};
```

```
var my_mcl:MovieClipLoader = new MovieClipLoader();
my_mcl.addListener(mclListener);
my_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  holder_mc.img_mc);
```

El código anterior se divide en tres secciones independientes. La primera sección importa las clases y los paquetes necesarios. La segunda crea un clip de película anidado que se emplea para cargar una imagen y aplicar filtros al clip de película contenedor. La última sección crea una nueva instancia de MovieClipLoader y un detector para el cargador de clips de película. El objeto detector define una función de controlador de evento único, `onLoadInit`, que se inicia una vez que la imagen se ha cargado correctamente y aparece en el escenario. En primer lugar, la imagen se vuelve a colocar en el centro del escenario y, a continuación, se crea el objeto Tween que anima el clip de película y aplica un filtro de desenfoque de 0 y 10.

3. Seleccione Control > Probar película para probar el documento de Flash.

## Utilización del método clone()

El método `clone()` de cada clase de filtro devuelve una nueva instancia de filtro con todas las propiedades de la instancia de filtro original. Cuando trabaje con filtros, es posible que desee hacer una copia de un filtro y para ello es preciso que haga un duplicado del filtro con el método `clone()`. Si no utiliza este método para la duplicación, Flash sólo crea una referencia al filtro original. En este caso, cualquier cambio realizado en el filtro duplicado también modifica el objeto de filtro original.

En el siguiente procedimiento se crea una nueva instancia de DropShadowFilter (`greenDropShadow`), se llama al método `clone()` para duplicar el filtro de sombra verde y se guarda un nuevo filtro con el nombre `redDropShadow`. El filtro clonado establece un nuevo color de sombra y ambos filtros se aplican a la instancia de clip de película `flower_mc` que se encuentra en el escenario.

### Para utilizar el método clone:

1. Cree un nuevo documento de Flash y asígnele el nombre **clone.fla**.
2. Cree un clip de película en el escenario.
3. Seleccione la instancia de clip de película y escriba **flower_mc** en el cuadro de texto Nombre de instancia del inspector de propiedades.

4. Seleccione el fotograma 1 de la línea de tiempo y añada el siguiente código en el panel Acciones:

```
import flash.filters.DropShadowFilter;
var greenDropShadow:DropShadowFilter = new DropShadowFilter();
greenDropShadow.color = 0x00FF00; // verde
var redDropShadow:DropShadowFilter = greenDropShadow.clone();
redDropShadow.color = 0xFF0000; // rojo
flower_mc.filters = [greenDropShadow, redDropShadow];
```

   El código anterior crea una nueva instancia del filtro de sombra y le asigna el nombre `greenDropShadow`. El objeto de sombra verde se duplica mediante el método `DropShadowFilter.clone()` y se crea un nuevo objeto de filtro denominado `redDropShadow`. Ambos filtros se aplican a la instancia de clip de película `flower_mc` en el escenario. Si no llamó al método `clone()`, ambas sombras aparecerán en rojo. Esto se debe a que al establecer la propiedad `redDropShadow.color`, se cambian tanto el objeto de sombra roja como de sombra verde puesto que el primero contiene una referencia al segundo.

5. Seleccione Control > Probar película para probar el documento de Flash.

   El filtro se duplica y clona y ambos filtros se aplican a la instancia `flower_mc`.

Para más información sobre el método `clone()`, consulte %{clone (método DropShadowFilter.clone)}% en *Referencia del lenguaje ActionScript 2.0*. Para obtener información relacionada, también puede consultar el método `clone()` de cualquier clase de filtro.

# Creación de mapas de bits con la clase BitmapData

La clase BitmapData permite crear imágenes de mapa de bits transparentes u opacas de tamaño arbitrario y manipularlas de distintas formas durante la ejecución. Cuando se manipula una instancia de BitmapData directamente mediante código ActionScript, es posible crear imágenes muy complejas sin recargar cada fotograma al redibujar constantemente el contenido de datos vectoriales en Flash Player. Los métodos de la clase BitmapData admiten diversos efectos que no están disponibles a través de la ficha Filtros del espacio de trabajo de Flash.

Un objeto BitmapData contiene una matriz de datos de píxel. Estos datos pueden representar un mapa de bits completamente opaco o un mapa de bits transparente que contiene datos del canal alfa. Ambos tipos de objetos BitmapData se almacenan como búfer de enteros de 32 bits. Cada entero de 32 bits determina las propiedades de un píxel único del mapa de bits. Cada entero de 32 bits es una combinación de cuatro valores *channel* de 8 bits (de cero a 255) que describen los valores de transparencia alfa y rojo, verde y azul (RVA alfa) del píxel.

Para obtener información sobre cómo trabajar con paquetes, consulte "Utilización de paquetes de filtros" en la página 534.

Puede encontrar un archivo de origen de ejemplo que utiliza la clase BitmapData para manipular una imagen. El archivo se denomina BitmapData.fla y se encuentra en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\BitmapData.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/BitmapData.

En el siguiente procedimiento se carga dinámicamente una imagen JPEG en el escenario y se utiliza la clase BitmapData para crear un efecto de ruido similar a la electricidad estática en un televisor. El efecto de ruido se vuelve a dibujar con un patrón aleatorio cada 100 milisegundos (1 décima de segundo). El movimiento del puntero del ratón a lo largo del eje *x* y del eje *y* tiene un efecto en la cantidad de electricidad estática que se aplica a cada intervalo.

### Para crear un efecto de ruido con la clase BitmapData:

1. Cree un nuevo documento de Flash y guárdelo como **noise.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
import flash.display.BitmapData;
this.createTextField("status_txt", 90, 0, 0, 100, 20);
status_txt.selectable = false;
status_txt.background = 0xFFFFFF;
status_txt.autoSize = "left";
function onMouseMove() {
  status_txt._x = _xmouse;
  status_txt._y = _ymouse-20;
  updateAfterEvent();
}
this.createEmptyMovieClip("img_mc", 10);
img_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
var noiseBmp:BitmapData = new BitmapData(Stage.width, Stage.height,
  true);
this.attachBitmap(noiseBmp, 20);
setInterval(updateNoise, 100);
var grayScale:Boolean = true;
```

```
function updateNoise():Void {
  var low:Number = 30 * _xmouse / Stage.width;
  var high:Number = 200 * _ymouse / Stage.height;
  status_txt.text = "low:" + Math.round(low) + ", high:" +
  Math.round(high);
  noiseBmp.noise(Math.round(Math.random() * 100000), low, high, 8,
  true);
}
```

Este código crea un campo de texto con el nombre de instancia `status_txt`, que sigue al puntero del ratón y muestra los valores actuales de los parámetros `high` y `low` del método `noise()`. La función `setInterval()` cambia el efecto de ruido, que se actualiza cada 100 milisegundos (1 décima de segundo) al llamar de forma continuada a la función `updateNoise()`. Los parámetros `high` y `low` del método `noise()` se determinan calculando la posición actual del puntero en el escenario.

3. Seleccione Control > Probar película para probar el documento.

   El movimiento del puntero del ratón a lo largo del eje *x* tiene un efecto en el parámetro `low`; la misma operación en el eje *y* afecta al parámetro `high`.

La clase BitmapData también permite distorsionar una imagen cargada dinámicamente mediante la combinación de un efecto de método `perlinNoise()` y de un filtro de mapa de desplazamiento. En el procedimiento siguiente se incluye una demostración.

### Para aplicar un filtro de mapa de desplazamiento a una imagen:

1. Cree un nuevo documento de Flash y guárdelo como **displacement.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
// Importar clases.
import flash.filters.DisplacementMapFilter;
import flash.display.BitmapData;
import flash.geom.Point;
// Crear un clip y un clip anidado.
var shapeClip:MovieClip = this.createEmptyMovieClip("shapeClip", 1);
shapeClip.createEmptyMovieClip("holderClip", 1);
// Cargar JPEG.
var imageLoader:MovieClipLoader = new MovieClipLoader();
imageLoader.loadClip("http://www.helpexamples.com/flash/images/
  image4.jpg", shapeClip.holderClip);
// Crear instancia de BitmapData.
var perlinBmp:BitmapData = new BitmapData(Stage.width, Stage.height);
perlinBmp.perlinNoise(Stage.width, Stage.height, 10,
  Math.round(Math.random() * 100000), false, true, 1, false);
// Crear y aplicar el filtro de mapa de desplazamiento.
var displacementMap:DisplacementMapFilter = new
  DisplacementMapFilter(perlinBmp, new Point(0, 0), 1, 1, 100, 100,
  "color", 1);
shapeClip.filters = [displacementMap];
```

```
// Crear y aplicar un detector.
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  perlinBmp.perlinNoise(Stage.width, Stage.height, 10,
  Math.round(Math.random() * 100000), false, true, 1, false);
  shapeClip.filters = [displacementMap];
}
Mouse.addListener(mouseListener);
```

Este ejemplo de código se divide en cinco secciones lógicas. La primera sección importa las clases necesarias para el ejemplo. El segundo bloque de código crea un clip de película anidado y carga una imagen JPEG de un servidor remoto. El tercer bloque crea una nueva instancia de BitmapData denominada `perlinBmp`, que tiene las mismas dimensiones que el escenario. La instancia `perlinBmp` contiene los resultados del efecto de ruido Perlin y se utiliza como parámetros para el filtro de mapa de desplazamiento. El cuarto bloque de código crea y aplica el efecto de filtro de mapa de desplazamiento a la imagen cargada dinámicamente que se creó en un paso anterior. El quinto y último bloque crea un detector de ratón que regenera el ruido Perlin que emplea el filtro de mapa de desplazamiento siempre que el usuario mueve el puntero del ratón.

3. Seleccione Control > Probar película para probar el documento de Flash.

# Modos de mezcla

Puede aplicar modos de mezcla a objetos de clip de película desde el espacio de trabajo de Flash (Flash Professional 8) o mediante código ActionScript (Flash Basic 8 y Flash Professional 8). Durante la ejecución, se mezclan varios gráficos como una única forma. Por esta razón no se pueden aplicar diferentes modos de mezcla a diferentes símbolos gráficos.

Para más información sobre el uso de ActionScript para aplicar modos de mezcla, consulte "Aplicación de modos de mezcla" en la página 572.

Los modos de mezcla implican combinar los colores de una imagen (la imagen base) con los de otra imagen (la imagen de mezcla) para producir una tercera imagen. Cada valor de píxel de una imagen se procesa con el correspondiente valor de píxel de la otra imagen para producir un valor de píxel para esa misma posición en el resultado.

La propiedad `MovieClip.blendMode` admite los siguientes modos de mezcla:

**add** Suele utilizarse para crear un efecto de disolución de aclarado animado entre dos imágenes.

**alpha** Suele utilizarse para aplicar la transparencia del primer plano al fondo.

**darken** Suele utilizarse para superponer el tipo.

**difference** Suele utilizarse para crear colores más vivos.

**erase** Suele utilizarse para *recortar* (borrar) parte del fondo utilizando el alfa de primer plano.

**hardlight** Suele utilizase para crear efectos de sombreado.

**Invert** Se utiliza para invertir el fondo.

**layer** Se utiliza para forzar la creación de un búfer temporal para la composición previa de un determinado clip de película.

**lighten** Suele utilizarse para superponer el tipo.

**multiply** Suele utilizarse para crear efectos de sombras y profundidad.

**normal** Se utiliza para especificar que los valores de píxeles de la imagen mezclada sustituyan a los de la imagen base.

**overlay** Suele utilizase para crear efectos de sombreado.

**screen** Suele utilizarse para crear resaltados y destellos de lentes.

**subtract** Suele utilizarse para crear un efecto de disolución de oscurecimiento animado entre dos imágenes.

## Aplicación de modos de mezcla

En el siguiente procedimiento se carga una imagen dinámica y se permite aplicar distintos modos de mezcla mediante la selección de un modo en un cuadro combinado en el escenario.

### Para aplicar distintos modos de mezcla a una imagen:

1. Cree un nuevo documento de Flash y guárdelo como **blendmodes.fla**.
2. Arrastre una instancia de componente ComboBox al escenario y asígnele el nombre de instancia **blendMode_cb**.
3. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
var blendMode_dp:Array = new Array();
blendMode_dp.push({data:"add", label:"add"});
blendMode_dp.push({data:"alpha", label:"alpha"});
blendMode_dp.push({data:"darken", label:"darken"});
blendMode_dp.push({data:"difference", label:"difference"});
blendMode_dp.push({data:"erase", label:"erase"});
blendMode_dp.push({data:"hardlight", label:"hardlight"});
blendMode_dp.push({data:"invert", label:"invert"});
blendMode_dp.push({data:"layer", label:"layer"});
blendMode_dp.push({data:"lighten", label:"lighten"});
blendMode_dp.push({data:"multiply", label:"multiply"});
blendMode_dp.push({data:"normal", label:"normal"});
blendMode_dp.push({data:"overlay", label:"overlay"});
blendMode_dp.push({data:"screen", label:"screen"});
blendMode_dp.push({data:"subtract", label:"subtract"});
blendMode_cb.dataProvider = blendMode_dp;
```

```
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  var blendModeClip:MovieClip =
  target_mc.createEmptyMovieClip("blendModeType_mc", 20);
  with (blendModeClip) {
    beginFill(0x999999);
    moveTo(0, 0);
    lineTo(target_mc._width / 2, 0);
    lineTo(target_mc._width / 2, target_mc._height);
    lineTo(0, target_mc._height);
    lineTo(0, 0);
    endFill();
  }
  target_mc._x = (Stage.width - target_mc._width) / 2;
  target_mc._y = (Stage.height - target_mc._height) / 2;
  blendModeClip.blendMode = blendMode_cb.value;
};
this.createEmptyMovieClip("img_mc", 10);
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mclListener);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img_mc);

function cbListener(eventObj:Object):Void {
  img_mc.blendModeType_mc.blendMode = eventObj.target.value;
}
blendMode_cb.addEventListener("change", cbListener);
```

Este código ActionScript rellena el cuadro combinado con cada tipo de modo de mezcla para que el usuario pueda ver cada efecto en la imagen cargada dinámicamente. Se crea un objeto detector, que se utiliza con la instancia MovieClipLoader. El objeto detector define un detector de evento único, `onLoadInit`, al que se llama cuando la imagen se ha descargado completamente y se ha inicializado en Flash. El detector de eventos crea un nuevo clip de película denominado `blendModeType_mc` y utiliza la interfaz API de dibujo para dibujar una forma rectangular sobre la mitad izquierda de la imagen. El modo de mezcla seleccionado actualmente para la instancia de ComboBox se aplica al clip de película `blendModeType_mc`.

El resto del código establece la instancia de MovieClipLoader, que es la responsable de cargar la imagen especificada en un clip de película en el escenario. Por último, se define un detector para la instancia de ComboBox `blendMode_cb`, que aplica el modo de mezcla seleccionado siempre que se selecciona un nuevo elemento de la instancia de ComboBox.

4. Seleccione Control > Probar película para probar el documento.

# Orden de operaciones

La siguiente lista representa el orden de las operaciones en el que la matriz de filtros, los modos de mezcla, las transformaciones de color y las capas de máscara se asocian o realizan en una instancia de clip de película:

1. El mapa de bits del clip de película se actualiza a partir del contenido vectorial (la propiedad `cacheAsBitmap` se establece en `true`).
2. Si utiliza el método `setMask()` y la máscara tiene una caché de mapa de bits, Flash realiza una mezcla de alfa entre las dos imágenes.
3. A continuación se aplican los filtros (desenfoque, sombra, iluminado, etc.)
4. Si utiliza la clase ColorTransform, se realiza la operación de transformación de color y se almacena en caché como resultado de mapa de bits.
5. Si aplica un modo de mezcla, la mezcla se realiza en este momento (mediante un procesador de vectores).
6. Si aplica capas de máscaras externas, las capas ejecutan las máscaras (mediante un procesador de vectores).

# Dibujo con código ActionScript

Puede utilizar métodos de la clase MovieClip para trazar líneas y rellenos en el escenario. Esto permite crear herramientas de dibujo para los usuarios y dibujar formas en el archivo SWF como respuesta a los eventos. Los siguientes son los métodos de dibujo de la clase MovieClip:

- `beginFill()`
- `beginGradientFill()`
- `clear()`
- `curveTo()`
- `endFill()`
- `lineTo()`
- `lineStyle()`
- `moveTo()`

Puede utilizar estos métodos de dibujo con cualquier clip de película. Sin embargo, si utiliza los métodos de dibujo con un clip de película que se ha creado en modo de edición, los métodos se ejecutarán antes de dibujar el clip. Es decir, el contenido creado en modo de edición se dibuja encima del contenido dibujado con los métodos de dibujo.

Puede utilizar clips de película con métodos de dibujo como máscaras; sin embargo, al igual que en todas las máscaras de clip de película, los trazos se pasan por alto.

Para más información sobre el dibujo con ActionScript, consulte los siguientes temas:

- "Utilización de métodos de dibujo para dibujar líneas, curvas y figuras" en la página 575
- "Dibujo de formas específicas" en la página 577
- "Utilización de rellenos con degradado complejos" en la página 580
- "Utilización de estilos de línea" en la página 581
- "Utilización de animaciones mediante script y métodos de la interfaz API de dibujo" en la página 588

Puede encontrar un ejemplo de archivo de origen, drawingapi.fla, en la carpeta Samples del disco duro, que muestra cómo utilizar la interfaz API de dibujo en una aplicación de Flash.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\DrawingAPI.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/DrawingAPI.

## Utilización de métodos de dibujo para dibujar líneas, curvas y figuras

Puede utilizar la interfaz API de dibujo para crear dinámicamente formas en el escenario durante la ejecución. El contenido de estas formas se puede enmascarar dinámicamente, además de aplicarles filtros o animarlas por el escenario. También puede emplear la interfaz API de dibujo para crear distintas herramientas de dibujo, que permiten utilizar el ratón o el teclado para dibujar formas en el archivo SWF.

### Para dibujar una línea:

1. Cree un nuevo documento de Flash y guárdelo como **line.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
this.createEmptyMovieClip("line_mc", 10);
line_mc.lineStyle(1, 0x000000, 100);
line_mc.moveTo(0, 0);
line_mc.lineTo(200, 100);
line_mc._x = 100;
line_mc._y = 100;
```

   Este código dibuja una línea en el escenario desde 0,0 a 200,100. A continuación, sus coordenadas `_x` e `_y` se modifican para colocarla en 100,100.
3. Guarde el documento de Flash y seleccione Control > Probar película para probar el archivo SWF.

Para dibujar una forma más compleja, siga llamando al método `MovieClip.lineTo()` y dibuje un rectángulo, un cuadrado o un óvalo, como se muestra en el siguiente procedimiento.

### Para dibujar una curva:

1. Cree un nuevo documento de Flash y guárdelo como **curve.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
this.createEmptyMovieClip("circle_mc", 1);
with (circle_mc) {
  lineStyle(4, 0x000000, 100);
  beginFill(0xFF0000);
  moveTo(200, 300);
  curveTo(300, 300, 300, 200);
  curveTo(300, 100, 200, 100);
  curveTo(100, 100, 100, 200);
  curveTo(100, 300, 200, 300);
  endFill();
}
```

3. Guarde el documento de Flash y seleccione Control > Probar película para probarlo.

   Este código utiliza la interfaz API de dibujo para dibujar un círculo en el escenario. La forma de círculo sólo emplea cuatro llamadas al método `MovieClip.curveTo()` y por tanto puede aparecer algo distorsionada. Para ver otro ejemplo que utiliza la interfaz API de dibujo para crear un círculo, consulte el procedimiento al respecto en "Dibujo de formas específicas" en la página 577 para obtener un código que utiliza ocho llamadas al método `MovieClip.curveTo()` para que el círculo resulte más realista.

### Para dibujar un triángulo:

1. Cree un nuevo documento de Flash y guárdelo como **triangle.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
this.createEmptyMovieClip("triangle_mc", 1);
```

   Este código utiliza el método `MovieClip.createEmptyMovieClip()` para crear un clip de película vacío en el escenario. El nuevo clip de película es un elemento secundario de un clip de película existente (en este caso, de la línea de tiempo principal).

3. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo, detrás del código añadido en el paso anterior:

```
with (triangle_mc) {
  lineStyle(5, 0xFF00FF, 100);
  moveTo(200, 200);
  lineTo(300, 300);
  lineTo(100, 300);
  lineTo(200, 200);
}
```

   En este código, el clip de película vacío (`triangle_mc`) llama a los métodos de dibujo. Este código dibuja un triángulo con líneas de color púrpura de 5 píxeles y sin relleno.

4. Guarde el documento de Flash y seleccione Control > Probar película para probarlo.

Para obtener información detallada sobre estos métodos, consulte las entradas correspondientes en `%{MovieClip}%` en *Referencia del lenguaje ActionScript 2.0*.

Puede encontrar un ejemplo de archivo de origen, drawingapi.fla, en la carpeta Samples del disco duro, que muestra cómo utilizar la interfaz API de dibujo en una aplicación de Flash.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\DrawingAPI.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/DrawingAPI.

## Dibujo de formas específicas

En esta sección se muestra cómo crear métodos más flexibles que permiten dibujar formas más avanzadas, como rectángulos redondeados y círculos.

### Para crear un rectángulo:

1. Cree un nuevo documento de Flash y guárdelo como **rect.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
this.createEmptyMovieClip("rectangle_mc", 10);
rectangle_mc._x = 100;
rectangle_mc._y = 100;
drawRectangle(rectangle_mc, 240, 180, 0x99FF00, 100);
function drawRectangle(target_mc:MovieClip, boxWidth:Number,
  boxHeight:Number, fillColor:Number, fillAlpha:Number):Void {
  with (target_mc) {
    beginFill(fillColor, fillAlpha);
    moveTo(0, 0);
    lineTo(boxWidth, 0);
    lineTo(boxWidth, boxHeight);
    lineTo(0, boxHeight);
    lineTo(0, 0);
    endFill();
  }
}
```

3. Guarde el documento de Flash y seleccione Control > Probar película para probarlo.

   Flash dibuja un rectángulo verde sencillo en el escenario y lo coloca en 100,100. Para cambiar sus dimensiones o su color de relleno o transparencia, puede cambiar dichos valores en la llamada al método `drawRectangle()` en lugar de tener que modificar el contenido de `MovieClip.beginFill()`.

También puede crear un rectángulo con las esquinas redondeadas mediante la interfaz API de dibujo, como muestra el procedimiento siguiente.

### Para crear un rectángulo redondeado:

1. Cree un nuevo documento de Flash y guárdelo como **roundrect.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
this.createEmptyMovieClip("rectangle_mc", 10);
rectangle_mc._x = 100;
rectangle_mc._y = 100;
drawRoundedRectangle(rectangle_mc, 240, 180, 20, 0x99FF00, 100);
function drawRoundedRectangle(target_mc:MovieClip, boxWidth:Number,
  boxHeight:Number, cornerRadius:Number, fillColor:Number,
  fillAlpha:Number):Void {
  with (target_mc) {
    beginFill(fillColor, fillAlpha);
    moveTo(cornerRadius, 0);
    lineTo(boxWidth - cornerRadius, 0);
    curveTo(boxWidth, 0, boxWidth, cornerRadius);
    lineTo(boxWidth, cornerRadius);
    lineTo(boxWidth, boxHeight - cornerRadius);
    curveTo(boxWidth, boxHeight, boxWidth - cornerRadius, boxHeight);
    lineTo(boxWidth - cornerRadius, boxHeight);
    lineTo(cornerRadius, boxHeight);
    curveTo(0, boxHeight, 0, boxHeight - cornerRadius);
    lineTo(0, boxHeight - cornerRadius);
    lineTo(0, cornerRadius);
    curveTo(0, 0, cornerRadius, 0);
    lineTo(cornerRadius, 0);
    endFill();
  }
}
```

3. Guarde el documento de Flash y seleccione Control > Probar película para probarlo.

   Aparece un rectángulo de color verde en el escenario que tiene 240 píxeles de ancho por 180 de alto y esquinas redondeadas de 20-píxeles. Puede generar varias instancias de los rectángulos redondeados creando nuevos clips de película con `MovieClip.createEmptyMovieClip()` y llamando a la función personalizada `drawRoundedRectangle()`.

Asimismo, puede crear un círculo perfecto mediante la interfaz API de dibujo como se muestra en el siguiente procedimiento.

### Para crear un círculo:

1. Cree un nuevo documento de Flash y guárdelo como **circle2.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
this.createEmptyMovieClip("circle_mc", 10);
circle_mc._x = 100;
circle_mc._y = 100;
drawCircle(circle_mc, 100, 0x99FF00, 100);

function drawCircle(target_mc:MovieClip, radius:Number,
  fillColor:Number, fillAlpha:Number):Void {
  var x:Number = radius;
  var y:Number = radius;
  with (target_mc) {
    beginFill(fillColor, fillAlpha);
    moveTo(x + radius, y);
    curveTo(radius + x, Math.tan(Math.PI / 8) * radius + y,
  Math.sin(Math.PI / 4) * radius + x, Math.sin(Math.PI / 4) * radius +
  y);
    curveTo(Math.tan(Math.PI / 8) * radius + x, radius + y, x, radius +
  y);
    curveTo(-Math.tan(Math.PI / 8) * radius + x, radius+ y, -
  Math.sin(Math.PI / 4) * radius + x, Math.sin(Math.PI / 4) * radius +
  y);
    curveTo(-radius + x, Math.tan(Math.PI / 8) * radius + y, -radius +
  x, y);
    curveTo(-radius + x, -Math.tan(Math.PI / 8) * radius + y, -
  Math.sin(Math.PI / 4) * radius + x, -Math.sin(Math.PI / 4) * radius +
  y);
    curveTo(-Math.tan(Math.PI / 8) * radius + x, -radius + y, x, -radius
  + y);
    curveTo(Math.tan(Math.PI / 8) * radius + x, -radius + y,
  Math.sin(Math.PI / 4) * radius + x, -Math.sin(Math.PI / 4) * radius +
  y);
    curveTo(radius + x, -Math.tan(Math.PI / 8) * radius + y, radius + x,
  y);
    endFill();
  }
}
```

3. Guarde el documento de Flash y seleccione Control > Probar película para probar el archivo SWF.

Este código crea un círculo más complejo y realista que el del ejemplo anterior. En lugar de utilizar sólo cuatro llamadas a `curveTo()`, emplea ocho llamadas a este método, lo que da a la forma una apariencia mucho más redondeada.

La interfaz API de dibujo también permite crear un triángulo, como se muestra en el siguiente procedimiento.

### Para crear un triángulo elaborado:

1. Cree un nuevo documento de Flash y guárdelo como **fancytriangle.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
this.createEmptyMovieClip("triangle_mc", 10);
triangle_mc._x = 100;
triangle_mc._y = 100;
drawTriangle(triangle_mc, 100, 0x99FF00, 100);

function drawTriangle(target_mc:MovieClip, sideLength:Number,
  fillColor:Number, fillAlpha:Number):Void {
  var tHeight:Number = sideLength * Math.sqrt(3) / 2;
  with (target_mc) {
    beginFill(fillColor, fillAlpha);
    moveTo(sideLength / 2, 0);
    lineTo(sideLength, tHeight);
    lineTo(0, tHeight);
    lineTo(sideLength / 2, 0);
    endFill();
  }
}
```

   La interfaz API de dibujo dibuja un triángulo equilátero y lo rellena con el color y la cantidad de transparencia alfa especificados.
3. Guarde el documento de Flash y seleccione Control > Probar película para probarlo.

Puede encontrar un ejemplo de archivo de origen, drawingapi.fla, en la carpeta Samples del disco duro, que muestra cómo utilizar la interfaz API de dibujo en una aplicación de Flash.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\DrawingAPI.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/DrawingAPI.

## Utilización de rellenos con degradado complejos

La interfaz API de dibujo admite rellenos con degradado y continuos. El siguiente procedimiento crea un nuevo clip de película en el escenario, utiliza la interfaz API de dibujo para crear un cuadrado y lo rellena con un degradado lineal rojo y azul.

### Para crear un degradado complejo:

1. Cree un nuevo documento de Flash y guárdelo como **radialgradient.fla**.

2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
this.createEmptyMovieClip("gradient_mc", 10);
var fillType:String = "radial";
var colors:Array = [0xFF0000, 0x0000FF];
var alphas:Array = [100, 100];
var ratios:Array = [0, 0xFF];
var matrix:Object = {a:200, b:0, c:0, d:0, e:200, f:0, g:200, h:200,
  i:1};
var spreadMethod:String = "reflect";
var interpolationMethod:String = "linearRGB";
var focalPointRatio:Number = 0.9;
with (gradient_mc) {
  beginGradientFill(fillType, colors, alphas, ratios, matrix,
  spreadMethod, interpolationMethod, focalPointRatio);
  moveTo(100, 100);
  lineTo(100, 300);
  lineTo(300, 300);
  lineTo(300, 100);
  lineTo(100, 100);
  endFill();
}
```

   El código ActionScript anterior utiliza la interfaz API de dibujo para crear un cuadrado en el escenario y llama al método beginGradientFill() para rellenarlo con un degradado circular rojo y azul.

3. Guarde el documento de Flash y seleccione Control > Probar película para ver el archivo de Flash.

## Utilización de estilos de línea

La interfaz API de dibujo permite especificar un estilo de línea que Flash emplea para posteriores llamadas a `MovieClip.lineTo()` y `MovieClip.curveTo()` hasta que se llama a `MovieClip.lineStyle()` con distintos parámetros, como se muestra a continuación:

```
lineStyle(thickness:Number, rgb:Number, alpha:Number, pixelHinting:Boolean,
  noScale:String, capsStyle:String, jointStyle:String, miterLimit:Number)
```

Puede llamar a `MovieClip.lineStyle()` en mitad de un trazado para especificar diferentes estilos para los distintos segmentos de línea de un trazado.

Para más información sobre el uso de ActionScript para establecer estilos de línea, consulte las secciones siguientes:

- "Definición de estilos de trazo y extremos" en la página 582
- "Definición de parámetros de estilos de línea" en la página 583

## Definición de estilos de trazo y extremos

Flash 8 incluye varias mejoras para el dibujo de líneas. Entre los nuevos parámetros añadidos en Flash Player 8 se encuentran los siguientes: `pixelHinting`, `noScale`, `capsStyle`, `jointStyle` y `miterLimit`.

En el siguiente procedimiento se muestra la diferencia existente entre los tres nuevos estilos de extremos de Flash Player 8: `none`, `round` y `square`.

### Para definir estilos de extremos mediante código ActionScript:

1. Cree un nuevo documento de Flash y guárdelo como **capstyle.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
// Definir cuadrícula de clip de película.
this.createEmptyMovieClip("grid_mc", 50);
grid_mc.lineStyle(0, 0x999999, 100);
grid_mc.moveTo(50, 0);
grid_mc.lineTo(50, Stage.height);
grid_mc.moveTo(250, 0);
grid_mc.lineTo(250, Stage.height);
// línea 1 (capsStyle: round)
this.createEmptyMovieClip("line1_mc", 10);
with (line1_mc) {
  createTextField("label_txt", 1, 5, 10, 100, 20);
  label_txt.text = "round";
  lineStyle(20, 0x99FF00, 100, true, "none", "round", "miter", 0.8);
  moveTo(0, 0);
  lineTo(200, 0);
  _x = 50;
  _y = 50;
}
// línea 2 (capsStyle: square)
this.createEmptyMovieClip("line2_mc", 20);
with (line2_mc) {
  createTextField("label_txt", 1, 5, 10, 100, 20);
  label_txt.text = "square";
  lineStyle(20, 0x99FF00, 100, true, "none", "square", "miter", 0.8);
  moveTo(0, 0);
  lineTo(200, 0);
  _x = 50;
  _y = 150;
}
// línea 3 (capsStyle: none)
this.createEmptyMovieClip("line3_mc", 30);
with (line3_mc) {
  createTextField("label_txt", 1, 5, 10, 100, 20);
  label_txt.text = "none";
  lineStyle(20, 0x99FF00, 100, true, "none", "none", "miter", 0.8);
  moveTo(0, 0);
  lineTo(200, 0);
  _x = 50;
  _y = 250;
}
```

El código anterior crea dinámicamente cuatro clips de película y utiliza la interfaz API de dibujo para crear una serie de líneas en el escenario. El primer clip de película contiene dos líneas verticales, una a 50 píxeles y la otra a 250 píxeles en el eje x. Cada uno de los tres clips de película siguientes dibuja una línea verde en el escenario y define su propiedad capsStyle como round, square o none.

3. Seleccione Control > Probar película para probar el documento.

   Los distintos estilos de extremos aparecen en el escenario durante la ejecución.

## Definición de parámetros de estilos de línea

Es posible definir los parámetros de los estilos de línea para cambiar la apariencia de los trazos. Puede utilizar dichos parámetros para cambiar el grosor, el color, el valor alfa, la escala y otros atributos del estilo de línea.

### Definición del grosor de línea

El parámetro `thickness` del método `MovieClip.lineStyle()` permite especificar el grosor de la línea dibujada en puntos como número. Puede dibujar una línea de cualquier grosor entre 0 y 255 puntos de ancho, si bien el grosor 0 crea lo que se denomina un *grosor muy fino*, en el que el trazo es siempre de 1 píxel, independientemente de si el archivo SWF se ha acercado o cambiado de tamaño.

En el siguiente procedimiento se muestra la diferencia entre una línea con un grosor estándar de 1 píxel y con un grosor muy fino.

#### Para crear un trazo de grosor muy fino:

1. Cree un nuevo documento de Flash y guárdelo como **hairline.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
this.createEmptyMovieClip("drawing_mc", 10);
// crear una línea roja de grosor muy fino
drawing_mc.lineStyle(0, 0xFF0000, 100);
drawing_mc.moveTo(0, 0);
drawing_mc.lineTo(200, 0);
drawing_mc.lineTo(200, 100);
// crear una línea azul con un grosor de 1 píxel
drawing_mc.lineStyle(1, 0x0000FF, 100);
drawing_mc.lineTo(0, 100);
drawing_mc.lineTo(0, 0);
drawing_mc._x = 100;
drawing_mc._y = 100;
```

   El código anterior utiliza la interfaz API de dibujo para dibujar dos líneas en el escenario. La primera línea es roja y tiene un grosor de 0, que indica el grosor muy fino; la segunda línea es azul y tiene un grosor de 1 píxel.

3. Guarde el documento de Flash y seleccione Control > Probar película para probar el archivo SWF.

   Inicialmente, tanto la línea roja como la azul tienen exactamente la misma apariencia. Si hace clic con el botón derecho en el archivo SWF y selecciona Acercar en el menú contextual, la línea roja siempre aparece como de 1 píxel; sin embargo, la azul se hace cada vez más larga a medida que se acerca el archivo SWF.

## Definición del color de línea (rgb)

El segundo parámetro del método `lineStyle()`, `rgb`, permite controlar el color del segmento de línea actual como un número. De forma predeterminada, Flash dibuja líneas negras (`#000000`), aunque se pueden especificar distintos colores estableciendo un nuevo valor de color hexadecimal con la sintaxis `0xRRGGBB`. En esta sintaxis, `RR` es el valor rojo (entre `00` y `FF`), `GG` es el verde (`00` a `FF`) y `BB` es el azul (`00` a `FF`).

Por ejemplo, se representa una línea roja como `0xFF0000`, una verde como `0x00FF00`, una azul como `0x0000FF`, una púrpura como `0xFF00FF` (roja y azul), una blanca como `#FFFFFF`, una gris como `#999999`, y así sucesivamente.

## Definición del valor alfa de línea

El tercer parámetro en el método `lineStyle()`, `alpha`, permite controlar el nivel de transparencia (alfa) de la línea. La transparencia es un valor numérico entre 0 y 100, donde 0 representa una línea completamente transparente y 100 una completamente opaca (visible).

## Definición de consejos de píxeles de línea (pixelHinting)

El uso de consejos de píxeles para el parámetro de trazos, `pixelHinting`, implica que los anclajes de líneas y curvas se establecen en píxeles exactos. Los trazos se dibujan a píxeles exactos en cualquier grosor, lo que significa que nunca se verá una línea vertical u horizontal imprecisa. El parámetro `pixelHinting` se establece en un valor booleano (`true` o `false`).

## Definición de la escala de línea (noScale)

El parámetro `noScale` se establece mediante un valor String que permite especificar un modo de escala para la línea. Se puede utilizar un trazo no escalable en el modo horizontal o vertical, escalar la línea (normal) o no utilizar la escala.

**SUGERENCIA**

Resulta útil activar la escala para los elementos de la interfaz de usuario cuando se acerca el contenido, pero no si sólo se escala un clip de película vertical u horizontalmente.

Puede emplear uno de los cuatro modos disponibles para especificar cuándo se debe realizar la escala y cuándo no. Los siguientes son los posibles valores de la propiedad `noScale`:

`normal` Siempre cambia la escala del grosor (*valor predeterminado*).

`vertical` No cambia la escala del grosor si el objeto sólo cambia de escala vertical.

`horizontal` No cambia la escala del grosor si el objeto sólo cambia de escala horizontal.

`none` Nunca cambia la escala del grosor.

## Definición de extremos (capsStyle) y juntas (jointStyle) de línea

Puede establecer tres tipos de estilos de extremo para el parámetro `capsStyle`:

- `round` (*valor predeterminado*)
- `square`
- `none`

En el siguiente procedimiento se muestran las diferencias entre los tres estilos de extremos. Cuando se prueba el archivo SWF, aparece en el escenario una representación visual de cada estilo.

### Para establecer distintos estilos de extremos:

1. Cree un nuevo documento de Flash y guárdelo como **capsstyle2.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
var lineLength:Number = 100;
// round
this.createEmptyMovieClip("round_mc", 10);
round_mc.lineStyle(20, 0xFF0000, 100, true, "none", "round");
round_mc.moveTo(0, 0);
round_mc.lineTo(lineLength, 0);
round_mc.lineStyle(0, 0x000000);
round_mc.moveTo(0, 0);
round_mc.lineTo(lineLength, 0);
round_mc._x = 50;
round_mc._y = 50;
var lbl:TextField = round_mc.createTextField("label_txt", 10, 0, 10,
  lineLength, 20);
lbl.text = "round";
var lineLength:Number = 100;
// square
this.createEmptyMovieClip("square_mc", 20);
square_mc.lineStyle(20, 0xFF0000, 100, true, "none", "square");
square_mc.moveTo(0, 0);
square_mc.lineTo(lineLength, 0);
square_mc.lineStyle(0, 0x000000);
square_mc.moveTo(0, 0);
square_mc.lineTo(lineLength, 0);
square_mc._x = 200;
square_mc._y = 50;
```

```
var lbl:TextField = square_mc.createTextField("label_txt", 10, 0, 10,
  lineLength, 20);
lbl.text = "square";
// none
this.createEmptyMovieClip("none_mc", 30);
none_mc.lineStyle(20, 0xFF0000, 100, true, "none", "none");
none_mc.moveTo(0, 0);
none_mc.lineTo(lineLength, 0);
none_mc.lineStyle(0, 0x000000);
none_mc.moveTo(0, 0);
none_mc.lineTo(lineLength, 0);
none_mc._x = 350;
none_mc._y = 50;
var lbl:TextField = none_mc.createTextField("label_txt", 10, 0, 10,
  lineLength, 20);
lbl.text = "none";
```

El código anterior utiliza la interfaz API de dibujo para dibujar tres líneas, cada una de ellas con un valor distinto para capsStyle.

3. Seleccione Control > Probar película para probar el documento de Flash.

Se pueden establecer los siguientes tres tipos de estilos de junta para el parámetro `jointStyle`:

- `round` (*valor predeterminado*)
- `miter`
- `bevel`

En el siguiente procedimiento se muestran las diferencias entre los tres estilos de junta.

### Para establecer distintos estilos de junta:

1. Cree un nuevo documento de Flash y guárdelo como **jointstyles.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
var lineLength:Number = 100;
// miter
this.createEmptyMovieClip("miter_mc", 10);
miter_mc.lineStyle(25, 0xFF0000, 100, true, "none", "none", "miter",
  25);
miter_mc.moveTo(0, lineLength);
miter_mc.lineTo(lineLength / 2, 0);
miter_mc.lineTo(lineLength, lineLength);
miter_mc.lineTo(0, lineLength);
miter_mc._x = 50;
miter_mc._y = 50;
var lbl:TextField = miter_mc.createTextField("label_txt", 10, 0,
  lineLength + 20, lineLength, 20);
lbl.autoSize = "center";
lbl.text = "miter";
```

```
// round
this.createEmptyMovieClip("round_mc", 20);
round_mc.lineStyle(25, 0xFF0000, 100, true, "none", "none", "round");
round_mc.moveTo(0, lineLength);
round_mc.lineTo(lineLength / 2, 0);
round_mc.lineTo(lineLength, lineLength);
round_mc.lineTo(0, lineLength);
round_mc._x = 200;
round_mc._y = 50;
var lbl:TextField = round_mc.createTextField("label_txt", 10, 0,
  lineLength + 20, lineLength, 20);
lbl.autoSize = "center";
lbl.text = "round";
// bevel
this.createEmptyMovieClip("bevel_mc", 30);
bevel_mc.lineStyle(25, 0xFF0000, 100, true, "none", "none", "bevel");
bevel_mc.moveTo(0, lineLength);
bevel_mc.lineTo(lineLength / 2, 0);
bevel_mc.lineTo(lineLength, lineLength);
bevel_mc.lineTo(0, lineLength);
bevel_mc._x = 350;
bevel_mc._y = 50;
var lbl:TextField = bevel_mc.createTextField("label_txt", 10, 0,
  lineLength + 20, lineLength, 20);
lbl.autoSize = "center";
lbl.text = "bevel";
```

Flash utiliza la interfaz API de dibujo para dibujar tres triángulos en el escenario. Cada uno de ellos presenta un valor distinto para su estilo de junta.

3. Guarde el documento de Flash y seleccione Control > Probar película para probarlo.

## Definición del angular de línea (miterLimit)

La propiedad `miterLimit` es un valor numérico que indica el límite en que se corta una junta de angular (consulte "Definición de extremos (capsStyle) y juntas (jointStyle) de línea" en la página 585). El valor `miterLimit` es un multiplicador general de un trazo. Por ejemplo, con un valor de 2,5, `miterLimit` se corta con un tamaño 2,5 veces superior al del trazo. Los valores válidos van del 0 al 255 (si se establece un valor `miterLimit` como `undefined`, el valor predeterminado es `3`). La propiedad `miterLimit` sólo se utiliza si `jointStyle` se establece en `miter`.

# Utilización de animaciones mediante script y métodos de la interfaz API de dibujo

Puede combinar la interfaz API de dibujo con las clases Tween y TransitionManager para crear excelentes resultados de animación, para lo que se debe escribir muy poco código ActionScript.

En el siguiente procedimiento se carga una imagen JPEG y se crea dinámicamente una máscara de la misma para que la imagen se muestre lentamente mediante la interpolación de la máscara.

### Para animar máscaras dinámicas:

1. Cree un nuevo documento de Flash y guárdelo como **dynmask.fla**.
2. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

```
import mx.transitions.Tween;
import mx.transitions.easing.*;
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  target_mc._visible = false;
  // Centrar la imagen en el escenario.
  target_mc._x = (Stage.width - target_mc._width) / 2;
  target_mc._y = (Stage.height - target_mc._height) / 2;
  var maskClip:MovieClip = target_mc.createEmptyMovieClip("mask_mc",
  20);
  with (maskClip) {
    // Dibujar una máscara del mismo tamaño que la imagen cargada.
    beginFill(0xFF00FF, 100);
    moveTo(0, 0);
    lineTo(target_mc._width, 0);
    lineTo(target_mc._width, target_mc._height);
    lineTo(0, target_mc._height);
    lineTo(0, 0);
    endFill();
  }
  target_mc.setMask(maskClip);
  target_mc._visible = true;
  var mask_tween:Object = new Tween(maskClip, "_yscale", Strong.easeOut,
  0, 100, 2, true);
};
this.createEmptyMovieClip("img_mc", 10);
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mclListener);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img_mc);
```

Este ejemplo de código importa la clase Tween y cada una de las clases del paquete de suavizado. A continuación, crea un objeto que actúa como objeto detector para una instancia de MovieClipLoader que se crea en una sección posterior del código. El objeto detector define un detector de evento único, `onLoadInit`, que centra la imagen JPEG cargada dinámicamente en el escenario. Una vez que el código cambia la posición de la imagen, se crea una nueva instancia de clip de película dentro del clip `target_mc` (que contiene la imagen JPEG cargada dinámicamente). El código de la interfaz API de dibujo dibuja un rectángulo con las mismas dimensiones que la imagen JPEG dentro del nuevo clip de película. Este último enmascara la imagen JPEG llamando al método `MovieClip.setMask()`. Una vez que la máscara se ha dibujado y definido, emplea la clase Tween para crear la animación, lo que hace que la imagen se muestre lentamente.

3. Guarde el documento de Flash y seleccione Control > Probar película para probar el archivo SWF.

| NOTA | Para animar el valor `_alpha` en el ejemplo anterior y no `_yscale`, interpole `target_mc` directamente en lugar del clip de película de máscara. |
|---|---|

Puede encontrar un ejemplo de archivo de origen, drawingapi.fla, en la carpeta Samples del disco duro, que muestra cómo utilizar la interfaz API de dibujo en una aplicación de Flash.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\DrawingAPI.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/DrawingAPI.

# Aspectos básicos de la escala y las guías de división

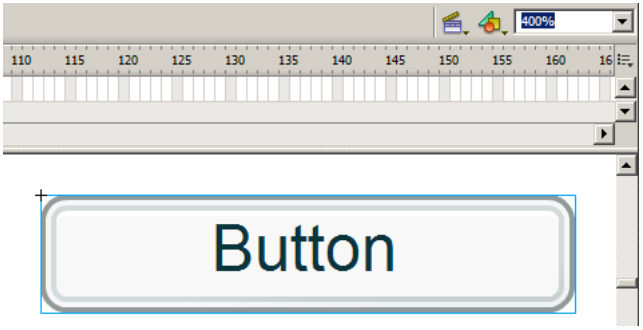
Puede utilizar la escala en 9 divisiones (Scale-9) para especificar un escalado en estilo de componente para los clips de película. Esta escala le permite crear símbolos de clip de película que se escalan debidamente como componentes de interfaz de usuario en lugar de utilizar el tipo de escala que normalmente se aplica a los gráficos y los elementos de diseño.

## Aspectos básicos del funcionamiento de la escala en 9 divisiones

La forma más sencilla de explicar cómo funciona la escala en 9 divisiones es ver un ejemplo en Flash.

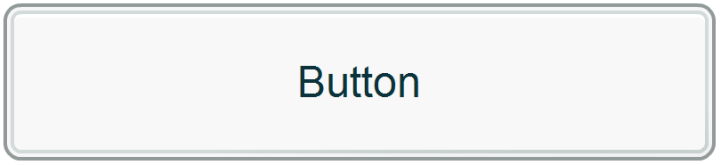
### Para conocer el funcionamiento de la escala en Flash:

1. Cree un nuevo documento de Flash y guárdelo como **dynmask.fla**.
2. Arrastre al escenario una copia del componente Button desde el panel Componentes (Ventana > Componentes).
3. Aumente el nivel de zoom del escenario a 400% con la herramienta Zoom.



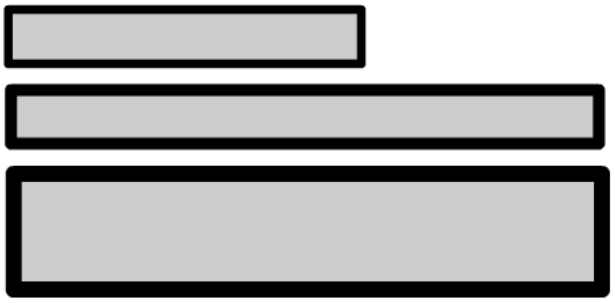

De forma predeterminada, la instancia del componente Button tiene 100 píxeles de ancho y 22 píxeles de alto.

4. Cambie su tamaño a 200 píxeles de ancho por 44 píxeles de alto mediante el inspector de propiedades.




Observe que aunque se ha cambiado el tamaño del componente, la etiqueta de texto y el borde de Button no se distorsionan. La etiqueta del botón permanece centrada y mantiene su tamaño de fuente. Aunque los componentes de la versión 2 de la arquitectura de componentes de Macromedia no utilizan una escala en 9 divisiones, los componentes gestionan la escala en dicha versión de forma que los contornos no cambien de tamaño (como se muestra en la siguiente ilustración).

Imagine que la instancia de botón se divide en nueve partes independientes o en una cuadrícula de 3 x 3 similar a un teclado numérico de teléfono u ordenador. Cuando se cambia el tamaño de la instancia de botón horizontalmente, sólo se expanden los tres segmentos verticales del centro (los números 2, 5 y 8 del teclado) para que el contenido no aparezca distorsionado. Si se cambia el tamaño verticalmente, sólo se hace en los tres segmentos horizontales del centro (los números 4, 5 y 6 del teclado). Las cuatro esquinas de la cuadrícula de escala no se escalan, lo que permite que el componente aumente de tamaño sin parecer que se ha expandido (consulte las imágenes siguientes).



**SUGERENCIA**

Los trazos se crean desde los bordes tras la transformación de escala en 9 divisiones y por tanto no se deforman ni pierden detalles.

Puede activar guías para la escala en 9 divisiones en el entorno de Flash desde los cuadros de diálogo Convertir en símbolo o Propiedades de símbolo. La casilla de verificación Activar guías para escala en 9 divisiones sólo se encuentra disponible si se publica para Flash Player 8 y el comportamiento se establece como clip de película. Las guías de escala en 9 divisiones no están disponibles en versiones anteriores de Flash o si se crea un símbolo de gráfico o botón. La escala en 9 divisiones se puede activar en ActionScript estableciendo la propiedad `scale9Grid` en una instancia de clip de película.

Tanto si creó las guías de división con la interfaz de usuario o mediante ActionScript, puede trazar la coordenada *x*, la coordenada y, la anchura y la altura estableciendo la propiedad `scale9Grid` del clip de película.

```
trace(my_mc.scale9Grid); // (x=20, y=20, w=120, h=120)
```

Este fragmento de código traza el valor del objeto Rectangle que utiliza la propiedad `scale9Grid`. El rectángulo tiene unas coordenadas *x* e *y* de 20 píxeles, una anchura de 120 píxeles y una altura de 120 píxeles.

# Utilización de escala en 9 divisiones en ActionScript

En el siguiente ejemplo se utilizan las herramientas de dibujo para dibujar un cuadrado de 300 x 300 píxeles cuyo tamaño se cambia mediante la escala en 9 divisiones. El cuadrado se divide en nueve cuadrados más pequeños con un tamaño de aproximadamente 100 píxeles de ancho por 100 de alto. Cuando se cambia de tamaño, cada segmento que no está en una esquina se expande para que coincida con la anchura y la altura especificadas.

### Para utilizar la escala en 9 divisiones con ActionScript.

1. Cree un nuevo documento de Flash y guárdelo como **ninescale.fla**.
2. Arrastre un componente Button a la biblioteca del documento actual.
3. Seleccione la herramienta Rectángulo y dibuje en el escenario un cuadrado rojo (300 x 300 píxeles) con un trazo negro de 15 píxeles.
4. Seleccione la herramienta Óvalo y dibuje en el escenario un círculo púrpura (50 x 50 píxeles) con un trazo negro de 2 píxeles.
5. Seleccione dicho círculo y arrástrelo a la esquina superior izquierda del cuadrado rojo creado anteriormente.
6. Seleccione la herramienta Óvalo, dibuje un nuevo círculo de aproximadamente 200 x 200 píxeles y colóquelo en el escenario.
7. Seleccione dicho círculo y arrástrelo de forma que su punto central se encuentre en la esquina inferior izquierda del cuadrado.
8. Haga clic en la parte exterior de la instancia de círculo para anular la selección del mismo.
9. Vuelva a hacer doble clic en el círculo para seleccionarlo y presione la tecla Retroceso para eliminar la forma y quitar una parte circular del cuadrado.
10. Con el ratón, seleccione el cuadrado rojo y el círculo púrpura de su interior.
11. Presione F8 para convertir la forma en un símbolo de clip de película.
12. Asígnele el nombre de instancia **my_mc**.
13. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
import mx.controls.Button;
import flash.geom.Rectangle;

var grid:Rectangle = new Rectangle(100, 100, 100, 100);

var small_button:Button = this.createClassObject(Button, "small_button",
  10, {label:"Small"});
small_button.move(10, 10);
small_button.addEventListener("click", smallHandler);
function smallHandler(eventObj:Object):Void {
     my_mc._width = 100;
     my_mc._height = 100;
}
```

```
var large_button:Button = this.createClassObject(Button, "large_button",
  20, {label:"Large"});
large_button.move(120, 10);
large_button.addEventListener("click", largeHandler);
function largeHandler(eventObj:Object):Void {
    my_mc._width = 450;
    my_mc._height = 300;
}

var toggle_button:Button = this.createClassObject(Button,
  "toggle_button", 30, {label:"scale9Grid=OFF", toggle:true,
  selected:false});
toggle_button.move(420, 10);
toggle_button.setSize(120, 22);
toggle_button.addEventListener("click", toggleListener);
function toggleListener(eventObj:Object):Void {
  if (eventObj.target.selected) {
    eventObj.target.label = "scale9Grid=ON";
    my_mc.scale9Grid = grid;
  } else {
    eventObj.target.label = "scale9Grid=OFF";
    my_mc.scale9Grid = undefined;
  }
}
```

El código anterior se divide en cinco secciones independientes. La primera sección importa dos clases: mx.controls.Button (la clase de componente Button) y flash.geom.Rectangle. La segunda sección crea una nueva instancia de clase Rectangle y especifica las coordenadas *x* e *y* de 100 píxeles así como un ancho y un alto de 100 píxeles. Esta instancia de rectángulo se utiliza para establecer una cuadrícula de escala en 9 divisiones para una forma de clip de película que se creará posteriormente.

A continuación, cree una nueva instancia de componente Button y asígnele el nombre de instancia `small_button`. Siempre que haga clic en este botón, el tamaño del clip de película creado anteriormente cambia a 100 píxeles de ancho por 100 píxeles de alto. La cuarta sección del código crea dinámicamente una nueva instancia de Button llamada `large_button` que, cuando se hace clic en ella, cambia el tamaño del clip de película a 450 píxeles de ancho por 300 píxeles de alto. La última sección crea una nueva instancia de Button que el usuario puede activar o desactivar. Cuando el botón está activado, se aplica la cuadrícula de 9 divisiones. Si está desactivado, se desactiva la cuadrícula de 9 divisiones.

14. Guarde el documento de Flash y seleccione Control > Probar película para probar el archivo SWF.

    En este ejemplo de código se añaden y colocan tres instancias de componente Button en el escenario y se crea un detector de eventos para cada botón. Si hace clic en el botón grande con la cuadrícula de 9 divisiones desactivada, puede ver que la imagen se distorsiona y se muestra expandida. Active la cuadrícula haciendo clic en el conmutador y vuelva a hacer clic en el botón. Cuando la cuadrícula de 9 divisiones esté activada, el círculo de la esquina superior izquierda no debería aparecer distorsionado.

CAPÍTULO 14

# Creación de interacciones con ActionScript

14

En animaciones sencillas, Macromedia Flash Player reproduce las escenas y los fotogramas de un archivo SWF de forma secuencial. En un archivo SWF interactivo, los usuarios utilizan el teclado y el ratón para ir a distintas partes del archivo SWF, mover objetos, introducir información en formularios y llevar a cabo otras operaciones interactivas.

ActionScript sirve para crear scripts que indican a Flash Player la acción que debe llevar a cabo cuando ocurra un evento. Algunos eventos que desencadenan un script se producen cuando la cabeza lectora llega a un fotograma, cuando se carga o descarga un clip de película o cuando el usuario hace clic en un botón o presiona una tecla.

Los scripts pueden constar de un solo comando, como indicar a un archivo SWF que se detenga, o bien de una serie de comandos y sentencias, como comprobar primero una condición y, a continuación, realizar una acción. Muchos comandos de ActionScript son sencillos y permiten crear controles básicos para un archivo SWF. Otras requieren un cierto dominio de los lenguajes de programación y están pensadas para operaciones avanzadas.

Para más información sobre la creación de interacción con ActionScript, consulte los siguientes temas:

# Eventos e interacciones

Siempre que un usuario hace clic en el ratón o presiona una tecla, dicha acción genera un evento. Estos tipos de eventos suelen denominarse *eventos* de *usuario*, ya que se generan como respuesta a alguna acción llevada a cabo por el usuario. Puede escribir código de ActionScript para responder a estos eventos o *gestionarlos*. Por ejemplo, cuando un usuario hace clic en un botón, puede enviar la cabeza lectora a otro fotograma del archivo SWF o cargar una nueva página Web en el navegador.

En un archivo SWF, los botones, los clips de película y los campos de texto generan eventos a los que se puede responder. ActionScript dispone de tres modos de gestionar eventos: métodos de controlador de eventos, detectores de eventos y controladores `on()` y `onClipEvent()`. Para más información sobre eventos y su gestión, consulte el Capítulo 9, "Gestión de eventos".

# Control de la reproducción de archivos SWF

Las funciones de ActionScript siguientes permiten controlar la cabeza lectora en la línea de tiempo y cargar una nueva página Web en una ventana del navegador:

- Las funciones `gotoAndPlay()` y `gotoAndStop()` envían la cabeza lectora a otro fotograma o escena. Estas son las funciones globales a las que puede llamar desde cualquier script. También puede utilizar los métodos `MovieClip.gotoAndPlay()` y `MovieClip.gotoAndStop()` para desplazarse por la línea de tiempo de un objeto de clip de película específico. Consulte "Salto a fotogramas o escenas" en la página 597.
- Las acciones `play()` y `stop()` reproducen y detienen archivos SWF. Consulte "Reproducción y detección de clips de película" en la página 597.
- La acción `getURL()` lleva a otra URL. Consulte "Salto a otra URL" en la página 599.

Para más información, consulte los siguientes temas:

- "Salto a fotogramas o escenas" en la página 597
- "Reproducción y detección de clips de película" en la página 597
- "Salto a otra URL" en la página 599

## Salto a fotogramas o escenas

Para ir a una escena o un fotograma específicos del archivo SWF, puede utilizar las funciones globales `gotoAndPlay()` y `gotoAndStop()` o los métodos `MovieClip.gotoAndPlay()` y `MovieClip.gotoAndStop()` equivalentes de la clase MovieClip. Cada función o método permite especificar un fotograma de la escena actual. Si el documento contiene varias escenas, puede especificar una escena y un fotograma al que desea desplazarse.

En el ejemplo siguiente se utiliza la función global `gotoAndPlay()` en el controlador de eventos `onRelease` de un objeto de botón para enviar la cabeza lectora de la línea de tiempo que contiene el botón al fotograma 10.

```
jump_btn.onRelease = function () {
  gotoAndPlay(10);
};
```

En el ejemplo siguiente, el método `MovieClip.gotoAndStop()` envía la línea de tiempo de una instancia de clip de película llamada `categories_mc` al fotograma 10 y se detiene. Cuando se utilizan los métodos MovieClip `gotoAndPlay()` y `gotoAndStop()`, debe especificar la instancia a la que se aplica el método.

```
jump_btn.onPress = function () {
  categories_mc.gotoAndStop(10);
};
```

En el ejemplo final, la función `gotoAndStop()` global se utiliza para mover la cabeza lectora al fotograma 1 de la escena 2. Si no se ha especificado la escena, la cabeza lectora va al fotograma especificado en la escena actual. Puede utilizar el parámetro scene únicamente en la línea de tiempo de raíz, no en las líneas de tiempo de los clips de película u otros objetos del documento.

```
nextScene_mc.onRelease = function() {
  gotoAndStop("Scene 2", 1);
}
```

## Reproducción y detección de clips de película

A menos que se indique lo contrario, una vez que se inicia un archivo SWF se reproduce en todo el fotograma de la línea de tiempo. Puede iniciar o detener un archivo SWF mediante las funciones globales `play()` y `stop()` o los métodos MovieClip equivalentes. Por ejemplo, puede utilizar la función `stop()` para detener un archivo SWF al final de una escena antes de continuar con la escena siguiente. Una vez detenido el archivo, debe volver a iniciarse de forma explícita llamando a la función `play()` o `gotoAndPlay()`.

Puede utilizar las funciones `play()` y `stop()` o los métodos MovieClip para controlar la línea de tiempo principal o la línea de tiempo de cualquier clip de película o archivo SWF cargado. El clip de película que desea controlar debe tener un nombre de instancia y, además, debe estar presente en la línea de tiempo.

El controlador `on(press)` siguiente que se encuentra asociado a un botón inicia el movimiento de la cabeza lectora del archivo SWF o del clip de película que contiene el objeto de botón:

```
// Asociado a una instancia de botón
on (press) {
  // Reproduce la línea de tiempo que contiene el botón
  play();
}
```

El mismo código del controlador de eventos `on()` produce un resultado diferente al asociarlo a un objeto de clip de película, en lugar de a un botón. Al asociar dicho código a un objeto de botón, las sentencias que se creen en un controlador `on()` se aplican, de forma predeterminada, a la línea de tiempo que contiene el botón. Sin embargo, al asociarlas a un objeto de clip de película, las sentencias realizadas en un controlador `on()` se aplican al clip de película al que se encuentra asociado al controlador `on()`.

Por ejemplo, el siguiente código del controlador `onPress()` detiene la línea de tiempo del clip de película al que se encuentra asociado el controlador, no la línea de tiempo que contiene el clip de película:

```
//Asociado a la instancia del clip de película myMovie_mc
myMovie_mc.onPress() {
  stop();
};
```

Las mismas condiciones se aplican a los controladores `onClipEvent()` que se encuentran asociados a objetos de clip de película. Por ejemplo, el código siguiente detiene la línea de tiempo del clip de película que contiene el controlador `onClipEvent()` cuando el clip se carga inicialmente o cuando aparece en el escenario:

```
onClipEvent(load) {
  stop();
}
```

## Salto a otra URL

Para abrir una página Web en una ventana del navegador o para transferir datos a otra aplicación en una URL definida, puede utilizar la función global `getURL()` o el método `MovieClip.getURL()`. Por ejemplo, puede disponer de un botón que lleve a un nuevo sitio Web, o bien enviar variables de línea de tiempo a un script CGI para procesarlo igual que se procesaría un formulario HTML. También puede especificar la ventana que desea utilizar, del mismo modo que lo haría al elegir una ventana para usar con una etiqueta de anclaje HTML (`<a></a>`).

Por ejemplo, el código siguiente abre la página principal de macromedia.com en una ventana en blanco del navegador cuando el usuario hace clic en la instancia de botón llamada `homepage_btn`:

```
//Asociar a fotograma
homepage_btn.onRelease = function () {
  getURL("http://www.macromedia.com", "_blank");
};
```

También puede enviar variables junto con la URL, mediante los métodos `GET` o `POST`. Esto resulta útil si la página que está cargando desde un servidor de aplicaciones, como la página del servidor ColdFusion (CFM), espera recibir variables de formulario. Por ejemplo, suponga que desea cargar una página de CFM llamada addUser.cfm que espera obtener dos variables de formulario: first`Name` y `age`. Para ello, puede crear un clip de película llamado `variables_mc` que defina dichas variables tal como se muestra en el siguiente ejemplo:

```
variables_mc.firstName = "Francois";
variables_mc.age = 32;
```

A continuación, el código siguiente carga addUser.cfm en una ventana en blanco del navegador y pasa las variables `variables_mc.name` y `variables_mc.age` del encabezado `POST` a la página de CFM:

```
variables_mc.getURL("addUser.cfm", "_blank", "POST");
```

La funcionalidad de `getURL()` es dependiente del navegador que se utilice. La forma más fiable de hacer que todos los navegadores funcionen de la misma manera es llamar a una función de JavaScript en el código HTML que utiliza el método `window.open()` de JavaScript para abrir una ventana. Añada el siguiente código HTML y JavaScript en la plantilla HTML:

```
<script language="JavaScript">
<--
  function openNewWindow(myURL) {
    window.open(myURL, "targetWindow");
  }
// -->
</script>
```

Puede utilizar el siguiente código ActionScript para llamar a `openNewWindow` desde el archivo SWF:

```
var myURL:String = "http://foo.com";
getURL("javascript:openNewWindow('" + String(myURL) + "');");
```

Para más información, consulte %{getURL function}% en *Referencia del lenguaje ActionScript 2.0*.

# Creación de interactividad y efectos visuales

Para crear interactividad y otros efectos visuales, debe entender las técnicas siguientes:



## Creación de un puntero de ratón personalizado

Un puntero de ratón estándar es la representación en la pantalla del sistema operativo de la posición del ratón del usuario. Al reemplazar el puntero estándar por uno diseñado en Flash, puede integrar el movimiento del ratón del usuario en el archivo SWF con mayor precisión. En el ejemplo de esta sección se utiliza un puntero personalizado que parece una flecha grande. La clave de esta función, sin embargo, radica en la capacidad de convertir el puntero personalizado en cualquier cosa: por ejemplo, un balón que debe llevarse a la portería o una muestra de tela que se coloca sobre una silla para cambiarle el color.

Para crear un puntero personalizado, diseñe el clip de película de puntero en el escenario. A continuación, en ActionScript, oculte el puntero estándar y realice un seguimiento de su movimiento. Para ocultar el puntero estándar, utilice el método `hide()` de la clase Mouse incorporada (consulte %{hide (método Mouse.hide)}% en *Referencia del lenguaje ActionScript 2.0*).

### Para crear un puntero personalizado:

1. Cree un clip de película para utilizarlo como puntero personalizado y coloque una instancia del clip en el escenario.
2. Seleccione la instancia de clip de película en el escenario.
3. En el inspector de propiedades, escriba **cursor_mc** en el cuadro de texto Nombre de instancia.
4. Seleccione el fotograma 1 de la línea de tiempo y escriba el siguiente código en el panel Acciones:

```
Mouse.hide();
cursor_mc.onMouseMove = function() {
  this._x = _xmouse;
  this._y = _ymouse;
  updateAfterEvent();
};
```

   El método `Mouse.hide()` oculta el puntero cuando el clip de película aparece inicialmente en el escenario; la función `onMouseMove` sitúa el puntero personalizado en el mismo sitio que el puntero y llama a `updateAfterEvent()` siempre que el usuario mueve el ratón.

   La función `updateAfterEvent()` actualiza la pantalla inmediatamente después de que se haya producido el evento especificado, en lugar de cuando se dibuja el fotograma siguiente, que es lo que suele ocurrir. (Consulte %{updateAfterEvent function}% en *Referencia del lenguaje ActionScript 2.0*.)
5. Seleccione Control > Probar película para probar el puntero personalizado.

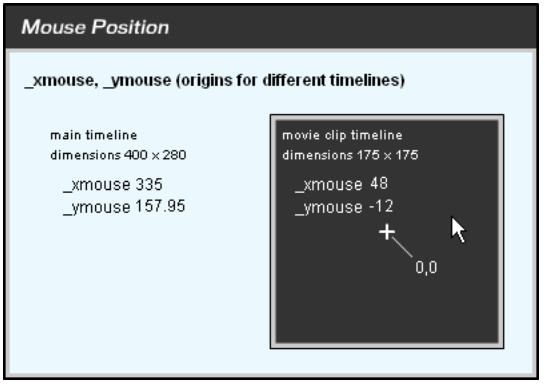
Los botones continúan funcionado al utilizar un puntero de ratón personalizado. Se recomienda colocar el puntero personalizado en la capa superior de la línea de tiempo de modo que se mueva por encima de los botones y de otros objetos cuando desplace el ratón por el archivo SWF. El puntero personalizado aparece encima de los botones y de otros objetos de otras capas. Además, la "punta" de un puntero de ratón personalizado es el punto de registro del clip de película que se utiliza como puntero personalizado. En consecuencia, si desea que una parte del clip de película actúe como la punta del puntero, configure las coordenadas del punto de registro del clip para que sea ese punto.

Para más información sobre los métodos de la clase Mouse, consulte %{Mouse}% en *Referencia del lenguaje ActionScript 2.0*.

# Obtención de la posición del puntero

Puede utilizar la propiedad `_xmouse` y la propiedad `_ymouse` para determinar la ubicación del puntero en un archivo SWF. Estas propiedades podrían emplearse, por ejemplo, en una aplicación de mapa que obtiene los valores de las propiedades `_xmouse` e `_ymouse` y los utiliza para calcular la longitud y la latitud de un lugar concreto.

Cada línea de tiempo tiene una propiedad `_xmouse` e `_ymouse` que indica la ubicación del puntero dentro de su sistema de coordenadas. La posición siempre es relativa al punto de registro. Para la línea de tiempo principal (`_level0`), el punto de registro corresponde a la esquina superior izquierda. En el caso de un clip de película, el punto de registro depende del punto de registro establecido cuando se creó el clip o su colocación en el escenario.




*Propiedades `_xmouse` e `_ymouse` de la línea de tiempo principal y de la línea de tiempo de un clip de película*

El siguiente procedimiento muestra diversas formas de obtener la posición del puntero dentro de la línea de tiempo principal o dentro de un clip de película.

### Para obtener la posición actual del puntero:

1. Cree dos campos de texto dinámicos y asígneles los nombres **box1_txt** y **box2_txt**.
2. Añada etiquetas para los cuadros de texto: *x* position e *y* position, respectivamente.
3. Seleccione Ventana > Acciones para abrir el panel Acciones en el caso de que no esté abierto.

4. Añada el código siguiente al panel Script:

```
var mouseListener:Object = new Object();
mouseListener.onMouseMove = function() {
  // devuelve la posición X e Y del ratón
  box1_txt.text = _xmouse;
  box2_txt.text = _ymouse;
};
Mouse.addListener(mouseListener);
```

5. Seleccione Control > Probar película para probar la película de Flash. Los campos box1_txt y box2_txt muestran la posición del puntero mientras lo desplaza por el escenario.

Para más información sobre las propiedades `_xmouse` e `_ymouse`, consulte %{_xmouse (propiedad MovieClip._xmouse)}% y %{_ymouse (propiedad MovieClip._ymouse)}% en *Referencia del lenguaje ActionScript 2.0*.

## Captura de teclas presionadas

Puede utilizar el controlador global `on()` para detectar el comportamiento incorporado de las teclas presionadas en Flash Player, como se muestra en el siguiente ejemplo:

```
/* Al presionar la tecla de flecha izquierda o derecha, el clip de película
  al que está asociado el controlador cambia de transparencia. */
on (keyPress "<Left>") {
  this._alpha -= 10;
}
on (keyPress "<Right>") {
  this._alpha += 10;
}
```

Asegúrese de que selecciona Control > Deshabilitar métodos abreviados de teclado, ya que, de lo contrario, algunas teclas con comportamiento incorporado no se sustituirán cuando utilice Control > Probar película para probar la aplicación. Consulte el parámetro `keyPress` de %{on handler}% en *Referencia del lenguaje ActionScript 2.0*.

Puede utilizar los métodos de la clase incorporada Key para detectar la última tecla que ha presionado el usuario. La clase Key no requiere una función constructora; para utilizar sus métodos, llame a los métodos de la clase, como se muestra en el ejemplo siguiente:

```
Key.getCode();
```

Puede obtener códigos de tecla virtuales o valores ASCII (código americano estándar para intercambio de información) de la tecla que haya sido presionada:

- Para obtener el código de tecla virtual de la última tecla presionada, utilice el método `getCode()`.
- Para obtener el valor ASCII de la última tecla presionada, utilice el método `getAscii()`.

Se asigna un código de tecla virtual a cada tecla física del teclado. Por ejemplo, la tecla de flecha izquierda tiene el código de tecla virtual 37. Al utilizar un código de tecla virtual, se garantiza que los controles del archivo SWF sean los mismos en todos los teclados, independientemente del idioma o de la plataforma.

Los valores ASCII se asignan a los primeros 127 caracteres de cada conjunto de caracteres. Los valores ASCII proporcionan información sobre un carácter de la pantalla. Por ejemplo, la letra "A" y la letra "a" tiene diferentes valores ASCII.

Para decidir qué teclas va a utilizar y determinar sus códigos virtuales, utilice alguno de los siguientes métodos:

- Consulte la lista de códigos de tecla del Apéndice C, "Teclas del teclado y valores de códigos de tecla".
- Utilice una constante de la clase Key. (En la caja de herramientas Acciones, haga clic en Clases de ActionScript 2.0 > Película > Tecla > Constantes.)
- Asigne el controlador `onClipEvent()` siguiente a un clip de película y seleccione Control > Probar película y presione la tecla que desee:

```
onClipEvent(keyDown) {
  trace(Key.getCode());
}
```

El código de la tecla deseada aparece en el panel Salida.

Un lugar habitual para utilizar los métodos de la clase Key es dentro de un controlador de eventos. En el ejemplo siguiente, el usuario mueve el coche mediante las teclas de flecha. El método `Key.isDown()` indica si la tecla que se está presionando es la flecha derecha, izquierda, hacia arriba o hacia abajo. El controlador de eventos, `Key.onKeyDown`, determina el valor `Key.isDown(keyCode)` de las sentencias `if`. En función del valor, el controlador da instrucciones a Flash Player para que actualice la posición del coche y muestre la dirección.

El ejemplo siguiente muestra cómo se pueden capturar teclas presionadas para mover un clip de película hacia arriba, abajo, la izquierda y la derecha en el escenario, en función de la tecla de flecha correspondiente que se presione (arriba, abajo, izquierda o derecha). Asimismo, un campo de texto muestra el nombre de la tecla que se presiona.

### Para crear un clip de película activado por teclado:

1. En el escenario, cree un clip de película que se moverá como respuesta a las flechas del teclado.

   En este ejemplo, el nombre de la instancia de clip de película es `car_mc`.

2. Seleccione el fotograma 1 en la línea de tiempo; a continuación, seleccione Ventana > Acciones para abrir el panel Acciones en caso de que todavía no aparezca en pantalla.
3. Para establecer cuánto avanza el coche en la pantalla cada vez que se presiona una tecla, defina una variable `distance` y establezca el valor de la misma en 10:

```
var distance:Number = 10;
```

4. Añada el siguiente código ActionScript debajo del código existente en el panel Acciones:

```
this.createTextField("display_txt", 999, 0, 0, 100, 20);
```

5. Para crear un controlador de eventos para el clip de película del coche que compruebe qué tecla de flecha (izquierda, derecha, arriba o abajo) se encuentra presionada, añada el código siguiente en el panel Acciones:

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
};
Key.addListener(keyListener);
```

6. Para comprobar que está presionando la tecla de flecha izquierda y para mover el clip de película del coche en esa dirección, añada el código al cuerpo del controlador de eventos `onEnterFrame`.

   El código debería parecerse al del siguiente ejemplo (el código nuevo está en negrita):

```
var distance:Number = 10;
this.createTextField("display_txt", 999, 0, 0, 100, 20);
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.LEFT)) {
    car_mc._x = Math.max(car_mc._x - distance, 0);
    display_txt.text = "Left";
  }
};
Key.addListener(keyListener);
```

   Si se presiona la tecla de flecha izquierda, la propiedad _x del coche se establece con el valor actual de _x menos la distancia o el valor 0, en función de cuál de éstos sea mayor. Por consiguiente, el valor de la propiedad _x nunca podrá ser menor que 0. Asimismo, la palabra *Left* debe aparecer en el archivo SWF.

7. Utilice un código similar para comprobar si se está presionando la tecla de flecha derecha, arriba o abajo.

   El código completo debería parecerse al del siguiente ejemplo (el código nuevo está en negrita):

```
var distance:Number = 10;
this.createTextField("display_txt", 999, 0, 0, 100, 20);
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.isDown(Key.LEFT)) {
    car_mc._x = Math.max(car_mc._x - distance, 0);
    display_txt.text = "Left";
  } else if (Key.isDown(Key.RIGHT)) {
    car_mc._x = Math.min(car_mc._x + distance, Stage.width -
  car_mc._width);
    display_txt.text = "Right";
  } else if (Key.isDown(Key.UP)) {
    car_mc._y = Math.max(car_mc._y - distance, 0);
    display_txt.text = "Up";
  } else if (Key.isDown(Key.DOWN)) {
    car_mc._y = Math.min(car_mc._y + distance, Stage.height -
  car_mc._height);
    display_txt.text = "Down";
  }
};
Key.addListener(keyListener);
```

8. Seleccione Control > Probar película para probar el archivo.

Para más información sobre los métodos de la clase Key, consulte %{Key}% en *Referencia del lenguaje ActionScript 2.0*.

# Configuración de valores de color

Puede utilizar los métodos de la clase incorporada ColorTransform (flash.geom.ColorTransform) para ajustar el color de un clip de película. La propiedad `rgb` de la clase ColorTransform asigna valores hexadecimales RVA (rojo, verde, azul) al clip de película. En el siguiente ejemplo se utiliza `rgb` para cambiar el color de un objeto en función del botón en el que haga clic el usuario.

### Para establecer el valor de color de un clip de película:

1. Cree un nuevo documento de Flash y guárdelo como **setrgb.fla**.
2. Seleccione la herramienta Rectángulo para dibujar un gran cuadrado en el escenario.
3. Convierta la forma en un símbolo de clip de película y proporcione al símbolo el nombre de instancia **car_mc** en el inspector de propiedades.

4. Cree un símbolo de botón denominado colorChip, sitúe las cuatro instancias del botón en el escenario y asígneles los nombres `red_btn`, `green_btn`, `blue_btn` y `black_btn`.
5. Seleccione el fotograma 1 en la línea de tiempo principal y elija Ventana > Acciones.
6. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
var trans:Transform = new Transform(car_mc);
trans.colorTransform = colorTrans;
```

7. Para hacer que el botón de color azul cambie el color del clip de película `car_mc` a azul, añada el código siguiente al panel Acciones:

```
blue_btn.onRelease = function() {
  colorTrans.rgb = 0x333399; // azul
  trans.colorTransform = colorTrans;
};
```

   El fragmento anterior de código cambia la propiedad `rgb` del objeto de transformación de color y vuelve a aplicar el efecto de transformación de color para el clip de película `car_mc` cada vez que se presione el botón.
8. Repita el paso 7 para los demás botones (`red_btn`, `green_btn` y `black_btn`) para cambiar el color del clip de película al color que corresponda.

   El código debe parecerse ahora al del siguiente ejemplo (el código nuevo está en negrita):

```
import flash.geom.ColorTransform;
import flash.geom.Transform;

var colorTrans:ColorTransform = new ColorTransform();
var trans:Transform = new Transform(car_mc);
trans.colorTransform = colorTrans;

blue_btn.onRelease = function() {
  colorTrans.rgb = 0x333399; // azul
  trans.colorTransform = colorTrans;
};
red_btn.onRelease = function() {
  colorTrans.rgb = 0xFF0000; // rojo
  trans.colorTransform = colorTrans;
};
green_btn.onRelease = function() {
  colorTrans.rgb = 0x006600; // verde
  trans.colorTransform = colorTrans;
};
black_btn.onRelease = function() {
  colorTrans.rgb = 0x000000; // negro
  trans.colorTransform = colorTrans;
};
```

9. Seleccione Control > Probar película para cambiar el color del clip de película.

Para más información sobre los métodos de la clase ColorTransform, consulte %{ColorTransform (flash.geom.ColorTransform)}% en *Referencia del lenguaje ActionScript 2.0*.

## Creación de controles de sonido

Utilice la clase incorporada Sound para controlar los sonidos de un archivo SWF. Para utilizar los métodos de la clase Sound, deberá crear primero un nuevo objeto Sound. A continuación, puede utilizar el método `attachSound()` para insertar un sonido de la biblioteca en un archivo SWF mientras el archivo SWF se está reproduciendo.

El método `setVolume()` de la clase Sound controla el volumen y el método `setPan()` ajusta el balance izquierdo y derecho de un sonido.

Los siguientes procedimientos muestran cómo crear controles de sonido.

### Para asociar un sonido a una línea de tiempo:

1. Seleccione Archivo > Importar para importar un sonido.
2. Seleccione el sonido en la biblioteca, haga clic con el botón derecho del ratón (Windows) o con la tecla Control (Macintosh) y seleccione Vinculación.
3. Seleccione Exportar para ActionScript y Exportar en primer fotograma; a continuación, asocie el sonido al identificador `a_thousand_ways`.
4. Añada un botón al escenario y asígnele el nombre `play_btn`.
5. Añada un botón al escenario y asígnele el nombre `stop_btn`.
6. Seleccione el fotograma 1 en la línea de tiempo principal y elija Ventana > Acciones.

   Añada el código siguiente al panel Acciones:

   ```
   var song_sound:Sound = new Sound();
   song_sound.attachSound("a_thousand_ways");
   play_btn.onRelease = function() {
     song_sound.start();
   };
   stop_btn.onRelease = function() {
     song_sound.stop();
   };
   ```

   Este código primero detiene el clip de película speaker. A continuación, crea un objeto Sound (`song_sound`) nuevo y lo asocia al sonido cuyo identificador de vínculo es `a_thousand_ways`. Los controladores de eventos `onRelease` asociados con los objetos `playButton` y `stopButton` inician y detienen el sonido mediante los métodos `Sound.start()` y `Sound.stop()`. Asimismo, reproducen y detienen el sonido asociado.
7. Seleccione Control > Probar película para oír el sonido.

### Para crear un control deslizable de volumen:

1. Con la herramienta Rectángulo, dibuje un rectángulo pequeño del escenario, de aproximadamente 30 píxeles de alto por 10 píxeles de ancho.
2. Seleccione la herramienta Selección y haga doble clic en el escenario.
3. Presione F8 para abrir el cuadro de diálogo Convertir en símbolo.
4. Seleccione el tipo de botón, introduzca el nombre de símbolo **volume** y haga clic en Aceptar.
5. Con el símbolo de botón seleccionado en el escenario, introduzca el nombre de instancia **handle_btn** en el inspector de propiedades.
6. Seleccione el botón y, a continuación, elija Modificar > Convertir en símbolo.

   Asegúrese de elegir el comportamiento del clip de película. Esta acción crea un clip de película con el botón en el fotograma 1.
7. Seleccione el clip de película e introduzca **volume_mc** como nombre de la instancia en el inspector de propiedades.
8. Seleccione el fotograma 1 en la línea de tiempo principal y elija Ventana > Acciones.
9. Introduzca los códigos siguientes en el panel Acciones:

```
this.createTextField("volume_txt", 10, 30, 30, 200, 20);
volume_mc.top = volume_mc._y;
volume_mc.bottom = volume_mc._y;
volume_mc.left = volume_mc._x;
volume_mc.right = volume_mc._x + 100;
volume_mc._x += 100;

volume_mc.handle_btn.onPress = function() {
  startDrag(this._parent, false, this._parent.left, this._parent.top,
  this._parent.right, this._parent.bottom);
};
volume_mc.handle_btn.onRelease = function() {
  stopDrag();
  var level:Number = Math.ceil(this._parent._x - this._parent.left);
  this._parent._parent.song_sound.setVolume(level);
  this._parent._parent.volume_txt.text = level;
};
volume_mc.handle_btn.onReleaseOutside = slider_mc.handle_btn.onRelease;
```

   Los parámetros de `starDrag()` `left`, `top`, `right` y `bottom` son variables definidas en una acción de clip de película.
10. Seleccione Control > Probar película para utilizar el deslizador de volumen.

### Para crear un control de deslizador de balance:

1. Con la herramienta Rectángulo, dibuje un rectángulo pequeño del escenario, de aproximadamente 30 píxeles de alto por 10 píxeles de ancho.
2. Seleccione la herramienta Selección y haga doble clic en el escenario.
3. Presione F8 para abrir el cuadro de diálogo Convertir en símbolo.
4. Seleccione el tipo de botón, introduzca el nombre de símbolo **balance** y haga clic en Aceptar.
5. Con el símbolo de botón seleccionado en el escenario, introduzca el nombre de instancia **handle_btn** en el inspector de propiedades.
6. Seleccione el botón y, a continuación, elija Modificar > Convertir en símbolo.

   Asegúrese de elegir el comportamiento del clip de película. Esta acción crea un clip de película con el botón en el fotograma 1.
7. Seleccione el clip de película e introduzca **balance_mc** como nombre de la instancia en el inspector de propiedades.
8. Introduzca los códigos siguientes en el panel Acciones:

```
balance_mc.top = balance_mc._y;
balance_mc.bottom = balance_mc._y;
balance_mc.left = balance_mc._x;
balance_mc.right = balance_mc._x + 100;
balance_mc._x += 50;
balance_mc.handle_btn.onPress = function() {
  startDrag(this._parent, false, this._parent.left, this._parent.top,
  this._parent.right, this._parent.bottom);
};
balance_mc.handle_btn.onRelease = function() {
  stopDrag();
  var level:Number = Math.ceil((this._parent._x - this._parent.left -
  50) * 2);
  this._parent._parent.song_sound.setPan(level);
};
balance_mc.handle_btn.onReleaseOutside =
  balance_mc.handle_btn.onRelease;
```

   Los parámetros de `starDrag()` `left`, `top`, `right` y `bottom` son variables definidas en una acción de clip de película.
9. Seleccione Control > Probar película para utilizar el deslizador de balance.

Para más información sobre los métodos de la clase Sound, consulte %{Sound}% en *Referencia del lenguaje ActionScript 2.0*.

# Detección de colisiones

El método `hitTest()` de la clase MovieClip detecta las colisiones dentro de un archivo SWF. Comprueba si el objeto ha colisionado con un clip de película y devuelve un valor booleano (`true` o `false`).

Desea saber si se ha producido una colisión para probar si el usuario ha llegado a un área estática específica en el escenario o determinar cuándo un clip de película ha alcanzado a otro. Con el método `hitTest()`, puede determinar estos resultados.

Puede utilizar los parámetros del método `hitTest()` para especificar las coordenadas *x* e *y* de una zona activa en el escenario o utilizar la ruta de destino de otro clip de película como zona activa. Al especificar *x* e *y*, `hitTest()` devuelve `true` si el punto identificado por (*x*, *y*) no es un punto transparente. Cuando se pasa un destino a `hitTest()`, se comparan los recuadros de delimitación de los dos clips de película. Si se solapan, `hitTest()` devuelve `true`. Si los dos cuadros no tienen ningún punto de intersección, `hitTest()` devuelve `false`.

También puede utilizar el método `hitTest()` para probar la colisión entre dos clips de película.

En el siguiente ejemplo se muestra cómo se detecta la colisión entre un ratón y los clips de película del escenario.

### Para detectar la colisión entre un clip de película y el puntero de ratón:

1. Seleccione el primer fotograma de la Capa 1 en la línea de tiempo.
2. Seleccione Ventana > Acciones para abrir el panel Acciones en el caso de que no esté abierto.
3. Introduzca el código siguiente en el panel Acciones:

```
this.createEmptyMovieClip("box_mc", 10);
with (box_mc) {
  beginFill(0xFF0000, 100);
  moveTo(100, 100);
  lineTo(200, 100);
  lineTo(200, 200);
  lineTo(100, 200);
  lineTo(100, 100);
  endFill();
}

this.createTextField("status_txt", 999, 0, 0, 100, 22);

var mouseListener:Object = new Object();
mouseListener.onMouseMove = function():Void {
  status_txt.text = _level0.hitTest(_xmouse, _ymouse, true);
}
Mouse.addListener(mouseListener);
```

4. Seleccione Control > Probar película y mueva el puntero sobre el clip de película para comprobar la colisión.

   El valor `true` aparece cuando el puntero está sobre un píxel no transparente.

### Para detectar una colisión en dos clips de película:

1. Arrastre dos clips de película al escenario y asígneles los nombres de instancia `car_mc` y `area_mc`.
2. Seleccione el fotograma 1 de la línea de tiempo.
3. Seleccione Ventana > Acciones para abrir el panel Acciones en el caso de que no esté abierto.
4. Introduzca los códigos siguientes en el panel Acciones:

```
this.createTextField("status_txt", 999, 10, 10, 100, 22);
area_mc.onEnterFrame = function() {
  status_txt.text = this.hitTest(car_mc);
};

car_mc.onPress = function() {
  this.startDrag(false);
  updateAfterEvent();
};
car_mc.onRelease = function() {
  this.stopDrag();
};
```

5. Seleccione Control > Probar película y arrastre el clip de película para comprobar la detección de una colisión.

   Siempre que el recuadro de delimitación del coche tenga un punto de intersección con el recuadro de delimitación del área, el estado es `true`.

Para más información, consulte %{hitTest (método MovieClip.hitTest)}% en *Referencia del lenguaje ActionScript 2.0*.

## Creación de una herramienta sencilla de dibujo de líneas

Puede utilizar los métodos de la clase MovieClip para dibujar líneas y rellenos en el escenario mientras se reproduce el archivo SWF. Esto permite crear herramientas de dibujo para los usuarios y dibujar formas en el archivo SWF como respuesta a los eventos. Los métodos de dibujo son `beginFill()`, `beginGradientFill()`, `clear()`, `curveTo()`, `endFill()`, `lineTo()`, `lineStyle()` y `moveTo()`.

Puede aplicar estos métodos a cualquier instancia de clip de película (por ejemplo, `myClip.lineTo()`) o a un nivel (`_level0.curveTo()`).

Los métodos `lineTo()` y `curveTo()` permiten dibujar líneas y curvas, respectivamente. Utilice el método `lineStyle()` para especificar el color de línea, el grosor y el parámetro alfa para una línea o curva. El método de dibujo `moveTo()` establece la posición del dibujo actual en las coordenadas de escenario *x* e *y* que especifique.

Los métodos `beginFill()` y `beginGradientFill()` rellenan un trazado cerrado con un relleno degradado o sólido respectivamente, y `endFill()` aplica el relleno especificado en la última llamada a `beginFill()` o a `beginGradientFill()`. El método `clear()` elimina lo que se haya dibujado en el objeto del clip de película especificado.

### Para crear una herramienta sencilla de trazo de líneas:

1. En un documento nuevo, cree un nuevo botón en el escenario y escriba `clear_btn` como nombre de instancia en el inspector de propiedades.
2. Seleccione el fotograma 1 de la línea de tiempo.
3. Seleccione Ventana > Acciones para abrir el panel Acciones en el caso de que no esté abierto.
4. En el panel Acciones, especifique el siguiente código:

```
this.createEmptyMovieClip("canvas_mc", 999);
var isDrawing:Boolean = false;
//
clear_btn.onRelease = function() {
  canvas_mc.clear();
};
//
var mouseListener:Object = new Object();
mouseListener.onMouseDown = function() {
  canvas_mc.lineStyle(5, 0xFF0000, 100);
  canvas_mc.moveTo(_xmouse, _ymouse);
  isDrawing = true;
};
mouseListener.onMouseMove = function() {
  if (isDrawing) {
    canvas_mc.lineTo(_xmouse, _ymouse);
    updateAfterEvent();
  }
};
mouseListener.onMouseUp = function() {
  isDrawing = false;
};
Mouse.addListener(mouseListener);
```

5. Seleccione Control > Probar película para probar el documento.
6. Haga clic y arrastre el puntero para dibujar una línea en el escenario.
7. Haga clic en el botón para borrar lo que ha dibujado.

# Creación de vinculaciones de datos durante la ejecución mediante ActionScript

Si utiliza componentes para crear aplicaciones, suele ser necesario añadir vinculaciones entre dichos componentes para que pueda interactuar con datos o que un componente interactúe con otro. La interacción entre los componentes es necesaria para crear formularios o interfaces con los que los usuarios puedan interactuar. Puede utilizar la ficha Vinculaciones del inspector de componentes para añadir vinculaciones entre componentes del escenario. Puede utilizar la ficha Vinculaciones del inspector de componentes para vincular datos entre componentes en el escenario.

Para más información sobre el uso de la ficha Vinculaciones, consulte "Trabajo con vinculaciones en la ficha Vinculaciones (sólo Flash Professional)" en la página 444 en *Utilización de Flash*. También puede encontrar información adicional en los siguientes artículos en línea: Building a Tip of the day Application (Part 2), Data Binding in Macromedia Flash MX Professional 2004 y Building a Google Search Application with Macromedia Flash MX Professional.

Puede utilizar ActionScript en lugar de la ficha Vinculaciones para crear vinculaciones entre componentes. La adición de código suele ser más rápida y eficiente que el uso del entorno de edición. La utilización de código ActionScript para crear vinculaciones es necesaria cuando se utiliza código para añadir componentes a una aplicación. Puede optar por utilizar el método `createClassObject()` para añadir componentes en el escenario de forma dinámica; no obstante, no se puede utilizar la ficha Vinculaciones para crear una vinculación porque los componentes no existen hasta su ejecución. La utilización de código ActionScript para añadir vinculaciones de datos se conoce como *vinculación de datos en tiempo de ejecución*.

Para más información, consulte los siguientes temas:

- Creación de vinculaciones entre componentes de interfaz de usuario mediante código ActionScript
- "Utilización de componentes, vinculaciones y formateadores personalizados" en la página 619
- "Adición y vinculación de componentes en el escenario" en la página 622

## Creación de vinculaciones entre componentes de interfaz de usuario mediante código ActionScript

No es difícil vincular datos entre dos componentes durante la ejecución. Puede realizar esta operación en Flash Basic 8 o Flash Professional 8. No olvide incluir el componente DataBindingClasses en el documento para que funcione, ya que ese componente contiene las clases que necesita para trabajar.

### Para crear una vinculación entre dos componentes TextInput mediante código ActionScript:

1. Cree un nuevo documento de Flash denominado **panel_as.fla**.
2. Arrastre dos copias del componente TextInput al escenario.
3. Asigne a los componentes los siguientes nombres de instancias: **in_ti** y **out_ti**.
4. Seleccione Ventana > Bibliotecas comunes > Clases y abra la nueva biblioteca común llamada Classes.fla.
5. Arrastre una copia del componente DataBindingClasses al panel Biblioteca o arrastre el componente al escenario y elimínelo.

   Podrá cerrar la biblioteca común cuando termine. Tras eliminar el componente DataBindingClasses del escenario, Flash conserva una copia en la biblioteca.

| SUGERENCIA | Si olvida eliminar el componente DataBindingClasses del escenario, el icono del componente estará visible durante la ejecución. |
|---|---|

| NOTA | Cuando creó una vinculación empleando el inspector de componentes en el ejemplo anterior, Flash añadió automáticamente el componente DataBindingClasses al archivo FLA. Al utilizar código ActionScript para crear vinculaciones de datos, deberá copiar la clase en la biblioteca, como se muestra en el siguiente paso. |
|---|---|

6. Inserte una nueva capa y asígnele el nombre **acciones**.
7. Añada el siguiente código ActionScript al fotograma 1 de la capa acciones:

```
var src:mx.data.binding.EndPoint = new mx.data.binding.EndPoint();
src.component = in_ti;
src.property = "text";
src.event = "focusOut";
var dest:mx.data.binding.EndPoint = new mx.data.binding.EndPoint();
dest.component = out_ti;
dest.property = "text";
new mx.data.binding.Binding(src, dest);
```

Si prefiere la versión abreviada, puede importar las clases de vinculación y utilizar el siguiente código:

```
import mx.data.binding.*;
var src:EndPoint = new EndPoint();
src.component = in_ti;
src.property = "text";
src.event = "focusOut";
var dest:EndPoint = new EndPoint();
dest.component = out_ti;
dest.property = "text";
new Binding(src, dest);
```

Este código ActionScript crea dos puntos finales de la vinculación de datos, uno por cada componente vinculado. El primer punto final que cree definirá cuál es el componente de origen de la vinculación (`in_ti`), qué propiedad debe controlarse (`text`) y qué evento accionará la vinculación (`focusOut`). El segundo punto final creado sólo incluye el componente y la propiedad (`out_ti` y `text`, respectivamente). Finalmente, la vinculación entre dos puntos finales se crea al llamar al constructor de la clase Binding (`new Binding(src, dest)`).

No es necesario utilizar nombres de clases completos (como, por ejemplo, `mx.data.binding.EndPoint`) en el código ActionScript, como se observa en el primer fragmento de código. Si utiliza la sentencia `import` al comienzo del código, evitará tener que utilizar los nombres completos. Al importar todas las clases del paquete `mx.data.binding` mediante el comodín `*` (el paquete incluye las clases EndPoint y Binding), conseguirá que el código sea más breve y que se haga referencia directamente a las clases EndPoint y Binding. Para más información sobre sentencias import, consulte la entrada `import` de *Referencia del lenguaje ActionScript 2.0*.

8. Seleccione Control > Probar película para comprobar este código en el entorno de prueba. Introduzca texto en el campo de introducción de texto `in_ti`.

   Una vez que la instancia `in_ti` deje de estar seleccionada (haga clic en el escenario, presione Tab o haga clic en el segundo campo), Flash copiará el texto introducido en `in_ti` en el campo de texto `out_ti`.

9. Seleccione Archivo > Guardar para guardar los cambios.

Puede que el código resulte mucho más complicado si desea modificar el texto del campo de introducción de texto `out_ti` del ejercicio anterior. Si utiliza el inspector de componentes para configurar vinculaciones, se creará una conexión bidireccional de manera predeterminada. Esto significa que, si cambia cualquiera de los campos de texto del escenario, el otro campo de texto también cambiará. Cuando cree vinculaciones mediante código ActionScript, la aplicación funcionará a la inversa. Las vinculaciones de datos de tiempo de ejecución son unidireccionales de manera predeterminada, a no ser que especifique lo contrario, como se muestra en el siguiente ejemplo.

Para utilizar ActionScript para crear una vinculación bidireccional, deberá realizar un par de modificaciones en los fragmentos de código del ejemplo anterior. Este ejemplo utiliza el segundo fragmento abreviado de código ActionScript del paso 7.

### Para crear una vinculación bidireccional:

1. Abra **panel_as.fla** del ejemplo anterior.
2. Modifique el código ActionScript (consulte el código **en negrita**) para que coincida con el siguiente código ActionScript:

```
import mx.data.binding.*;
var src:EndPoint = new EndPoint();
src.component = in_ti;
src.property = "text";
src.event = "focusOut";
var dest:EndPoint = new EndPoint();
dest.component = out_ti;
dest.property = "text";
dest.event = "focusOut";
new Binding(src, dest, null, true);
```

   Los dos cambios que debe realizar en el código ActionScript hacen lo siguiente:

   - Definir una propiedad de evento para la instancia EndPoint de destino.
   - Definir dos parámetros adicionales para el constructor de Binding.

   Utilice el primer parámetro para opciones de formato avanzadas; el valor puede establecerse como `null` o `undefined`. El segundo parámetro define si la vinculación es bidireccional (`true`) o unidireccional (`false`).

   Puede que se pregunte por la procedencia del evento `focusOut`. Ese es el punto en el que el código ActionScript se vuelve complicado. Puede investigar la clase TextInput y utilizar algunos de los métodos enumerados (como `change()` o `enter()`), pero no encontrará en ella el evento `focusOut`. La clase TextInput hereda de las clases UIObject y UIComponent. Si observa la clase UIComponent, que añade capacidad de selección para componentes, verá cuatro eventos adicionales: `focusIn`, `focusOut`, `keyDown` y `keyUp`. Puede utilizar estos eventos con el componente TextInput.

3. (Opcional) Si desea que el ejemplo anterior actualice el valor del campo de introducción de datos `out_ti`, puede cambiar el evento de `focusOut` a `change`.
4. Seleccione Control > Probar película para probar el documento.

   Flash cambia el segundo valor del campo de introducción de datos `in_ti` y actualiza el valor para `out_ti`. Ha creado correctamente una conexión bidireccional.

Puede utilizar las clases de vinculación con la mayoría de componentes de interfaz de la versión 2 de la arquitectura de componentes de Macromedia, no sólo con el componente TextInput. En el siguiente ejemplo se muestra cómo utilizar ActionScript para vincular instancias de CheckBox y componentes Label durante la ejecución.

### Para utilizar clases de vinculación con el componente CheckBox:

1. Cree un nuevo documento de Flash.
2. Seleccione Archivo > Guardar como y asigne al nuevo archivo el nombre **checkbox_as.fla**.
3. Seleccione Window > Bibliotecas comunes > Clases.
4. Arrastre una copia de la clase DataBindingClasses a la biblioteca del documento.
5. Arrastre una copia del componente CheckBox al escenario y asígnele el nombre de instancia **my_ch**.
6. Arrastre una copia del componente Label al escenario y asígnele el nombre de instancia **my_lbl**.
7. Cree una nueva capa y asígnele el nombre **actions**.
8. Añada el siguiente código ActionScript al fotograma 1 de la capa actions:

```
var srcEndPoint:Object = {component:my_ch, property:"selected",
  event:"click"};
var destEndPoint:Object = {component:my_lbl, property:"text"};
new mx.data.binding.Binding(srcEndPoint, destEndPoint);
```

   Deberá utilizar objetos para definir los puntos finales en lugar de crear nuevas instancias de la clase EndPoint, como se muestra en los ejercicios anteriores de esta sección. El fragmento de código de este paso crea dos objetos que actúan como puntos finales de la vinculación. La vinculación se crea al llamar al constructor de la clase Binding. Para reducir aún más la cantidad de código (pero también la legibilidad de éste), defina objetos en línea como se muestra en el siguiente fragmento:

```
new mx.data.binding.Binding({component:my_ch, property:"selected",
  event:"click"}, {component:my_lbl, property:"text"});
```

   Este código ActionScript reduce la legibilidad del código, pero también reduce la cantidad de código que es preciso introducir. Si comparte los archivos FLA (o de ActionScript), puede que desee utilizar el primer fragmento de código ActionScript, ya que resulta más fácil de leer.

# Utilización de componentes, vinculaciones y formateadores personalizados

Los formateadores personalizados le ayudan a formatear datos complejos de una forma concreta. También puede utilizar el formato personalizado para contribuir a la visualización de imágenes, texto con formato HTML u otros componentes dentro de un componente como DataGrid. En el siguiente ejemplo se muestra lo útiles que pueden ser los formateadores personalizados.

### Para utilizar formateadores personalizados en un documento:

1. Cree un nuevo archivo FLA y añada la clase DataBindingClasses a la biblioteca (Ventana > Bibliotecas comunes > Clases).
2. Arrastre una copia del componente DateChooser al escenario y asígnele el nombre de instancia **my_dc**.
3. Arrastre una copia del componente Label al escenario y asígnele el nombre de instancia **my_lbl**.
4. Inserte una nueva capa y asígnele el nombre **acciones**.
5. Añada el siguiente código ActionScript al fotograma 1 de la capa acciones:

```
import mx.data.binding.*;
var src:EndPoint = new EndPoint();
src.component = my_dc;
src.property = "selectedDate";
src.event = "change";
var dest:EndPoint = new EndPoint();
dest.component = my_lbl;
dest.property = "text";
new Binding(src, dest);
```

   Este código crea una vinculación entre la propiedad `selectedDate` de DateChooser y la propiedad `text` del componente Label del escenario. Cada vez que haga clic en una nueva fecha del calendario, la fecha seleccionada aparecerá en el componente Label.
6. Guarde el documento de Flash como **customformat.fla** en el lugar que prefiera del disco duro.

   (Lo reciclará en el siguiente ejercicio.)
7. Seleccione Control > Probar película para probar el documento.

   Intente cambiar las fechas del componente Calendar y observará que la fecha actualmente seleccionada aparece en el componente Label. El componente Label no es lo suficientemente ancho como para mostrar la fecha completa, por lo que Flash corta el texto.

8. Cierre el archivo SWF de prueba y regrese al entorno de edición.

   Cambie el tamaño del componente Label del escenario o seleccione el componente Label y establezca la propiedad `autoSize` con el valor `left` en la ficha Parámetros del inspector de propiedades.

9. Seleccione Control > Probar película para probar el documento de nuevo.

   Ahora el campo de texto muestra la fecha completa, aunque resulta un poco rara y carece de formato. Dependiendo de la zona horaria y la fecha seleccionadas, la fecha puede mostrarse así: `Thu Nov 4 00:00:00 GMT-0800 2004`

   Aunque la vinculación funciona correctamente y muestra la propiedad `selectedDate`, estas fechas no son muy fáciles de usar. Los desplazamientos de zona horaria con respecto al meridiano GMT no son fáciles de entender y, además, es posible que no desee mostrar horas, minutos y segundos. Lo que necesita es alguna manera de dar formato a la fecha para que resulte más fácil de leer y menos mecánica. Los formateadores personalizados son especialmente útiles para aplicar formato al texto.

## Aplicación de formato a datos empleando la clase CustomFormatter

La clase CustomFormatter define dos métodos, `format()` y `unformat()`, que proporcionan la capacidad de transformar valores de datos de un tipo de datos determinado en una cadena y viceversa. De forma predeterminada, estos métodos no realizan ninguna función. Debe implementarlos en una subclase de `mx.data.binding.CustomFormatter`. La clase CustomFormatter le permite convertir tipos de datos en cadenas y a la inversa. En este caso, desea convertir la propiedad `selectedDate` del componente DateChooser en una cadena bien formateada cuando el valor se copie al componente Label.

En el siguiente ejemplo se muestra cómo crear un formateador personalizado para mostrar la fecha como `NOV 4, 2004` en lugar de mostrar una cadena de fecha predeterminada.

NOTA

Deberá realizar el ejercicio a partir de "Utilización de componentes, vinculaciones y formateadores personalizados" en la página 619 antes de empezar éste.

### Para aplicar formato a datos empleando la clase CustomFormatter:

1. Seleccione Archivo > Nuevo y elija Archivo ActionScript para crear un nuevo archivo AS.
2. Seleccione Archivo > Guardar como y guarde el nuevo archivo como **DateFormat.as**.
3. Introduzca el siguiente código en la ventana Script:

```
class DateFormat extends mx.data.binding.CustomFormatter {
  function format(rawValue:Date):String {
    var returnValue:String;
    var monthName_array:Array =
  ["JAN","FEB","MAR","APR","MAY","JUN","JUL","AUG","SEP","OCT","NOV","D
  EC"];
    returnValue = monthName_array[rawValue.getMonth()]+"
  "+rawValue.getDate()+", "+rawValue.getFullYear();
    return returnValue;
  }
}
```

En la primera sección del código, se define la nueva clase denominada DateFormat, que amplía la clase CustomFormatter del paquete `mx.data.binding`. Recuerde que Flash compila las clases de vinculación del archivo de componente DataBindingClasses, por lo que no puede verlas directamente ni localizarlas en la carpeta Classes del directorio de instalación de Flash.

El único método que deberá utilizar es el método `format()`, que convierte la instancia de fecha a un formato de cadena personalizado. El siguiente paso consiste en crear una matriz de nombres de meses para que el resultado final se parezca más a `NOV 4, 2004` que al formato de fecha predeterminado. Recuerde que las matrices están basadas en cero en Flash, por lo que, si el valor de `rawValue.getMonth()` devuelve `1`, representa a February (febrero) y no a January (enero, ya que January es el mes `0`). El código restante crea la cadena con formato personalizado concatenando valores y devolviendo la cadena `returnValue`.

Podría surgir un problema al trabajar con clases con un clip compilado, lo que puede ver en el fragmento anterior. Dado que está ampliando una clase ubicada en la clase DataBindingClasses y ésta no está disponible de inmediato para Flash, verá el siguiente error al revisar la sintaxis de la clase anterior:

```
**Error** <path to DateFormat class>\DateFormat.as: Line 1: The class
  'mx.data.binding.CustomFormatter' could not be loaded.
     class DateFormat extends mx.data.binding.CustomFormatter {

Total ActionScript Errors: 1  Reported Errors: 1
```

Probablemente el código no presente ningún error. Este problema tiene lugar cuando Flash no encuentra la clase y, a consecuencia de ello, se produce un error durante la revisión sintáctica.

4. Guarde el archivo DateFormat.as.
5. Abra el archivo customformat.fla del ejercicio realizado en "Utilización de componentes, vinculaciones y formateadores personalizados". Asegúrese de que guarda o copia DateFormat.as en el mismo directorio que este archivo.
6. En customformat.fla, modifique el código ActionScript del fotograma 1 de la capa acciones para que coincida con el siguiente código:

```
import mx.data.binding.*;
var src:EndPoint = new EndPoint();
src.component = my_dc;
src.property = "selectedDate";
src.event = "change";
var dest:EndPoint = new EndPoint();
dest.component = my_lbl;
dest.property = "text";
new Binding(src, dest, {cls:mx.data.formatters.Custom,
  settings:{classname:"DateFormat", classname_class:DateFormat}});
```

   En esta ocasión, se define un objeto `customFormatter` que indica a Flash que está utilizando la clase DateFormat recién creada para dar formato al punto final de la vinculación.
7. Guarde los cambios en el documento y seleccione Control > Probar película para comprobar el código.

## Adición y vinculación de componentes en el escenario

Una de las mayores ventajas que aporta el uso de clases de vinculación con código ActionScript es que le permite crear vinculaciones entre componentes que Flash ha añadido al escenario durante la ejecución. Supongamos que crea su propia clase personalizada para añadir al escenario los campos de texto adecuados durante la ejecución y luego validar los datos necesarios y añadir las vinculaciones pertinentes. Siempre y cuando tenga componentes en la biblioteca, podrá añadirlos dinámicamente y utilizar un par de líneas de código adicional para crear vinculaciones.

### Para añadir y luego vincular componentes del escenario empleando código ActionScript:

1. Cree un nuevo documento de Flash.
2. Arrastre un componente ComboBox y Label a la biblioteca del documento.
3. Inserte una nueva capa y asígnele el nombre **acciones**.

4. Añada el siguiente código al fotograma 1 de la capa acciones:

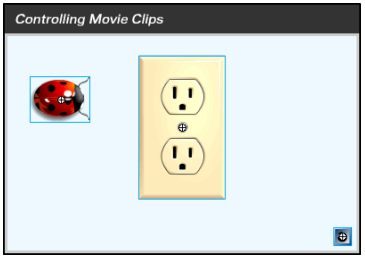
```
import mx.data.binding.*;
this.createClassObject(mx.controls.ComboBox, "my_cb", 1, {_x:10,
  _y:10});
this.createClassObject(mx.controls.Label, "my_lbl", 2, {_x:10, _y:40});
my_cb.addItem("JAN", 0);
my_cb.addItem("FEB", 1);
my_cb.addItem("MAR", 2);
my_cb.addItem("APR", 3);
my_cb.addItem("MAY", 4);
my_cb.addItem("JUN", 5);
var src:EndPoint = new EndPoint();
src.component = my_cb;
src.property = "value";
src.event = "change";
var dest:EndPoint = new EndPoint();
dest.component = my_lbl;
dest.property = "text";
new Binding(src, dest);
```

La primera línea del código ActionScript importa las clases del paquete `mx.data.binding` para que no tenga que utilizar las rutas completas en el código. Las dos líneas siguientes del código ActionScript asocian los componentes de la biblioteca del documento al escenario. Seguidamente, se sitúan los componentes en el escenario.

Finalmente, deberá añadir datos a la instancia de ComboBox y crear la vinculación entre el componente ComboBox `my_cb` y el componente Label `my_lbl` del escenario.

# Análisis de un script de ejemplo

En el archivo SWF de muestra zapper.swf (que puede visualizarse en el apartado Utilización de Flash de la Ayuda), cuando un usuario arrastra la mariquita hasta la toma eléctrica, la mariquita cae y la toma empieza a vibrar. La línea de tiempo principal sólo tiene un fotograma y contiene tres objetos: la mariquita, la toma eléctrica y un botón de restablecimiento. Cada objeto es una instancia de clip de película.



El siguiente script está asociado al fotograma 1 de la línea de tiempo principal:

```
var initx:Number = bug_mc._x;
var inity:Number = bug_mc._y;
var zapped:Boolean = false;

reset_btn.onRelease = function() {
  zapped = false;
  bug_mc._x = initx;
  bug_mc._y = inity;
  bug_mc._alpha = 100;
  bug_mc._rotation = 0;
};

bug_mc.onPress = function() {
  this.startDrag();
};
bug_mc.onRelease = function() {
  this.stopDrag();
};
bug_mc.onEnterFrame = function() {
  if (this.hitTest(this._parent.zapper_mc)) {
    this.stopDrag();
    zapped = true;
    bug_mc._alpha = 75;
    bug_mc._rotation = 20;
    this._parent.zapper_mc.play();
  }
  if (zapped) {
    bug_mc._y += 25;
  }
};
```

El nombre de instancia de la mariquita es `bug_mc` y el de la toma es `zapper_mc`. En el script se hace referencia a la mariquita como `this` porque el script está asociado a la mariquita y la palabra reservada `this` hace referencia al objeto que la contiene.

Hay controladores de eventos con varios eventos diferentes: `onRelease()`, `onPress` y `onEnterFrame()`. Los controladores de eventos se definen en el fotograma 1 después de cargarse el archivo SWF. Las acciones del controlador de eventos `onEnterFrame()` se ejecutan cada vez que la cabeza lectora accede a un fotograma. Aunque se trate de archivos SWF de un solo fotograma, la cabeza lectora accederá al fotograma reiteradamente y el script se ejecutará de la misma forma.

Se definen dos variables, `initx` e `inity`, para almacenar las posiciones *x* e *y* iniciales de la instancia de clip de película `bug_mc`. Se define una función y se asigna al controlador de eventos `onRelease` de la instancia `reset_btn`. Se llama a esta función cada vez que se presiona y suelta el botón del ratón en el botón `reset_btn`. La función coloca la mariquita de nuevo en su posición inicial en el escenario, restablece sus valores de rotación y alfa, y restablece la variable `zapped` en `false`.

Una sentencia `if` condicional utiliza el método `hitTest()` para comprobar si la instancia del insecto está tocando la instancia de la toma eléctrica (`this._parent.zapper_mc`). Los dos resultados posibles de la comprobación son `true` o `false`:

- Si el método `hitTest()` devuelve el valor `true`, Flash llama al método `stopDrag()`, establece la variable `zapper_mc` con el valor `true`, cambia el valor de las propiedades alpha y rotation e indica a la instancia `zapped` que se reproduzca.
- Si el método `hitTest()` devuelve `false`, no se ejecuta ningún código especificado entre (`{}`) que aparezca inmediatamente después de la sentencia `if`.

Las acciones de la sentencia `onPress()` se ejecutan al presionar el botón del ratón sobre la instancia `bug_mc`. Las acciones de la sentencia `onRelease()` se ejecutan al soltar el botón del ratón sobre la instancia `bug_mc`.

La acción `startDrag()` le permite arrastrar la mariquita. Puesto que el script está asociado a la instancia `bug_mc`, la palabra clave `this` indica que la instancia `bug` es la que puede arrastrar:

```
bug_mc.onPress = function() {
  this.startDrag();
};
```

La acción `stopDrag()` detiene la acción de arrastre:

```
bug_mc.onRelease = function() {
  this.stopDrag();
};
```

CAPÍTULO 15

# Utilización de imágenes, sonido y vídeo

15

Si importa una imagen o un sonido mientras edita un documento en Macromedia Flash Basic 8 o en Macromedia Flash Professional 8, la imagen y el sonido se empaquetan y se almacenan en el archivo SWF al publicarlo. Además de importar elementos multimedia durante el proceso de edición, puede cargar elementos multimedia externos, incluidos otros archivos SWF, en tiempo de ejecución. Puede que desee excluir archivos multimedia de un documento de Flash por varias razones.

**Reducir el tamaño de archivo** Si excluye grandes archivos multimedia del documento de Flash y los carga en tiempo de ejecución, puede reducir el tiempo de descarga inicial de las aplicaciones y presentaciones, especialmente si la conexión a Internet es lenta.

**Dividir en módulos grandes presentaciones** Puede dividir una presentación o aplicación de gran tamaño en archivos SWF independientes y cargarlos como desee en tiempo de ejecución. Este proceso reduce el tiempo de descarga inicial y facilita el mantenimiento y la actualización de la presentación.

**Separar el contenido de la presentación** Éste es un tema común en el desarrollo de aplicaciones, especialmente en las aplicaciones basadas en datos. Por ejemplo, una aplicación de cesta de la compra puede mostrar una imagen de cada producto. Si carga cada imagen en tiempo de ejecución, puede actualizar fácilmente la imagen de un producto sin modificar el archivo FLA original.

**Beneficiarse de las funciones que son sólo de tiempo de ejecución** Algunas funciones, como por ejemplo la reproducción de Flash Video (FLV) y MP3 cargados dinámicamente, sólo están disponibles en tiempo de ejecución mediante ActionScript.

En esta sección se describe la forma de trabajar con archivos de imágenes, sonido y vídeo FLV en las aplicaciones de Flash. Para más información, consulte los siguientes temas:

# Carga y trabajo con archivos multimedia externos

Puede cargar varios tipos de archivos multimedia en una aplicación de Flash en tiempo de ejecución. archivos SWF, MP3, JPEG, GIF, PNG y FLV. No obstante, no todas las versiones de Flash Player admiten todos los tipos de archivos multimedia. Para más información sobre los tipos de archivos de imagen que se admiten en Macromedia Flash Player 8, consulte "Carga de archivos de imagen y SWF externos" en la página 629. Para obtener información sobre la compatibilidad con vídeo FLV en Flash Player, consulte "Utilización de vídeo FLV" en la página 640.

Macromedia Flash Player puede cargar archivos multimedia externos desde cualquier dirección HTTP o FTP, desde un disco local utilizando una ruta relativa o mediante el protocolo `file://`.

Para cargar archivos SWF y de imagen externos, puede utilizar la función `loadMovie()` o `loadMovieNum()`, el método `MovieClip.loadMovie()` o el método `MovieClipLoader.loadClip()`. Los métodos de clases ofrecen normalmente más funciones y flexibilidad que las funciones globales, por lo que resultan adecuadas para aplicaciones complejas. Al cargar un archivo SWF o de imagen, debe especificarse un clip de película o nivel de archivo SWF como destino de dichos archivos. Para más información sobre cómo cargar archivos SWF y de imagen, consulte "Carga de archivos de imagen y SWF externos" en la página 629.

Para reproducir un archivo MP3 externo, utilice el método `loadSound()` de la clase Sound. Este método permite especificar si el archivo MP3 se debe descargar progresivamente o terminar de descargarse por completo antes de empezar a reproducirse. También puede leer la información ID3 incorporada en los archivos MP3, si están disponibles. Para más información, consulte "Lectura de etiquetas ID3 en archivos MP3" en la página 638.

Flash Video es el formato de vídeo nativo que utiliza Flash Player. Los archivos FLV se pueden reproducir a través de HTTP o desde el sistema de archivos local. La reproducción de archivos FLV externos ofrece varias ventajas frente a la incorporación de vídeo en un documento de Flash como, por ejemplo, mejor rendimiento y administración de la memoria, así como velocidades de fotogramas de vídeo y Flash independientes. Para más información, consulte "Reproducción dinámica de archivos FLV externos" en la página 643.

También puede precargar archivos multimedia externos o hacer un seguimiento del progreso de la descarga. Flash Player 7 presentó la clase MovieClipLoader, que se puede utilizar para hacer un seguimiento del progreso de la descarga de los archivos SWF o de imagen. Para precargar archivos MP3 y FLV, puede usar el método `getBytesLoaded()` de la clase Sound y la propiedad `bytesLoaded` de la clase NetStream. Para más información, consulte "Precarga de archivos FLV" en la página 647.

Puede encontrar muestras de aplicaciones de galerías de fotos en el disco duro. Estos archivos proporcionan ejemplos de cómo utilizar ActionScript para controlar dinámicamente clips de película mientras se cargan archivos de imagen en un archivo SWF. Los archivos de origen de muestra gallery_tree.fla y gallery_tween.fla se pueden encontrar en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Galleries.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Galleries.

# Carga de archivos de imagen y SWF externos

Para cargar un archivo de imagen o SWF, utilice las funciones globales `loadMovie()` o `loadMovieNum()`, el método `loadMovie()` de la clase MovieClip o el método `loadClip()` de la clase MovieClipLoader. Para más información sobre el método `loadClip()`, consulte `MovieClipLoader.loadClip()` en *Referencia del lenguaje ActionScript 2.0*.

Para archivos de imagen, Flash Player 8 admite el tipo de archivo JPEG (progresivo y no progresivo), imágenes GIF (transparentes y no transparentes, aunque sólo se carga el primer fotograma de un archivo GIF con animación) y archivos PNG (transparentes y no transparentes).

Para cargar un archivo SWF o de imagen en un nivel de Flash Player, utilice la función `loadMovieNum()`. Para cargar un archivo SWF o de imagen en un clip de película de destino, utilice la función o el método `loadMovie()`. En cualquiera de esos casos, el contenido cargado reemplaza al contenido del nivel o clip de película de destino especificado.

Cuando se carga un archivo SWF o de imagen en un clip de película de destino, la esquina superior izquierda del archivo SWF o de imagen se sitúa en el punto de registro del clip de película. Dado que este punto suele ser el centro del clip de película, el contenido cargado no aparecerá en el centro. Además, cuando se carga un archivo SWF o de imagen en una línea de tiempo raíz, la esquina superior izquierda de la imagen se sitúa en la esquina superior izquierda del escenario. El contenido cargado hereda la rotación y la escala del clip de película, pero el contenido original del clip de película se elimina.

También puede enviar variables de código de ActionScript con una llamada `loadMovie()` o `loadMovieNum()`. Esto es útil, por ejemplo, si la URL que especifica en la llamada de método es un script del servidor mediante el cual se devuelve un archivo SWF o de imagen según los datos que se envían desde la aplicación Flash.

Cuando utilice la función global `loadMovie()` o `loadMovieNum()`, especifique el nivel o clip de destino como parámetro. El siguiente ejemplo de código carga el archivo contents.swf de la aplicación Flash en una instancia de clip de película denominada `image_mc`:

```
loadMovie("contents.swf", image_mc);
```

Puede utilizar `MovieClip.loadMovie()` para conseguir el mismo resultado:

```
image_mc.loadMovie("contents.swf");
```

El ejemplo siguiente carga la imagen JPEG image1.jpg en la instancia del clip de película `image_mc`:

```
image_mc.loadMovie("http://www.helpexamples.com/flash/images/image1.jpg");
```

Para más información sobre cómo cargar archivos SWF y de imagen externos, consulte "Carga de archivos SWF y la línea de tiempo raíz" en la página 633.

Para precargar archivos SWF y JPEG en instancias de clip de película, se puede utilizar la clase MovieClipLoader. Esta clase proporciona un mecanismo detector de eventos para notificar sobre el estado de las descargas de archivos en clips de película. Para utilizar un objeto MovieClipLoader para precargar archivos SWF y JPEG, debe realizar lo siguiente:

**Cree un nuevo objeto MovieClipLoader** Puede usar un solo objeto MovieClipLoader para realizar un seguimiento del progreso de descarga de varios archivos o para crear un objeto por separado para el progreso de cada archivo. Cree un nuevo clip de película, cargue el contenido en él y luego cree el objeto MovieClipLoader como se muestre en el código siguiente:

```
this.createEmptyMovieClip("img_mc", 999);
var my_mcl:MovieClipLoader = new MovieClipLoader();
```

**Cree un objeto detector y cree controladores de eventos** El objeto detector puede ser un objeto de ActionScript cualquiera como, por ejemplo, un objeto genérico Object, un clip de película o un componente personalizado.

El ejemplo siguiente crea un objeto detector genérico denominado `loadListener` y define para sí las funciones `onLoadError`, `onLoadStart`, `onLoadProgress` y `onLoadComplete`.

```
// Cree un objeto detector:
var mclListener:Object = new Object();
mclListener.onLoadError = function(target_mc:MovieClip, errorCode:String,
  status:Number) {
  trace("Error loading image: " + errorCode + " [" + status + "]");
};
mclListener.onLoadStart = function(target_mc:MovieClip):Void {
  trace("onLoadStart: " + target_mc);
};
mclListener.onLoadProgress = function(target_mc:MovieClip,
  numBytesLoaded:Number, numBytesTotal:Number):Void {
  var numPercentLoaded:Number = numBytesLoaded / numBytesTotal * 100;
  trace("onLoadProgress: " + target_mc + " is " + numPercentLoaded + "%
  loaded");
};
mclListener.onLoadComplete = function(target_mc:MovieClip,
  status:Number):Void {
  trace("onLoadComplete: " + target_mc);
};
```

**NOTA**

Flash Player 8 permite comprobar el estado HTTP de una descarga de MovieClipLoader en los detectores de eventos onLoadComplete y onLoadError. Esta capacidad permite comprobar los motivos por los cuales no se cargó el archivo: si se trató de un error del servidor, no se pudo encontrar el archivo, etc.

**Registre el objeto detector con el objeto MovieClipLoader** Para que el objeto detector reciba los eventos de carga, debe registrarlo con el objeto MovieClipLoader, como se muestra en el siguiente código:

```
my_mcl.addListener(mclListener);
```

**Inicie la carga del archivo (imagen o SWF) en un clip de destino** Para iniciar la descarga de un archivo de imagen o SWF, use el método `MovieClipLoader.loadClip()`, como se muestra en el siguiente código:

```
my_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img_mc);
```

**NOTA**

Sólo puede usar los métodos MovieClipLoader para realizar un seguimiento del progreso de descarga de los archivos cargados con el método `MovieClipLoader.loadClip()`. No puede usar la función `loadMovie()` ni el método `MovieClip.loadMovie()`.

En el ejemplo siguiente se utiliza el método `setProgress()` del componente ProgressBar para mostrar el progreso de descarga de un archivo SWF. (Consulte "ProgressBar.setProgress()" en *Referencia del lenguaje de componentes*.)

### Para visualizar el progreso de la descarga mediante el componente ProgressBar:

1. Cree un nuevo documento de Flash y guárdelo como **progress.fla**.
2. Abra el panel Componentes (Ventana > Componentes).
3. Arrastre un componente ProgressBar desde el panel Componentes al escenario.
4. En el inspector de propiedades (Ventana > Propiedades > Propiedades), asigne al componente ProgressBar el nombre `my_pb`.
5. Seleccione el fotograma 1 en la línea de tiempo y abra el panel Acciones (Ventana > Acciones).
6. Añada el código siguiente al panel Acciones:

```
var my_pb:mx.controls.ProgressBar;
my_pb.mode = "manual";

this.createEmptyMovieClip("img_mc", 999);

var my_mcl:MovieClipLoader = new MovieClipLoader();
var mclListener:Object = new Object();
mclListener.onLoadStart = function(target_mc:MovieClip):Void {
  my_pb.label = "loading: " + target_mc._name;
};
mclListener.onLoadProgress = function(target_mc:MovieClip,
  numBytesLoaded:Number, numBytesTotal:Number):Void {
  var pctLoaded:Number = Math.ceil(100 * (numBytesLoaded /
  numBytesTotal));
  my_pb.setProgress(numBytesLoaded, numBytesTotal);
};
my_mcl.addListener(mclListener);
my_mcl.loadClip("http://www.helpexamples.com/flash/images/image1.jpg",
  img_mc);
```

7. Para probar el documento, seleccione Control > Probar película.

   La imagen se carga en el clip de película `img_mc`.
8. Seleccione Archivo > Configuración de publicación > Formatos y asegúrese de que las opciones SWF y HTML están seleccionadas.
9. Haga clic en Publicar y busque los archivos HTML y SWF en el disco duro.

   Se encuentran en la misma carpeta que progress.fla guardado en el paso 1.

10. Haga doble clic en el documento HTML para abrirlo en un navegador y observe la animación de la barra de progreso.

| NOTA | Cuando cargue archivos en el entorno de prueba, asegúrese de que carga un archivo sin caché de Internet y no un archivo local si desea ver el funcionamiento de la barra de progreso. Los archivos locales se cargan demasiado rápido, lo que impide ver el progreso. Como alternativa, cargue el archivo SWF y compruebe el documento en un servidor. |
|---|---|

Para obtener información relacionada, consulte "Carga de archivos SWF y la línea de tiempo raíz" en la página 633. Para más información sobre la clase MovieClipLoader, consulte `%{MovieClipLoader}%` en *Referencia del lenguaje ActionScript 2.0*. Para obtener información sobre la creación de la animación de una barra de progreso, consulte "Creación de una animación de progreso para cargar archivos SWF y de imagen" en la página 663.

Puede encontrar muestras de aplicaciones de galerías de fotos en el disco duro. Estos archivos proporcionan ejemplos de cómo utilizar ActionScript para controlar dinámicamente clips de película mientras se cargan archivos de imagen en un archivo SWF. Los archivos de origen de muestra gallery_tree.fla y gallery_tween.fla se pueden encontrar en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Galleries.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Galleries.

## Carga de archivos SWF y la línea de tiempo raíz

La propiedad de ActionScript `_root`, especifica o devuelve una referencia a la línea de tiempo raíz de un archivo SWF. Si carga un archivo SWF en un clip de película de otro archivo SWF, las referencias a `_root` del archivo SWF cargado se resolverán en la línea de tiempo raíz del archivo SWF principal, no en la línea de tiempo del archivo SWF cargado. En ocasiones, esta acción puede provocar un comportamiento inesperado en tiempo de ejecución (por ejemplo, cuando el archivo SWF principal y el archivo SWF cargado utilizan la propiedad `_root` para especificar una variable).

En Flash Player 7 y versiones posteriores, puede utilizar la propiedad `%{_lockroot (propiedad MovieClip._lockroot)}%` para conseguir que las referencias hechas a `_root` en un clip de película se resuelvan en la línea de tiempo del clip, en lugar de en la línea de tiempo del archivo SWF que contiene dicho clip de película. Para más información, consulte "Especificación de una línea de tiempo raíz para archivos SWF cargados" en la página 378. Para más información sobre la utilización de `_root` y `_lockroot`, consulte el Capítulo 19, "Recomendaciones y convenciones de codificación para ActionScript 2.0", en la página 775.

Un archivo SWF puede cargar otro archivo SWF desde cualquier ubicación de Internet. No obstante, para que un SWF pueda acceder a los datos (variables, métodos, etc.) definidos en el otro SWF, ambos archivos deben originarse en el mismo dominio. En Flash Player 7 y versiones posteriores, se prohíbe la creación de scripts en varios dominios, a menos que se especifique lo contrario en el archivo SWF cargado llamando a `System.security.allowDomain()`.

Para más información sobre System.security.allowDomain, consulte `%{allowDomain (método security.allowDomain)}%` en *Referencia del lenguaje ActionScript 2.0* y "Dominios, seguridad entre dominios y archivos SWF" en la página 735.

# Carga y utilización de archivos MP3 externos

Para cargar archivos MP3 en tiempo de ejecución, utilice el método `loadSound()` de la clase Sound. En primer lugar, cree un objeto Sound, como se muestra en el siguiente ejemplo:

```
var song1_sound:Sound = new Sound();
```

Utilice el nuevo objeto para llamar a `loadSound()` a fin de cargar un evento o un flujo de sonido. Los sonidos de evento se cargan completamente antes de reproducirse; los flujos de sonido se reproducen a medida que se descargan. Puede establecer el parámetro `isStreaming` del método `loadSound()` para especificar un sonido como un flujo de sonido o un sonido de evento. Tras cargar un sonido de evento, debe llamar al método `start()` de la clase Sound para reproducir el sonido. Los flujos de sonido empiezan a reproducirse cuando se han cargado suficientes datos en el archivo SWF; no es necesario utilizar `start()`.

Por ejemplo, el código siguiente crea un objeto Sound, denominado `my_sound`, y después carga un archivo MP3 denominado song1.mp3. Incluya el código ActionScript siguiente en el fotograma 1 de la línea de tiempo:

```
var my_sound:Sound = new Sound();
my_sound.loadSound("http://www.helpexamples.com/flash/sound/song1.mp3",
  true);
```

En la mayoría de los casos, deberá establecer el parámetro `isStreaming` con el valor `true`, especialmente si carga grandes archivos de sonido que deben empezar a reproducirse lo antes posible, por ejemplo al crear una aplicación MP3 "jukebox". No obstante, si descarga clips de sonido más cortos y tiene que reproducirlos en un momento concreto (por ejemplo, cuando un usuario hace clic en un botón), establezca `isStreaming` en `false`.

Para determinar si un sonido se ha descargado por completo, utilice el controlador de eventos `Sound.onLoad`. Este controlador de eventos recibe automáticamente un valor booleano (`true` o `false`) que indica si el archivo se ha descargado correctamente.

Para más información, consulte los siguientes temas:

- "Carga de un archivo MP3" en la página 635
- "Precarga de archivos MP3" en la página 636
- "Lectura de etiquetas ID3 en archivos MP3" en la página 638

El archivo de origen de muestra que carga archivos MP3, jukebox.fla, se puede encontrar en la carpeta Samples del disco duro. Este ejemplo muestra cómo crear un jukebox mediante el uso de tipos de datos, principios generales de programación y varios componentes:

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\Components\Jukebox.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/Components/Jukebox.

## Carga de un archivo MP3

Imagine que está creando un juego en línea que utiliza diferentes sonidos en función del nivel que haya alcanzado el usuario en el juego. El código siguiente carga un archivo MP3 (song2.mp3) en el objeto Sound `game_sound` y reproduce el sonido al terminar de descargarse.

### Para cargar un archivo MP3:

1. Cree un nuevo archivo FLA denominado **loadMP3.fla**.
2. Seleccione el fotograma 1 de la línea de tiempo y escriba el siguiente código en el panel Acciones:

```
var game_sound:Sound = new Sound();
game_sound.onLoad = function(success:Boolean):Void {
  if (success) {
    trace("Sound Loaded");
    game_sound.start();
  }
};
game_sound.loadSound("http://www.helpexamples.com/flash/sound/song2.mp3"
  false);‘
```

3. Seleccione Control > Probar película para probar el sonido.

Flash Player sólo admite el tipo de archivo de sonido MP3 para cargar archivos de sonido en tiempo de ejecución.

Para más información, consulte `Sound.loadSound()`, `Sound.start()` y `Sound.onLoad` en *Referencia del lenguaje ActionScript 2.0*. Para obtener información sobre la precarga de archivos MP3, consulte "Precarga de archivos MP3" en la página 636. Para obtener información sobre la creación de la animación de una barra de progreso al cargar un archivo de sonido, consulte "Creación de una barra de progreso para cargar archivos MP3 con ActionScript" en la página 665.

El archivo de origen de muestra que carga archivos MP3, jukebox.fla, se puede encontrar en la carpeta Samples del disco duro. Este ejemplo muestra cómo crear un jukebox mediante el uso de tipos de datos, principios generales de programación y varios componentes.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\Components\Jukebox.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/Components/Jukebox.

# Precarga de archivos MP3

Al precargar archivos MP3, puede utilizar la función `setInterval()` para crear un *mecanismo de sondeo* que compruebe los bytes cargados para un objeto Sound o NetStream en intervalos predeterminados. Para realizar un seguimiento del progreso de descarga de los archivos MP3, utilice los métodos `Sound.getBytesLoaded()` y `Sound.getBytesTotal()`.

El ejemplo siguiente utiliza `setInterval()` para comprobar los bytes cargados para un objeto Sound a intervalos predeterminados.

### Para precargar un archivo MP3:

1. Cree un nuevo archivo FLA denominado **preloadMP3.fla**.
2. Seleccione el fotograma 1 de la línea de tiempo y escriba el siguiente código en el panel Acciones:

```
// Cree un nuevo objeto Sound para reproducir el sonido.
var songTrack:Sound = new Sound();
// Cree la función de sondeo para realizar un seguimiento del progreso de
  la descarga.
// Ésta es la función con la que se realiza el sondeo. Dicha función
  comprueba
```

```
// el progreso de la descarga del objeto Sound pasado como referencia.
function checkProgress (soundObj:Object):Void {
  var numBytesLoaded:Number = soundObj.getBytesLoaded();
  var numBytesTotal:Number = soundObj.getBytesTotal();
  var numPercentLoaded:Number = Math.floor(numBytesLoaded /
  numBytesTotal * 100);
  if (!isNaN(numPercentLoaded)) {
    trace(numPercentLoaded + "% loaded.");
  }
};
//Cuando el archivo se termine de cargar, borre el sondeo de intervalo.
songTrack.onLoad = function ():Void {
  trace("load complete");
  clearInterval(poll);
};
// Cargue un archivo MP3 de flujo y empiece a llamar a checkProgress()
songTrack.loadSound("http://www.helpexamples.com/flash/sound/song1.mp3",
  true);
var poll:Number = setInterval(checkProgress, 100, songTrack);
```

3. Seleccione Control > Probar película para probar el sonido.

   El panel Salida muestra el progreso de la carga.

Puede utilizar la técnica de sondeo para precargar archivos FLV externos. Para obtener los bytes totales y el número de bytes cargados actualmente de un archivo FLV, utilice las propiedades `NetStream.bytesLoaded` y `NetStream.bytesTotal` (para más información, consulte %{bytesLoaded (propiedad NetStream.bytesLoaded)}% y %{bytesTotal (propiedad NetStream.bytesTotal)}% ).

Para más información, consulte `MovieClip.getBytesLoaded()`, `MovieClip.getBytesTotal()`, `setInterval()`, `Sound.getBytesLoaded()` y `Sound.getBytesTotal()` en *Referencia del lenguaje ActionScript 2.0*.

Para obtener información sobre la creación de la animación de una barra de progreso, consulte "Creación de una barra de progreso para cargar archivos MP3 con ActionScript" en la página 665.

El archivo de origen de muestra que carga archivos MP3, jukebox.fla, se puede encontrar en la carpeta Samples del disco duro. Este ejemplo muestra cómo crear un jukebox mediante el uso de tipos de datos, principios generales de programación y varios componentes.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\Components\Jukebox.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/Components/Jukebox.

# Lectura de etiquetas ID3 en archivos MP3

Las etiquetas ID3 son campos de datos añadidos a un archivo MP3, que contienen información sobre el archivo, como el título de una canción, el álbum y el artista.

Para leer etiquetas ID3 en un archivo MP3, utilice la propiedad `Sound.id3`, cuyas propiedades corresponden a los nombres de las etiquetas ID3 incluidas en el archivo MP3 que carga. Para determinar cuándo están disponibles las etiquetas ID3 para un archivo MP3 que se está descargando, utilice el controlador de eventos `Sound.onID3`. Flash Player 7 admite las etiquetas de las versiones 1.0, 1.1, 2.3 y 2.4; las etiquetas de la versión 2.2 no son compatibles.

El siguiente ejemplo carga un archivo MP3 denominado song1.mp3 en el objeto `song_sound` Sound. Cuando están disponibles las etiquetas ID3 para el archivo, aparece el campo de texto `display_txt` que muestra el nombre del artista y el título de la canción.

### Para leer etiquetas ID3 de un archivo MP3:

1. Cree un nuevo archivo FLA denominado **id3.fla**.
2. Seleccione el fotograma 1 de la línea de tiempo y escriba el siguiente código en el panel Acciones:

```
this.createTextField("display_txt", this.getNextHighestDepth(), 0, 0,
  100, 100);
display_txt.autoSize = "left";
display_txt.multiline = true;
var song_sound:Sound = new Sound();
song_sound.onLoad = function() {
  song_sound.start();
};
song_sound.onID3 = function():Void  {
  display_txt.text += "Artist:\t" + song_sound.id3.artist + "\n";
  display_txt.text += "Song:\t" + song_sound.id3.songname + "\n";
};
song_sound.loadSound("http://www.helpexamples.com/flash/sound/
  song1.mp3");
```

3. Seleccione Control > Probar película para probar el sonido.

   Aparece la etiqueta ID3 en el escenario y se reproduce el sonido.

Dado que las etiquetas ID3 2.0 se encuentran al principio de un archivo MP3 (antes de los datos de sonido), dichas etiquetas estarán disponibles en cuanto el archivo empiece a descargarse. No obstante, las etiquetas ID3 1.0 se encuentran al final del archivo (después de los datos de sonido) y, por lo tanto, no están disponibles hasta que el archivo MP3 termina de descargarse.

Cada vez que hay disponibles nuevos datos ID3, se llama al controlador de eventos `onID3`. Por lo tanto, si un archivo MP3 contiene etiquetas ID3 2.0 y 1.0, se llama dos veces al controlador `onID3`, ya que las etiquetas se encuentran en sitios distintos del archivo.

Para obtener una lista de etiquetas ID3 admitidas, consulte `%{id3 (propiedad Sound.id3)}%` en *Referencia del lenguaje ActionScript 2.0*.

El archivo de origen de muestra que carga archivos MP3, jukebox.fla, se puede encontrar en la carpeta Samples del disco duro. Este ejemplo muestra cómo crear un jukebox mediante el uso de tipos de datos, principios generales de programación y varios componentes:

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\Components\Jukebox.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/Components/Jukebox.

# Asignación de vinculación a elementos de la biblioteca

Puede asignar identificadores de vinculación a elementos de la biblioteca, como clips de película y símbolos de fuente. En Flash Basic 8 y Flash Professional 8, puede establecer identificadores de vinculación a elementos de imagen y sonido de la biblioteca. De este modo, se admite el uso de archivos de imagen y sonido con bibliotecas compartidas, así como con la nueva clase BitmapData.

El ejemplo siguiente añade una imagen de mapa de bits a la biblioteca con una vinculación establecida a `myImage`. A continuación, se añade la imagen al escenario y se incorpora la capacidad de que se pueda arrastrar.

### Para utilizar la vinculación con archivos de mapa de bits:

1. Cree un nuevo archivo FLA denominado **linkBitmap.fla**.
2. Importe una imagen de mapa de bits a la biblioteca.
3. Haga clic con el botón derecho del ratón (Windows) o con la tecla Control presionada (Macintosh) en la imagen de la biblioteca y seleccione Vinculación en el menú contextual.
4. Seleccione Exportar para ActionScript y Exportar en primer fotograma y escriba **myImage** en el cuadro de texto Identificador.
5. Haga clic en Aceptar para establecer el identificador de vinculación.

6. Seleccione el fotograma 1 de la línea de tiempo y escriba el siguiente código en el panel Acciones:

```
import flash.display.BitmapData;
// Crear imageBmp y asociar el mapa de bits de la biblioteca.
var imageBmp:BitmapData = BitmapData.loadBitmap("myImage");
// crear clip de película y asociar imageBmp
this.createEmptyMovieClip("imageClip", 10);
imageClip.attachBitmap(imageBmp, 2);
// añadir al clip la capacidad de poder arrastrarse
imageClip.onPress = function() {
     this.startDrag();
};
imageClip.onRelease = function() {
     this.stopDrag();
}
```

7. Seleccione Control > Probar película para probar el documento.

   El mapa de bits de la biblioteca aparece en el escenario y la imagen se puede arrastrar.

# Utilización de vídeo FLV

El formato del archivo FLV contiene datos de audio y vídeo codificados para publicar mediante Flash Player. Por ejemplo, si dispone de un archivo de vídeo de QuickTime o Windows Media, se utiliza un codificador (como Flash 8 Video Encoder o Sorensen Squeeze) para convertirlo en un archivo FLV.

Flash Player 7 admite archivos FLV que están codificados con códec de vídeo Sorenson Spark. Flash Player 8 admite archivos FLV codificados con el codificador Sorenson Spark u On2 VP6 en Flash Professional 8. El códec de vídeo On2 VP6 admite un canal alfa. Distintas versiones de Flash Player admiten FLV de varias formas. Para más información, consulte la siguiente tabla:

| Códec | Versión del archivo SWF (versión de publicación) | Versión de Flash Player necesaria para la reproducción |
|---|---|---|
| Sorenson Spark | 6 | 6, 7 u 8. |
| | 7 | 7, 8 |
| On2 VP6 | 6 | 8* |
| | 7 | 8 |
| | 8 | 8 |

* Si su archivo SWF carga un archivo FLV, puede utilizar vídeo On2 VP6 sin tener que volver a publicar el archivo SWF para Flash Player 8, siempre que los usuarios utilicen Flash Player 8 para ver el archivo SWF. Sólo Flash Player 8 admite tanto la publicación como la reproducción de vídeo On2 VP6.

Para obtener información sobre los conceptos básicos del vídeo, como aspectos relacionados con el flujo, la descarga progresiva, las dimensiones, la codificación, la importación y el ancho de banda, consulte el Capítulo 11, "Trabajo con vídeo" en *Utilización de Flash*.

En esta sección se describe el uso del vídeo FLV sin componentes. También puede utilizar el componente FLVPlayback para reproducir archivos FLV o emplear la clase VideoPlayback para crear un reproductor de vídeo personalizado que cargue archivos FLV dinámicamente (consulte www.macromedia.com/devnet o www.macromedia.com/support/documentation/). Para obtener información sobre la utilización de vídeo FLV con los componentes FLVPlayback y Media, consulte las secciones siguientes:

- "Componente FLVPlayback (sólo en Flash Professional)" en la página 527
- "Componentes multimedia (sólo en Flash Professional)" en la página 867

En lugar de importar vídeo directamente en el entorno de edición de Flash, puede utilizar código de ActionScript para reproducir de forma dinámica archivos FLV externos en Flash Player. Los archivos FLV se pueden reproducir desde una dirección HTTP o desde el sistema de archivos local. Para reproducir archivos FLV, utilice las clases NetConnection y NetStream y el método `attachVideo()` de la clase Video. Para más información, consulte `%{NetConnection}%`, `%{NetStream}%` y `%{attachVideo (método Video.attachVideo)}%` en *Referencia del lenguaje ActionScript 2.0*.

Puede crear archivos FLV importando vídeo a la herramienta de edición de Flash y exportándolo como archivo FLV. En Flash Professional 8, puede utilizar el complemento de exportación de FLV para exportar archivos FLV de aplicaciones de edición de vídeo compatibles.

La utilización de archivos FLV externos ofrece algunas posibilidades que no están disponibles al utilizar vídeo importado:

- Pueden utilizarse clips de vídeo más largos en documentos de Flash sin que ello ralentice la reproducción. Los archivos FLV externos se reproducen utilizando la *memoria caché*, de modo que los archivos grandes se almacenan en partes pequeñas y se accede a ellos de forma dinámica; además, requieren menos memoria que los archivos de vídeo incorporados.
- Un archivo FLV externo puede tener una velocidad de fotogramas distinta a la del documento de Flash en el que se reproduce. Por ejemplo, puede establecer la velocidad de fotogramas del documento de Flash en 30 fotogramas por segundo (fps) y la velocidad de fotogramas del vídeo en 21 fps. Esta opción le permite un mejor control del vídeo que el vídeo incorporado, de modo que garantiza una reproducción del vídeo sin problemas. Asimismo, permite reproducir archivos FLV a distintas velocidades de fotogramas sin necesidad de alterar el contenido de Flash existente.

- Con archivos FLV externos no es preciso interrumpir la reproducción de los documentos de Flash mientras se carga el archivo de vídeo. A veces, los archivos de vídeo importados pueden interrumpir la reproducción de un documento para realizar ciertas funciones, como acceder a una unidad de CD-ROM. Los archivos FLV pueden realizar funciones independientemente del documento de Flash, por lo que no interrumpen su reproducción.
- La rotulación de contenido de vídeo es más fácil con los archivos FLV, debido a que se pueden utilizar controladores de eventos para acceder a los metadatos del vídeo.

| SUGERENCIA | Para cargar archivos FLV de un servidor Web, puede que necesite registrar la extensión de archivo y el tipo MIME en el servidor Web; compruebe la documentación del servidor Web. El tipo MIME de los archivos FLV es video/x-flv. Para más información, consulte "Configuración de archivos FLV para alojar en el servidor" en la página 660. |
|---|---|

Para más información sobre el vídeo FLV, consulte los temas siguientes:



## Creación de un objeto de vídeo

Antes de poder cargar y manipular vídeo con ActionScript, debe crear un objeto de vídeo, arrastrarlo al escenario y asignarle un nombre de instancia. En el ejemplo siguiente se describe cómo añadir una instancia de vídeo a una aplicación.

### Para crear un objeto de vídeo:

1. Con un documento abierto en la herramienta de edición de Flash, seleccione Nuevo Vídeo en el menú emergente del panel Biblioteca (Ventana > Biblioteca).
2. En el cuadro de diálogo Propiedades de vídeo, asigne un nombre al símbolo de vídeo y seleccione Vídeo (controlado por ActionScript).
3. Haga clic en Aceptar para crear el objeto de vídeo.

4. Arrastre el objeto de vídeo desde el panel Biblioteca hasta el escenario para crear una instancia del mismo.
5. Con el objeto de vídeo seleccionado en el escenario, introduzca **my_video** en el cuadro de texto Nombre de instancia del inspector de propiedades (Ventana > Propiedades > Propiedades).

   Ahora dispone de una instancia de vídeo en el escenario, a la que puede añadir ActionScript para cargar vídeo o manipular la instancia de diversas formas.

Para obtener información sobre la carga de archivos FLV de forma dinámica, consulte "Reproducción dinámica de archivos FLV externos". Para obtener información sobre la creación de un anuncio de vídeo, consulte "Creación de un anuncio de vídeo" en la página 644.

## Reproducción dinámica de archivos FLV externos

Puede cargar archivos FLV en tiempo de ejecución para reproducir en un archivo SWF. Se pueden cargar en un objeto de vídeo o en un componente como FLVPlayback. El ejemplo siguiente muestra cómo reproducir un archivo denominado clouds.flv en un objeto de vídeo.

### Para reproducir un archivo FLV externo en un documento de Flash:

1. Cree un nuevo documento de Flash denominado **playFLV.fla**.
2. En el panel Biblioteca (Ventana > Biblioteca), seleccione Nuevo Vídeo en el menú emergente Biblioteca.
3. En el cuadro de diálogo Propiedades de vídeo, asigne un nombre al símbolo de vídeo y seleccione Vídeo (controlado por ActionScript).
4. Haga clic en Aceptar para crear el objeto de vídeo.
5. Arrastre el objeto de vídeo desde el panel Biblioteca hasta el escenario para crear una instancia del mismo.
6. Con el objeto de vídeo seleccionado en el escenario, introduzca **my_video** en el cuadro de texto Nombre de instancia del inspector de propiedades (Ventana > Propiedades > Propiedades).
7. Seleccione el fotograma 1 en la línea de tiempo y abra el panel Acciones (Ventana > Acciones).

8. Introduzca el código siguiente en el panel Acciones:

```
this.createTextField("status_txt", 999, 0, 0, 100, 100);
status_txt.autoSize = "left";
status_txt.multiline = true;
// Cree un objeto NetConnection
var my_nc:NetConnection = new NetConnection();
// Cree una conexión de transmisión local
my_nc.connect(null);
// Cree un objeto NetStream y defina una función onStatus()
var my_ns:NetStream = new NetStream(my_nc);
my_ns.onStatus = function(infoObject:Object):Void {
  status_txt.text += "status (" + this.time + " seconds)\n";
  status_txt.text += "\t Level: " + infoObject.level + "\n";
  status_txt.text += "\t Code: " + infoObject.code + "\n\n";
};
// Asocie la salida de vídeo NetStream al objeto Video
my_video.attachVideo(my_ns);
// Establezca el tiempo de búfer
my_ns.setBufferTime(5);
// Comience a reproducir el archivo FLV
my_ns.play("http://www.helpexamples.com/flash/video/clouds.flv");
```

9. Seleccione Control > Probar película para probar el documento.

Para obtener información sobre la precarga de archivos FLV, consulte "Precarga de archivos FLV" en la página 507. Para obtener información sobre la carga dinámica de vídeo FLV en componentes, consulte "Creación de aplicaciones con el componente FLVPlayback" en la página 529. Para obtener información sobre archivos FLV y el servidor, así como archivos FLV y la reproducción local de archivos FLV en Macintosh, consulte "Configuración de archivos FLV para alojar en el servidor" en la página 660.

## Creación de un anuncio de vídeo

El contenido de vídeo en anuncios de Flash se utiliza a menudo en la publicidad, como mostrar preestrenos de películas o anuncios de televisión de Flash. El siguiente ejemplo muestra cómo podría crear una instancia de vídeo y añadir ActionScript en un archivo FLA para crear un anuncio que contenga vídeo.

### Para crear un anuncio de vídeo:

1. Cree un nuevo documento de Flash denominado **vidBanner.fla**.
2. Seleccione Modificar > Documento.
3. Cambie las dimensiones del archivo FLA, introduzca **468** en el cuadro de texto de anchura y **60** en el cuadro de texto de altura.
4. En el panel Biblioteca (Ventana > Biblioteca), seleccione Nuevo Vídeo en las opciones del menú emergente Biblioteca.

5. En el cuadro de diálogo Propiedades de vídeo, asigne un nombre al símbolo de vídeo y seleccione Vídeo (controlado por ActionScript).
6. Haga clic en Aceptar para crear el objeto de vídeo.
7. Arrastre el objeto de vídeo desde el panel Biblioteca hasta el escenario para crear una instancia de vídeo.
8. Con el objeto de vídeo seleccionado en el escenario, introduzca **my_video** en el cuadro de texto Nombre de instancia del inspector de propiedades (Ventana > Propiedades > Propiedades).
9. Con la instancia de vídeo aún seleccionada, introduzca **105** en el cuadro de texto de anchura y **60** en el cuadro de texto de altura del inspector de propiedades.
10. Arrastre la instancia de vídeo a la posición del escenario o utilice el inspector de propiedades para establecer las coordenadas *x* e *y*.
11. Seleccione el fotograma 1 en la línea de tiempo y abra el panel Acciones (Ventana > Acciones).
12. Añada el código siguiente al panel Acciones:

```
var my_nc:NetConnection = new NetConnection();
my_nc.connect(null);
var my_ns:NetStream = new NetStream(my_nc);
my_video.attachVideo(my_ns);
my_ns.setBufferTime(5);
my_ns.play("http://www.helpexamples.com/flash/video/vbanner.flv");
```

13. Seleccione Insertar > Línea de tiempo > Capa para crear una nueva capa y asígnele el nombre **button**.
14. Seleccione la herramienta Rectángulo en el panel Herramientas.
15. En la sección Colores del panel Herramientas, haga clic en el icono de lápiz para seleccionar el control Color de trazo.
16. Seleccione Sin color, de modo que se desactiva el contorno del rectángulo.
17. Arrastre el puntero diagonalmente por el escenario para crear un rectángulo.

    El tamaño del rectángulo no importa ya que cambiará las dimensiones con el inspector de propiedades.
18. Haga clic en la herramienta Selección del panel Herramientas y, a continuación, haga clic en el rectángulo del escenario para seleccionarlo.
19. Con el rectángulo seleccionado, introduzca **468** en el cuadro de texto de anchura y **60** en el cuadro de texto de altura del inspector de propiedades. A continuación, cambie las coordenadas X e Y (cuadros X e Y) a **0**.
20. Con el rectángulo seleccionado en el escenario, presione F8 para cambiar el rectángulo a un símbolo.

21. En el cuadro de diálogo Convertir en símbolo, escriba **invisible btn** en el cuadro de texto Nombre, seleccione Button y haga clic en Aceptar.

22. Haga doble clic en el nuevo botón del escenario para introducir el modo de edición de símbolos.

    El rectángulo se encuentra actualmente en el primer fotograma Arriba del botón creado. Se trata del estado Arriba del botón: lo que los usuarios ven cuando el botón está en el escenario. Sin embargo, si desea que no se vea el botón en el escenario, debe mover el rectángulo al fotograma Zona activa, que es la zona activa del botón (la región activa en la que el usuario puede hacer clic para activar las acciones del botón).

23. Haga clic en el fotograma clave en el fotograma Arriba y mantenga pulsado el botón del ratón mientras arrastra el fotograma clave al fotograma Zona activa

    Ahora puede hacer clic en toda la zona del anuncio, pero no se verá ningún botón en el mismo.

24. Haga clic en Escena 1 para volver a la línea de tiempo principal.

    Aparece un rectángulo en el área del anuncio que representa la zona activa invisible del botón.

25. Seleccione el botón creado, abra el inspector de propiedades y escriba **inv_btn** en el cuadro de texto Nombre de instancia.

26. Seleccione el fotograma 1 de la línea de tiempo y escriba el siguiente código en el panel Acciones:

```
inv_btn.onRelease = function(){
  getURL("http://www.macromedia.com");
};
```

27. Realice otras modificaciones al anuncio, como añadir gráficos o texto.

28. Seleccione Control > Probar película para probar el anuncio en Flash Player.

En este ejemplo, ha creado un anuncio y cambiado las dimensiones a las establecidas y estandarizadas que especifica la IAB (Interactive Advertising Bureau, Agencia de la publicidad interactiva). Para obtener información sobre las dimensiones estándar de los anuncios (así como otras directrices útiles), consulte la página de estándares y directrices de IAB en www.iab.net/standards/adunits.asp.

Además de las directrices estandarizadas, asegúrese de que confirma las directrices de publicidad del servicio, cliente o sitio Web del que está realizando la publicidad en primer lugar. Si envía el anuncio a una compañía de publicidad, asegúrese de que el archivo cumple una directriz especificada de tamaño de archivo, dimensión, versión para Flash Player y velocidad de fotogramas. Además, puede que tenga que tener en cuenta reglas sobre los tipos de archivos multimedia que utiliza, el código de botones del archivo FLA, etc.

# Precarga de archivos FLV

Para realizar un seguimiento del progreso de descarga de los archivos FLV, utilice las propiedades `NetStream.bytesLoaded` y `NetStream.bytesTotal`. Para obtener los bytes totales y el número actual de bytes cargados de un archivo FLV, utilice las propiedades `NetStream.bytesLoaded` y `NetStream.bytesTotal`.

El siguiente ejemplo utiliza las propiedades `bytesLoaded` y `bytesTotal` que muestran el progreso de carga de video1.flv en la instancia de objeto de vídeo denominada my_video. Asimismo se crea de forma dinámica un campo de texto llamado loaded_txt para ver información sobre el proceso de carga.

### Para precargar un archivo FLV:

1. Cree un nuevo archivo FLA denominado **preloadFLV.fla**.
2. En el panel Biblioteca (Ventana > Biblioteca), seleccione Nuevo Vídeo en el menú emergente Biblioteca.
3. En el cuadro de diálogo Propiedades de vídeo, asigne un nombre al símbolo de vídeo y seleccione Vídeo (controlado por ActionScript).
4. Haga clic en Aceptar para crear el objeto de vídeo.
5. Arrastre el objeto de vídeo desde el panel Biblioteca hasta el escenario para crear una instancia del mismo.
6. Con el objeto de vídeo seleccionado en el escenario, introduzca **my_video** en el cuadro de texto Nombre de instancia del inspector de propiedades (Ventana > Propiedades > Propiedades).
7. Con la instancia de vídeo aún seleccionada, introduzca **320** en el cuadro de texto de anchura y **213** en el cuadro de texto de altura del inspector de propiedades.
8. Seleccione el fotograma 1 en la línea de tiempo y abra el panel Acciones (Ventana > Acciones).
9. Introduzca el código siguiente en el panel Acciones:

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("http://www.helpexamples.com/flash/video/
  lights_short.flv");
```

```
this.createTextField("loaded_txt", this.getNextHighestDepth(), 10, 10,
  160, 22);
var loaded_interval:Number = setInterval(checkBytesLoaded, 500,
  stream_ns);
function checkBytesLoaded(my_ns:NetStream) {
  var pctLoaded:Number = Math.round(my_ns.bytesLoaded / my_ns.bytesTotal
  * 100);
  loaded_txt.text = Math.round(my_ns.bytesLoaded / 1000) + " of " +
  Math.round(my_ns.bytesTotal / 1000) + " KB loaded (" + pctLoaded +
  "%)";
  progressBar_mc.bar_mc._xscale = pctLoaded;
  if (pctLoaded >= 100) {
    clearInterval(loaded_interval);
  }
}
```

**10.** Seleccione Control > Probar película para probar el código.

| NOTA | Si la barra de progreso se carga de forma instantánea, el vídeo se ha almacenado en la caché del disco duro (al probar este ejemplo o al cargarlo mediante otro procedimiento). Si ocurre esto, cargue un archivo FLV en el servidor y cárguelo en su lugar. |
|---|---|

Otra manera de precargar archivos FLV consiste en utilizar el método `NetStream.setBufferTime()`. Este método requiere un solo parámetro que indica el número de segundos que debe almacenarse en el búfer el flujo FLV antes comenzar la reproducción. Para más información, consulte `%{setBufferTime (método NetStream.setBufferTime)}%`, `%{getBytesLoaded (método MovieClip.getBytesLoaded)}%`, `%{getBytesTotal (método MovieClip.getBytesTotal)}%`, `%{bytesLoaded (propiedad NetStream.bytesLoaded)}%`, `%{bytesTotal (propiedad NetStream.bytesTotal)}%` y `%{setInterval function}%` en *Referencia del lenguaje ActionScript 2.0*

# Trabajo con puntos de referencia (cuepoints)

Puede utilizar diferentes tipos de puntos de referencia o cuepoints con Flash Video. ActionScript permite interactactuar con cuepoints que incorpore en un archivo FLV (al crear el archivo FLV) o que cree mediante ActionScript.

**Cuepoints de navegación** Los puntos de referencia o cuepoints de navegación del flujo FLV y el paquete de metadatos FLV se incorporan al codificar el archivo FLV. Los cuepoints de navegación se utilizan para permitir a los usuarios buscar una parte especificada de un archivo.

**Cuepoints de evento** Los puntos de referencia o cuepoints de evento del flujo FLV y el paquete de metadatos FLV se incorporan al codificar el archivo FLV. Puede escribir código para controlar eventos que se activan en puntos especificados durante la reproducción de FLV.

**Cuepoints de ActionScript** Cuepoints externos que se crean mediante código ActionScript. Puede escribir código para activar estos cuepoints en relación a la reproducción de vídeo. Estos cuepoints son menos precisos que los cuepoints incorporados (hasta una décima de segundo), ya que el reproductor de vídeo realiza un seguimiento de los mismos de forma independiente.

Los cuepoints de navegación crean un fotograma clave en una ubicación de cuepoint especificada, por lo que puede utilizar código para desplazar la cabeza lectora del reproductor de vídeo a dicha ubicación. Puede establecer puntos determinados en un archivo FLV donde desee que busquen los usuarios. Por ejemplo, si el vídeo incluyera varios capítulos o segmentos, podría controlarlo mediante la incorporación de cuepoints de navegación en el archivo de vídeo.

Si va a crear una aplicación en la que desea que los usuarios naveguen a un cuepoint, debe crear e incorporar cuepoints al codificar el archivo en lugar de utilizar cuepoints de ActionScript. Se recomienda que incorpore los cuepoints en el archivo FLV, ya que resultan más precisos para trabajar con ellos. Para más información sobre la codificación de archivos FLV con cuepoints, consulte "Inserción de cuepoints (puntos de referencia) (sólo en Flash Professional)" en la página 329 en *Utilización de Flash*.

Puede acceder a parámetros de cuepoint al escribir código ActionScript. Los parámetros de cuepoint constituyen una parte del objeto de evento recibido con el evento `cuePoint` (`event.info.parameters`).

## Trazado de cuepoints de un archivo FLV

Puede trazar los cuepoints incorporados en un documento FLV mediante `NetStream.onMetaData`. Necesita buscar recursivamente la estructura de los metadatos que vuelven para ver la información del cuepoint.

El siguiente código traza cuepoints en un archivo FLV:

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
stream_ns.onMetaData = function(metaProp:Object) {
  trace("The metadata:");
  traceMeta(metaProp);
  // traceObject(metaProp, 0);
};
my_video.attachVideo(stream_ns);
stream_ns.play("http://www.helpexamples.com/flash/video/cuepoints.flv");
```

```
function traceMeta(metaProp:Object):Void {
  var p:String;
  for (p in metaProp) {
    switch (p) {
    case "cuePoints" :
      trace("cuePoints: ");
      //cycles through the cue points
      var cuePointArr:Array = metaProp[p];
      for (var j:Number = 0; j < cuePointArr.length; j++) {
        //cycle through the current cue point parameters
        trace("\t cuePoints[" + j + "]:");
        var currentCuePoint:Object = metaProp[p][j];
        var metaPropPJParams:Object = currentCuePoint.parameters;
        trace("\t\t name: " + currentCuePoint.name);
        trace("\t\t time: " + currentCuePoint.time);
        trace("\t\t type: " + currentCuePoint.type);
        if (metaPropPJParams != undefined) {
          trace("\t\t parameters:");
          traceObject(metaPropPJParams, 4);
        }
      }
      break;
    default :
      trace(p + ": " + metaProp[p]);
      break;
    }
  }
}
function traceObject(obj:Object, indent:Number):Void {
  var indentString:String = "";
  for (var j:Number = 0; j < indent; j++) {
    indentString += "\t";
  }
  for (var i:String in obj) {
    if (typeof(obj[i]) == "object") {
      trace(indentString + " " + i + ": [Object]");
      traceObject(obj[i], indent + 1);
    } else {
      trace(indentString + " " + i + ": " + obj[i]);
    }
  }
}
```

Aparece el siguiente resultado:

```
The metadata:
canSeekToEnd: true
cuePoints:
  cuePoints[0]:
    name: point1
    time: 0.418
    type: navigation
    parameters:
      lights: beginning
  cuePoints[1]:
    name: point2
    time: 7.748
    type: navigation
    parameters:
      lights: middle
  cuePoints[2]:
    name: point3
    time: 16.02
    type: navigation
    parameters:
      lights: end
audiocodecid: 2
audiodelay: 0.038
audiodatarate: 96
videocodecid: 4
framerate: 15
videodatarate: 400
height: 213
width: 320
duration: 16.334
```

Para obtener información sobre el uso de cuepoints con el componente FLVPlayback, consulte “Utilización de puntos de referencia con el componente FLVPlayback (sólo en Flash Professional)”.

## Utilización de puntos de referencia con el componente FLVPlayback (sólo en Flash Professional)

Puede ver cuepoints de un archivo FLV en el inspector de propiedades cuando utilice el componente FLVPlayback. Una vez establecida la propiedad `contentPath` para la instancia FLVPlayback, puede ver todos los cuepoints incorporados en el archivo de vídeo. Con la ficha Parámetros, busque la propiedad `cuePoints` y haga clic en icono de lupa para ver una lista de los cuepoints del archivo.

| NOTA | Para ver los cuepoints de la ficha Parámetros, debe escribir el nombre del archivo FLV en el cuadro de texto contentPath en lugar de utilizar código para asignar contentPath. |
|---|---|

El ejemplo siguiente muestra cómo utilizar información de cuepoints con el componente FLVPlayback.

### Para utilizar cuepoints con el componente FLVPlayback:

1. Cree un nuevo documento de Flash denominado **cueFlv.fla**.
2. Abra el panel Componentes (Ventana > Componentes) y arrastre una instancia de los componentes FLVPlayback y TextArea al escenario.
3. Seleccione el componente TextArea e introduzca **my_ta** en el cuadro de texto Nombre de instancia del inspector de propiedades (Ventana > Propiedades > Propiedades).
4. Con el componente TextArea aún seleccionado, introduzca **200** en el cuadro de texto de anchura y **100** en el cuadro de texto de altura.
5. Seleccione la instancia FLVPlayback en el escenario y escriba **my_flvPb** en el cuadro de texto Nombre de instancia.
6. Seleccione el fotograma 1 de la línea de tiempo y escriba el siguiente código en el panel Acciones.

```
var my_flvPb:mx.video.FLVPlayback;
var my_ta:mx.controls.TextArea;
my_flvPb.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object) {
  my_ta.text += "Elapsed time in seconds: " + my_flvPb.playheadTime +
  "\n";
};
my_flvPb.addEventListener("cuePoint",listenerObject);
```

7. Seleccione Control > Probar película para probar el archivo SWF.

   El tiempo transcurrido aparece en la instancia TextArea cuando la cabeza lectora pasa por cada cuepoint incorporado en el documento.

Para más información sobre la utilización del componente FLVPlayback, consulte "Componente FLVPlayback (sólo en Flash Professional)" en la página 527.

## Creación de cuepoints con ActionScript para utilizar con componentes (sólo en Flash Professional)

Puede crear cuepoints con ActionScript para utilizarlos luego con una instancia de objeto de vídeo o uno de los componentes del reproductor de vídeo (FLVPlayback para Flash Player 8 o MediaPlayback para Flash Player 7). Los ejemplos siguientes muestran lo sencillo que resulta utilizar código ActionScript para crear cuepoints y utilizar después un script para acceder a ellos.

| NOTA | Incorpore cuepoints de navegación a un documento si desea añadir la funcionalidad de navegación a la aplicación. Para más información, consulte "Trabajo con puntos de referencia (cuepoints)" en la página 648. Para obtener un ejemplo de utilización de cuepoints, consulte "Utilización de puntos de referencia con el componente FLVPlayback (sólo en Flash Professional)" en la página 652. |
|---|---|

### Para crear y utilizar cuepoints con el componente FLVPlayback:

1. Cree un nuevo documento de Flash denominado **cueFlvPb.fla**.
2. Arrastre una instancia del componente FLVPlayback del panel Componentes (Ventana > Componentes) al escenario.

   El componente se encuentra en la carpeta FLVPlayback - Player 8.
3. Seleccione el componente y abra el inspector de propiedades (Ventana > Propiedades > Propiedades).
4. Introduzca **my_flvPb** en el cuadro de texto Nombre de instancia.
5. Arrastre una instancia del componente TextArea del panel Componentes al escenario.
6. Seleccione el componente TextArea y escriba **my_ta** en el cuadro de texto Nombre de instancia.
7. Con el componente TextArea aún seleccionado, introduzca **200** en el cuadro de texto de anchura y **100** en el cuadro de texto de altura.
8. Seleccione el fotograma 1 de la línea de tiempo y escriba el siguiente código en el panel Acciones:

```
var my_flvPb:mx.video.FLVPlayback;
my_flvPb.contentPath = "http://www.helpexamples.com/flash/video/
  clouds.flv";

// Crear objeto cuePoint.
var cuePt:Object = new Object();
cuePt.time = 1;
cuePt.name = "elapsed_time";
cuePt.type = "actionscript";
// Añadir objeto cuepoint de AS.
my_flvPb.addASCuePoint(cuePt);
```

```
// Añadir otro objeto cuepoint de AS.
my_flvPb.addASCuePoint(2, "elapsed_time2");

// Mostrar información de cuepoint en el campo de texto.
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject) {
  my_ta.text += "Elapsed time in seconds: " + my_flvPb.playheadTime +
  "\n";
};
my_flvPb.addEventListener("cuePoint",listenerObject);
```

9. Seleccione Control > Probar película para probar el código.

   Los cuepoints siguientes se trazan en el panel Salida:

```
Elapsed time in seconds: 1.034
Elapsed time in seconds: 2.102
```

Para obtener información sobre `addASCuePoint()`, consulte "FLVPlayback.addASCuePoint()" en la página 576. Para obtener información sobre el trabajo con cuepoints y el componente FLVPlayback, consulte "Utilización de cuepoints" en la página 536 y "Componente FLVPlayback (sólo en Flash Professional)" en la página 527.

El ejemplo siguiente muestra cómo añadir cuepoints en tiempo de ejecución y trazar después los cuepoints cuando se reproduce un archivo FLV en el componente MediaPlayback.

### Para crear y utilizar cuepoints con el componente MediaPlayback:

1. Cree un nuevo documento de Flash denominado cuePointMP.fla.
2. Arrastre una instancia del componente MediaPlayback del panel Componentes (Ventana > Componentes) al escenario.

   El componente se encuentra en la carpeta Media - Player 6 - 7.
3. Seleccione el componente y abra el inspector de propiedades (Ventana > Propiedades > Propiedades).
4. Introduzca **my_mp** en el cuadro de texto Nombre de instancia.
5. Seleccione la ficha Parámetros y haga clic en Iniciar inspector de componentes.
6. En el inspector de componentes, introduzca **http://www.helpexamples.com/flash/video/clouds.flv** en el cuadro de texto URL.

7. Abra el panel Acciones (Ventana > Acciones) e introduzca el código siguiente en el panel Script:

```
import mx.controls.MediaPlayback;
var my_mp:MediaPlayback;
my_mp.autoPlay = false;
my_mp.addEventListener("cuePoint", doCuePoint);
my_mp.addCuePoint("one", 1);
my_mp.addCuePoint("two", 2);
my_mp.addCuePoint("three", 3);
my_mp.addCuePoint("four", 4);
function doCuePoint(eventObj:Object):Void {
     trace(eventObj.type + " = {cuePointName:" + eventObj.cuePointName +
   " cuePointTime:" + eventObj.cuePointTime + "}");
}
```

8. Seleccione Control > Probar película para probar el código.

   Los cuepoints siguientes se trazan en el panel Salida:

```
cuePoint = {cuePointName:one cuePointTime:1}
cuePoint = {cuePointName:two cuePointTime:2}
cuePoint = {cuePointName:three cuePointTime:3}
cuePoint = {cuePointName:four cuePointTime:4}
```

Para más información sobre la utilización del componente MediaPlayback, consulte "Componentes multimedia (sólo en Flash Professional)" en la página 867. Para más información sobre la utilización del componente FLVPlayback, consulte "Componente FLVPlayback (sólo en Flash Professional)" en la página 527.

## Adición de funcionalidad de búsqueda con cuepoints (sólo en Flash Professional)

Puede incorporar cuepoints de navegación en un archivo FLV para añadir funcionalidad de búsqueda a las aplicaciones. El método `seekToNavCuePoint()` del componente FLVPlayback localiza el cuepoint en el archivo FLV con el nombre especificado, en o después del tiempo indicado. Puede especificar un nombre como una cadena (como `"part1"` o `"theParty"`).

También puede utilizar el método `seekToNextNavCuePoint()`, que busca el siguiente cuepoint de navegación, en función del `playheadTime` actual. Puede pasar el método como parámetro, `time`, que es el tiempo de inicio desde donde buscar el siguiente cuepoint de navegación. El valor predeterminado es el `playheadTime` actual.

Como alternativa, también puede buscar una duración especificada del archivo FLV mediante el método `seek()`.

En los ejemplos siguientes, añadirá un botón que utilizará para saltar entre cuepoints o a una duración especificada de un archivo FLV que se reproduce en el componentes FLVPlayback, así como un botón para saltar a un cuepoint especificado.

### Para buscar una duración especificada:

1. Cree un nuevo documento de Flash denominado **seekduration.fla**.
2. Arrastre una instancia del componente FLVPlayback del panel Componentes (Ventana > Componentes) al escenario.

   El componente se encuentra en la carpeta FLVPlayback - Player 8.
3. Seleccione el componente y abra el inspector de propiedades (Ventana > Propiedades > Propiedades).
4. Introduzca **my_flvPb** en el cuadro de texto Nombre de instancia.
5. Arrastre una instancia del componente Button del panel Componentes al escenario.
6. Seleccione el componente Button y escriba **my_button** en el cuadro de texto Nombre de instancia.
7. Seleccione el fotograma 1 de la línea de tiempo y escriba el siguiente código en el panel Acciones:

```
import mx.controls.Button;
import mx.video.FLVPlayback;
var seek_button:Button;
var my_flvPb:FLVPlayback;
my_flvPb.autoPlay = false;
my_flvPb.contentPath = "http://www.helpexamples.com/flash/video/
  sheep.flv";
seek_button.label = "Seek";
seek_button.addEventListener("click", seekFlv);
function seekFlv(eventObj:Object):Void {
  // buscar 2 segundos
  my_flvPb.seek(2);
}
```

8. Seleccione Control > Probar película para probar el código.

   Al hacer clic en el botón, la cabeza lectora del vídeo se desplaza a la duración especificada: 2 segundos en el vídeo.

### Para añadir funcionalidad de búsqueda con el componente FLVPlayback:

1. Cree un nuevo documento de Flash denominado **seek1.fla**.
2. Arrastre una instancia del componente FLVPlayback del panel Componentes (Ventana > Componentes) al escenario.

   El componente se encuentra en la carpeta FLVPlayback - Player 8.
3. Seleccione el componente y abra el inspector de propiedades (Ventana > Propiedades > Propiedades).
4. Introduzca **my_flvPb** en el cuadro de texto Nombre de instancia.

5. Arrastre una instancia del componente Button del panel Componentes al escenario.
6. Seleccione el componente Button y escriba **my_button** en el cuadro de texto Nombre de instancia.
7. Seleccione el fotograma 1 de la línea de tiempo y escriba el siguiente código en el panel Acciones:

```
import mx.video.FLVPlayback;
var my_flvPb:FLVPlayback;
my_flvPb.autoPlay = false;
my_flvPb.contentPath = "http://www.helpexamples.com/flash/video/
  cuepoints.flv";
my_button.label = "Next cue point";

function clickMe(){
  my_flvPb.seekToNextNavCuePoint();
}
my_button.addEventListener("click", clickMe);
```

8. Seleccione Control > Probar película para probar el código.

   El archivo cuepoints.flv contiene tres cuepoints de navegación: uno cerca del principio, en medio y al final del archivo de vídeo. Al hacer clic en el botón, la instancia FLVPlayback busca el siguiente cuepoint hasta que llega al último cuepoint del archivo de vídeo.

También puede buscar un cuepoint especificado en un archivo FLV mediante el método `seekToCuePoint()`, como se muestra en el ejemplo siguiente.

### Para buscar un cuepoint especificado:

1. Cree un nuevo documento de Flash denominado **seek2.fla**.
2. Arrastre una instancia del componente FLVPlayback del panel Componentes (Ventana > Componentes) al escenario.

   El componente se encuentra en la carpeta FLVPlayback - Player 8.
3. Seleccione el componente y abra el inspector de propiedades (Ventana > Propiedades > Propiedades).
4. Introduzca **my_flvPb** en el cuadro de texto Nombre de instancia.
5. Con la instancia FLVPlayback aún seleccionada, haga clic en la ficha Parámetros.
6. Introduzca **http://www.helpexamples.com/flash/video/cuepoints.flv** en el cuadro de texto contentPath.

   Al introducir la URL en el cuadro de texto contentPath, los cuepoints aparecen en la ficha Parámetros (junto al parámetro cuePoint). Por tanto, puede determinar el nombre del cuepoint que desea buscar en el código. Si hace clic en el icono de lupa, puede ver todos los cuepoints del archivo de vídeo así como la información sobre cada cuepoint en una tabla.
7. Arrastre una instancia del componente Button del panel Componentes al escenario.

8. Seleccione el componente Button y escriba **my_button** en el cuadro de texto Nombre de instancia.
9. Seleccione el fotograma 1 de la línea de tiempo y escriba el siguiente código en el panel Acciones:

```
import mx.video.FLVPlayback;
var my_flvPb:FLVPlayback;
my_flvPb.autoPlay = false;
my_button.label = "Seek to point2";

function clickMe(){
  my_flvPb.seekToNavCuePoint("point2");
}
my_button.addEventListener("click", clickMe);
```

10. Seleccione Control > Probar película para probar el código.

    El archivo cuepoints.flv contiene tres cuepoints de navegación: uno cerca del principio, en medio y al final del archivo de vídeo. Al hacer clic en el botón, la instancia FLVPlayback busca hasta el cuepoint especificado (`point2`).

Para más información sobre cuepoints, consulte "Utilización de cuepoints" en la página 536. Para más información sobre el componente FLVPlayback, consulte "Componente FLVPlayback (sólo en Flash Professional)" en la página 527.

## Utilización de metadatos

Puede utilizar el método `onMetaData` para ver la información de metadatos en el archivo FLV. Los metadatos incluyen información sobre el archivo FLV, como la duración, anchura, altura y velocidad de fotograma. La información de metadatos que añade al archivo FLV depende del software que utilice para codificar el archivo o del software que emplee para añadir información de metadatos.

| NOTA | Si el archivo de vídeo no tiene información de metadatos, puede utilizar herramientas para añadirla al archivo. |
|---|---|

Para trabajar con `NetStream.onMetaData`, debe tener Flash Video que contenga metadatos. Si codifica archivos FLV con Flash 8 Video Encoder, el archivo FLV incluirá información de metadatos (consulte el ejemplo siguiente para ver una lista de metadatos en un archivo FLV codificado con Flash 8 Video Encoder).

| NOTA | Flash Video Exporter 1.2 y versiones posteriores (incluido Flash 8 Video Exporter), añaden los metadatos a los archivos FLV. Sorenson Squeeze 4.1 y versiones posteriores también añaden metadatos a los archivos de vídeo. |
|---|---|

El ejemplo siguiente utiliza `NetStream.onMetaData` para trazar la información de metadatos de un archivo FLV codificado con Flash 8 Video Encoder.

### Para utilizar NetStream.onMetaData a fin de ver información de metadatos:

1. Cree un nuevo archivo FLA denominado **flvMetadata.fla**.
2. En el panel Biblioteca (Ventana > Biblioteca), seleccione Nuevo Vídeo en el menú emergente Biblioteca.
3. En el cuadro de diálogo Propiedades de vídeo, asigne un nombre al símbolo de vídeo y seleccione Vídeo (controlado por ActionScript).
4. Haga clic en Aceptar para crear el objeto de vídeo.
5. Arrastre el objeto de vídeo desde el panel Biblioteca hasta el escenario para crear una instancia del mismo.
6. Con el objeto de vídeo seleccionado en el escenario, introduzca **my_video** en el cuadro de texto Nombre de instancia del inspector de propiedades (Ventana > Propiedades > Propiedades).
7. Con la instancia de vídeo aún seleccionada, introduzca **320** en el cuadro de texto de anchura y **213** en el cuadro de texto de altura.
8. Seleccione el fotograma 1 en la línea de tiempo y abra el panel Acciones (Ventana > Acciones).
9. Introduzca el código siguiente en el panel Acciones:

```
// Crear un objeto NetConnection.
var netConn:NetConnection = new NetConnection();
// Crear una conexión de transmisión local.
netConn.connect(null);
// Crear un objeto NetStream y definir una función onStatus().
var nStream:NetStream = new NetStream(netConn);
// Asociar la salida de vídeo NetStream al objeto Video.
my_video.attachVideo(nStream);
// Establecer el tiempo de búfer.
nStream.setBufferTime(30);
// Reproduciendo el archivo FLV.
nStream.play("http://www.helpexamples.com/flash/video/
  lights_short.flv");
// Trazar los metadatos.
nStream.onMetaData = function(myMeta) {
     for (var i in myMeta) {
          trace(i + ":\t" + myMeta[i])
     }
};
```

**10.** Seleccione Control > Probar película para probar el código.

Se ve la información siguiente en el panel Salida:

```
canSeekToEnd:true
audiocodecid:2
audiodelay:0.038
audiodatarate:96
videocodecid:4
framerate:15
videodatarate:400
height:213
width:320
duration:8.04
```

**NOTA** Si el vídeo no tiene sonido, la información de metadatos relativa al audio (como `audiodatarate`) devuelve `undefined` ya que no se ha añadido información de audio al metadato durante la codificación.

También puede utilizar el siguiente formato para mostrar la mayora parte de la información de metadatos. Por ejemplo, el siguiente código muestra la duración de un archivo FLV:

```
nStream.onMetaData = function(myMeta) {
  trace("FLV duration: " + myMeta.duration + " sec.");
};
```

Este formato no puede trazar la información de metadatos de `cuePoint`. Para más información sobre el trazado de cuepoints, consulte "Trazado de cuepoints de un archivo FLV" en la página 649.

## Configuración de archivos FLV para alojar en el servidor

Al trabajar con archivos FLV, puede que tenga que configurar el servidor para que funcione con el formato de archivo FLV. MIME (Multipurpose Internet Mail Extensions) es una especificación de datos estandarizada que permite enviar archivos que no son ASCII a través de conexiones de Internet. Los navegadores Web y los clientes de correo electrónico se configuran para interpretar numerosos *tipos MIME* de modo que puedan enviar y recibir vídeo, audio, gráficos y texto con formato. Para cargar archivos FLV de un servidor Web, puede que necesite registrar la extensión de archivo y el tipo MIME en el servidor Web, por lo que debería comprobar la documentación del servidor Web. El tipo MIME de los archivos FLV es `video/x-flv`. La información completa del tipo de archivo FLV es la siguiente:

Tipo Mime: video/x-flv

Extensión de archivo: .flv

Parámetros necesarios: ninguno

Parámetros opcionales: ninguno

Consideraciones de codificación: los archivos FLV son archivos binarios, algunas aplicaciones podrían necesitar que se establezca el subtipo application/octet-stream.

Problemas de seguridad: ninguno

Especificación publicada: www.macromedia.com/go/flashfileformat.

Microsoft ha cambiado la forma en la que se controlan los flujos de medios en el servidor Web Servicios de Microsoft Internet Information Server (IIS) 6.0 desde las versiones anteriores. Las versiones anteriores de IIS no necesitan modificaciones para reproducir Flash Video. En IIS 6.0, el servidor Web predeterminado que incluye Windows 2003, el servidor necesita un tipo MIME para reconocer que los archivos FLV son medios de reproducción.

Cuando archivos SWF que reproducen archivos FLV externos se sitúan en un servidor Microsoft Windows 2003 y se ven en un navegador, el archivo SWF se reproduce sin problemas, pero no el vídeo FLV. Este problema afecta a todos los archivos FLV ubicados en un servidor Windows 2003, incluidos los archivos que se crean con versiones anteriores de la herramienta de edición de Flash, Macromedia Flash Video Kit para Dreamweaver MX 2004. Estos archivos funcionan correctamente si los prueba en otros sistemas operativos.

Para obtener información sobre la configuración de Microsoft Windows 2003 y Microsoft IIS Server 6.0 para reproducir vídeo FLV, consulte www.macromedia.com/go/tn_19439.

## Utilización de archivos FLV locales en Macintosh

Si intenta reproducir un archivo FLV local desde una unidad que no sea del sistema en un equipo Macintosh mediante una ruta que utilice una barra relativa (/), el vídeo no se reproducirá. Las *unidades que no son del sistema* incluyen, aunque no se limitan a, CD-ROM, discos duros con particiones, medios de almacenamiento extraíbles y dispositivos de almacenamiento conectados.

| NOTA | El motivo de este fallo es una limitación del sistema operativo, no de Flash ni de Flash Player |
|---|---|

Para que un archivo FLV se reproduzca en una unidad que no sea del sistema en Macintosh, haga referencia a ella con una ruta absoluta mediante la notación de dos puntos (:) en lugar de la basada en barras (/). La lista siguiente muestra la diferencia de los dos tipos de notación:

**Notación basada en barras** miUnidad/miCarpeta/miFLV.flv

**Notación basada en dos puntos** (Macintosh)miUnidad:miCarpeta:miFLV.flv

También puede crear un archivo de proyecto para un CD-ROM que va a utilizar para la reproducción en Macintosh. Para obtener la información más actualizada sobre CD-ROM y archivos FLV de Macintosh, consulte www.macromedia.com/go/3121b301.

# Creación de animaciones progresivas para archivos multimedia

ActionScript proporciona varias maneras de precargar o hacer un seguimiento del progreso de la descarga de archivos multimedia externos. Puede crear animaciones o barras de progreso para mostrar visualmente el transcurso de la carga o la cantidad de contenido que se ha cargado.

Para precargar archivos SWF y JPEG, utilice la clase MovieClipLoader, la cual proporciona un mecanismo detector de eventos para comprobar el progreso de la descarga. Para más información, consulte "Precarga de archivos SWF y JPEG" en la página 427.

Para realizar un seguimiento del progreso de la descarga de los archivos MP3, utilice los métodos `Sound.getBytesLoaded()` y `Sound.getBytesTotal()`; para realizar un seguimiento del progreso de la descarga de los archivos FLV, use las propiedades `NetStream.bytesLoaded` y `NetStream.bytesTotal`. Para más información, consulte "Precarga de archivos MP3" en la página 420.

Para obtener información sobre la creación de barras de progreso para cargar archivos multimedia, consulte los temas siguientes:

- "Creación de una animación de progreso para cargar archivos SWF y de imagen" en la página 663
- "Creación de una barra de progreso para cargar archivos MP3 con ActionScript" en la página 665
- "Creación de una barra de progreso para cargar archivos FLV con ActionScript" en la página 667

Puede encontrar un archivo de origen de muestra que utiliza animación con scripts para crear una animación de la barra de progreso. Busque tweenProgress.fla en la carpeta Samples del disco duro:

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Tween ProgressBar.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Tween ProgressBar.

# Creación de una animación de progreso para cargar archivos SWF y de imagen

Al cargar archivos SWF o de imagen de gran tamaño en una aplicación, podría desear crear una animación que muestre el progreso de la carga. Podría crear una barra de progreso que mostrara incrementos a medida que se cargara la animación. También podría crear una animación que cambiase mientras se carga el archivo. Para obtener información sobre cómo cargar archivos SWF y de imagen, consulte "Carga de archivos de imagen y SWF externos" en la página 629.

El ejemplo siguiente muestra cómo utilizar la clase MovieClipLoader y la API de dibujo para mostrar el progreso de carga de un archivo de imagen.

### Para crear una barra de progreso para cargar los archivos de imagen o SWF:

1. Cree un nuevo documento de Flash denominado **loadImage.fla**.
2. Seleccione Modificar > Documento e introduzca **700** en el cuadro de texto de anchura y **500** en el cuadro de texto de altura para cambiar las dimensiones del documento.
3. Seleccione el fotograma 1 de la línea de tiempo y escriba el siguiente código en el panel Acciones:

```
//crear clips para incluir el contenido
this.createEmptyMovieClip("progressBar_mc", 0);
progressBar_mc.createEmptyMovieClip("bar_mc", 1);
progressBar_mc.createEmptyMovieClip("stroke_mc", 2);
//utilizar métodos de dibujo para crear una barra de progreso
with (progressBar_mc.stroke_mc) {
   lineStyle(0, 0x000000);
   moveTo(0, 0);
   lineTo(100, 0);
   lineTo(100, 10);
   lineTo(0, 10);
   lineTo(0, 0);
}
with (progressBar_mc.bar_mc) {
   beginFill(0xFF0000, 100);
   moveTo(0, 0);
   lineTo(100, 0);
   lineTo(100, 10);
   lineTo(0, 10);
   lineTo(0, 0);
   endFill();
   _xscale = 0;
}
progressBar_mc._x = 2;
progressBar_mc._y = 2;
// cargar progreso
```

```
var mclListener:Object = new Object();
mclListener.onLoadStart = function(target_mc:MovieClip) {
   progressBar_mc.bar_mc._xscale = 0;
};
mclListener.onLoadProgress = function(target_mc:MovieClip,
  bytesLoaded:Number, bytesTotal:Number) {
   progressBar_mc.bar_mc._xscale = Math.round(bytesLoaded/
  bytesTotal*100);
};
mclListener.onLoadComplete = function(target_mc:MovieClip) {
   progressBar_mc.removeMovieClip();
};
mclListener.onLoadInit = function(target_mc:MovieClip) {
   target_mc._height = 500;
   target_mc._width = 700;
};
//Crear un clip para incluir la imagen.
this.createEmptyMovieClip("image_mc", 100);
var image_mcl:MovieClipLoader = new MovieClipLoader();
image_mcl.addListener(mclListener);
/* Cargar la imagen en el clip.
You can change the following URL to a SWF or another image file. */
image_mcl.loadClip("http://www.helpexamples.com/flash/images/gallery1/
  images/pic3.jpg", image_mc);
```

4. Seleccione Control > Probar película para ver la carga de imagen y observar la barra de progreso.

| NOTA | Si prueba este código una segunda vez, la imagen se almacenará en caché y la barra de progreso finalizará al instante. Para probar varias veces, utilice distintas imágenes y cárguelas desde una fuente externa. Una fuente local podría causar problemas al probar la aplicación porque el contenido se carga demasiado rápido. |
|---|---|

Puede encontrar un archivo de origen de muestra que utiliza animación con scripts para crear una animación de la barra de progreso. Busque tweenProgress.fla en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Tween ProgressBar.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Tween ProgressBar.

También encontrará muestras de aplicaciones de galerías de fotos. Estos archivos proporcionan ejemplos de cómo utilizar ActionScript para controlar dinámicamente clips de película mientras se cargan archivos de imagen en un archivo SWF. Los archivos de origen de muestra gallery_tree.fla y gallery_tween.fla se pueden encontrar en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Galleries.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Galleries.

## Creación de una barra de progreso para cargar archivos MP3 con ActionScript

El siguiente ejemplo carga varias canciones en un archivo SWF. Una barra de progreso, creada en la interfaz API de dibujo, muestra cómo avanza la carga. Cuando comienza y finaliza la carga de la música, aparece la información correspondiente en el panel Salida. Para obtener información sobre la carga de archivos MP3, consulte "Carga de un archivo MP3" en la página 635.

### Para crear una barra de progreso para la carga de archivos MP3:

1. Cree un nuevo documento de Flash denominado **loadSound.fla**.
2. Seleccione el fotograma 1 de la línea de tiempo y escriba el siguiente código en el panel Acciones.

```
var pb_height:Number = 10;
var pb_width:Number = 100;
var pb:MovieClip = this.createEmptyMovieClip("progressBar_mc",
  this.getNextHighestDepth());
pb.createEmptyMovieClip("bar_mc", pb.getNextHighestDepth());
pb.createEmptyMovieClip("vBar_mc", pb.getNextHighestDepth());
pb.createEmptyMovieClip("stroke_mc", pb.getNextHighestDepth());
pb.createTextField("pos_txt", pb.getNextHighestDepth(), 0, pb_height,
  pb_width, 22);

pb._x = 100;
pb._y = 100;
```

```
with (pb.bar_mc) {
   beginFill(0x00FF00);
   moveTo(0, 0);
   lineTo(pb_width, 0);
   lineTo(pb_width, pb_height);
   lineTo(0, pb_height);
   lineTo(0, 0);
   endFill();
   _xscale = 0;
}
with (pb.vBar_mc) {
   lineStyle(1, 0x000000);
   moveTo(0, 0);
   lineTo(0, pb_height);
}
with (pb.stroke_mc) {
   lineStyle(3, 0x000000);
   moveTo(0, 0);
   lineTo(pb_width, 0);
   lineTo(pb_width, pb_height);
   lineTo(0, pb_height);
   lineTo(0, 0);
}

var my_interval:Number;
var my_sound:Sound = new Sound();
my_sound.onLoad = function(success:Boolean) {
   if (success) {
      trace("sound loaded");
   }
};
my_sound.onSoundComplete = function() {
   clearInterval(my_interval);
   trace("Cleared interval");
}
my_sound.loadSound("http://www.helpexamples.com/flash/sound/song2.mp3",
  true);
my_interval = setInterval(updateProgressBar, 100, my_sound);

function updateProgressBar(the_sound:Sound):Void {
   var pos:Number = Math.round(the_sound.position / the_sound.duration *
  100);
   pb.bar_mc._xscale = pos;
   pb.vBar_mc._x = pb.bar_mc._width;
   pb.pos_txt.text = pos + "%";
}
```

3. Seleccione Control > Probar película para cargar el archivo MP3 y observar la barra de progreso.

| NOTA | Si prueba este código una segunda vez, la imagen se almacenará en caché y la barra de progreso finalizará al instante. Para probar varias veces, utilice distintas imágenes y cárguelas desde una fuente externa. Una fuente local podría causar problemas al probar la aplicación porque el contenido se carga demasiado rápido. |
|---|---|

Para más información sobre el uso de sonido, consulte la entrada de clase Sound, `%{Sound}%`, en *Referencia del lenguaje ActionScript 2.0*.

Puede encontrar un archivo de origen de muestra que utiliza animación con scripts para crear una animación de la barra de progreso. Busque tweenProgress.fla en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Tween ProgressBar.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Tween ProgressBar.

También encontrará un archivo de origen de muestra que carga archivos MP3, jukebox.fla, en la carpeta Samples del disco duro. Este ejemplo muestra cómo crear un jukebox mediante el uso de tipos de datos, principios generales de programación y varios componentes.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\Components\Jukebox.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/Components/Jukebox.

## Creación de una barra de progreso para cargar archivos FLV con ActionScript

Puede crear una barra de progreso para mostrar el progreso de carga de un archivo FLV. Para obtener información sobre la carga de archivos FLV en un archivo SWF, consulte "Precarga de archivos FLV" en la página 647. Para obtener otro tipo de información sobre los archivos FLV y Flash, consulte "Utilización de vídeo FLV" en la página 640.

El ejemplo siguiente utiliza la API de dibujo para crear una barra de progreso. El ejemplo también utiliza las propiedades `bytesLoaded` y `bytesTotal` que muestran el progreso de carga de video1.flv en la instancia de objeto de vídeo denominada `my_video`. Asimismo se crea de forma dinámica un campo de texto llamado `loaded_txt` para ver información sobre el proceso de carga.

### Para crear una barra de progreso que muestre el progreso de carga:

1. Cree un nuevo archivo FLA denominado **flvProgress.fla**.
2. En el panel Biblioteca (Ventana > Biblioteca), seleccione Nuevo Vídeo en el menú emergente Biblioteca.
3. En el cuadro de diálogo Propiedades de vídeo, asigne un nombre al símbolo de vídeo y seleccione Vídeo (controlado por ActionScript).
4. Haga clic en Aceptar para crear el objeto de vídeo.
5. Arrastre el objeto de vídeo desde el panel Biblioteca hasta el escenario para crear una instancia del mismo.
6. Con el objeto de vídeo seleccionado en el escenario, introduzca **my_video** en el cuadro de texto Nombre de instancia del inspector de propiedades (Ventana > Propiedades > Propiedades).
7. Con la instancia de vídeo aún seleccionada, introduzca **320** en el cuadro de texto de anchura y **213** en el cuadro de texto de altura.
8. Seleccione el fotograma 1 de la línea de tiempo y escriba el siguiente código en el panel Acciones:

```
var connection_nc:NetConnection = new NetConnection();
connection_nc.connect(null);
var stream_ns:NetStream = new NetStream(connection_nc);
my_video.attachVideo(stream_ns);
stream_ns.play("http://www.helpexamples.com/flash/video/
  typing_short.flv");

this.createTextField("loaded_txt", this.getNextHighestDepth(), 10, 10,
  160, 22);
this.createEmptyMovieClip("progressBar_mc", this.getNextHighestDepth());
progressBar_mc.createEmptyMovieClip("bar_mc",
  progressBar_mc.getNextHighestDepth());
with (progressBar_mc.bar_mc) {
   beginFill(0xFF0000);
   moveTo(0, 0);
   lineTo(100, 0);
   lineTo(100, 10);
   lineTo(0, 10);
   lineTo(0, 0);
   endFill();
   _xscale = 0;
}
```

```
progressBar_mc.createEmptyMovieClip("stroke_mc",
  progressBar_mc.getNextHighestDepth());
with (progressBar_mc.stroke_mc) {
   lineStyle(0, 0x000000);
   moveTo(0, 0);
   lineTo(100, 0);
   lineTo(100, 10);
   lineTo(0, 10);
   lineTo(0, 0);
}

var loaded_interval:Number = setInterval(checkBytesLoaded, 500,
  stream_ns);
function checkBytesLoaded(my_ns:NetStream) {
   var pctLoaded:Number = Math.round(my_ns.bytesLoaded /
  my_ns.bytesTotal * 100);
   loaded_txt.text = Math.round(my_ns.bytesLoaded / 1000) + " of " +
  Math.round(my_ns.bytesTotal / 1000) + " KB loaded (" + pctLoaded +
  "%)";
   progressBar_mc.bar_mc._xscale = pctLoaded;
   if (pctLoaded>=100) {
      clearInterval(loaded_interval);
   }
}
```

9. Seleccione Control > Probar película para probar el código.

   El vídeo se carga y una barra de animación y valores de texto que cambian comunican el progreso de carga. Si estos elementos se superponen con el vídeo, desplace el objeto de vídeo en el escenario. Puede personalizar el color de la barra de progreso mediante la modificación de `beginFill` y `lineStyle` en el fragmento de código anterior.

| NOTA | Si la barra de progreso se carga de forma instantánea, el vídeo se ha almacenado en la caché del disco duro (al probar ya este ejemplo o al cargarlo mediante otro procedimiento). Si ocurre esto, cargue un archivo FLV en el servidor y cárguelo en su lugar. |
|---|---|

Puede encontrar un archivo de origen de muestra que utiliza animación con scripts para crear una animación de la barra de progreso. Busque tweenProgress.fla en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Tween ProgressBar.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Tween ProgressBar.

CAPÍTULO 16

# Trabajo con datos externos

16

En Macromedia Flash Basic 8 y Macromedia Flash Professional 8, puede utilizar ActionScript para cargar datos desde fuentes externas a un archivo SWF. También puede enviar datos, que puede proporcionar el usuario o el servidor, desde un archivo SWF a un servidor de aplicaciones (por ejemplo, Macromedia ColdFusion o Macromedia JRun) u otro tipo de script de servidor, por ejemplo PHP o Perl. Macromedia Flash Player puede enviar y cargar datos a través de HTTP o HTTPS o cargarlos desde un archivo de texto local. También puede crear conexiones de socket TCP/IP persistentes para aplicaciones que requieren una latencia baja, como por ejemplo las aplicaciones de chat o los servicios de cotizaciones bursátiles. Una novedad de Flash Player 8 es la posibilidad de cargar archivos desde el equipo del usuario a un servidor y descargarlos desde un servidor al equipo del usuario.

A los datos que se cargan en un archivo SWF o se envían desde éste se les puede aplicar el formato XML (Lenguaje extensible de marcado, Extensible Markup Language) o el de pares de nombre/valor.

Flash Player también puede enviar datos a su entorno host o recibir datos de éste (por ejemplo un navegador Web) o de otra instancia de Flash Player del mismo equipo o página Web.

De manera predeterminada, un archivo SWF puede acceder sólo a datos que residan exactamente en el mismo dominio (por ejemplo, www.macromedia.com). (Para más información, consulte "Dominios, seguridad entre dominios y archivos SWF" en la página 735.)

Para más información sobre el trabajo con datos externos, consulte los temas siguientes:

# Envío y carga de variables

Un archivo SWF es una ventana que permite capturar y mostrar información, del mismo modo que en una página HTML. Sin embargo, los archivos SWF pueden permanecer cargados en el navegador y actualizarse continuamente con nueva información sin necesidad de volver a cargar toda la página. Las funciones y los métodos de ActionScript permiten enviar y recibir información de scripts de servidor, así como recibir información de archivos de texto y archivos XML.

Además, los scripts de servidor pueden solicitar información específica de una base de datos y enviarla a un archivo SWF. Los scripts de servidor pueden estar escritos en lenguajes diferentes: algunos de los más comunes son CFML, Perl, ASP (Páginas de Active Server de Microsoft, Microsoft Active Server Pages) y PHP. El hecho de almacenar información en una base de datos para después recuperarla permite crear contenido dinámico y personalizado para el archivo SWF. Por ejemplo, puede crear un tablero de mensajes, perfiles personales de los usuarios o una cesta de la compra que efectúe un seguimiento de las compras de un usuario.

Existen varias funciones y métodos de ActionScript que permiten pasar información hacia y desde un archivo SWF. Cada función y método utiliza un protocolo para transferir información y requiere que la información tenga un formato específico.

- Las funciones y los métodos MovieClip que utilizan el protocolo HTTP o HTTPS para enviar información en formato URL codificado son `getURL()`, `loadVariables()`, `loadVariablesNum()`, `loadMovie()` y `loadMovieNum()`.
- Los métodos LoadVars que utilizan el protocolo HTTP o HTTPS para enviar y cargar información en formato URL codificado son `load()`, `send()` y `sendAndLoad()`.
- Los métodos que utilizan el protocolo HTTP o HTTPS para enviar y cargar información como XML son `XML.send()`, `XML.load()` y `XML.sendAndLoad()`.
- Los métodos que crean y utilizan una conexión de socket TCP/IP para enviar y cargar información como XML son `XMLSocket.connect()` y `XMLSocket.send()`.

Para más información, consulte los siguientes temas:


- "Comprobación de los datos cargados" en la página 673
- "Creación de una barra de progreso para mostrar el progreso de la carga de datos" en la página 674

## Comprobación de los datos cargados

Cada función o método que carga datos en un archivo SWF (excepto `XMLSocket.send()`) es *asíncrono*: los resultados de una acción se devuelven en un tiempo indeterminado.

Para poder utilizar los datos cargados en un archivo SWF, primero debe comprobar que se han cargado. Por ejemplo, no puede cargar variables y manipular sus valores en el mismo script, ya que los datos para manipular no existen en el archivo hasta que se carga. En el siguiente script, no puede utilizar la variable `lastSiteVisited` hasta que esté seguro de que ésta se ha cargado desde el archivo myData.txt. En el archivo myData.txt, habría texto similar al que se muestra en el ejemplo siguiente:

```
lastSiteVisited=www.macromedia.com
```

Si hubiera utilizado el siguiente código, no podría realizar el seguimiento de los datos que se están cargando:

```
loadVariables("myData.txt", 0);
trace(lastSiteVisited); // undefined
```

Cada función o método tiene una técnica específica que puede utilizarse para comprobar los datos que han cargado. Si utiliza %{loadVariables function}% o %{loadMovie function}%, puede cargar información en un clip de película de destino y utilizar el controlador `onData` para ejecutar un script. Si utiliza %{loadVariables function}% para cargar los datos, el controlador `onData` se ejecutará al cargarse la última variable. Si usa %{loadMovie function}% para cargar los datos, el controlador `onData` se ejecutará cada vez que se envíe un fragmento del archivo SWF a Flash Player.

Por ejemplo, el siguiente código de ActionScript carga las variables del archivo myData.txt en el clip de película `loadTarget_mc`. Un controlador `onData()` asignado a la instancia `loadTarget_mc` utiliza la variable `lastSiteVisited`, que se carga desde el archivo myData.txt. Las siguientes acciones de trazado sólo aparecen cuando se han cargado todas las variables, incluida `lastSiteVisited`:

```
this.createEmptyMovieClip("loadTarget_mc", this.getNextHighestDepth());
this.loadTarget_mc.onData = function() {
  trace("Data Loaded");
  trace(this.lastSiteVisited);
};
loadVariables("myData.txt", this.loadTarget_mc);
```

Si utiliza los métodos `XML.load()`, `XML.sendAndLoad()` y `XMLSocket.connect()`, deberá definir un controlador que procese los datos cuando lleguen. Este controlador es una propiedad del objeto XML o XMLSocket a la que se le asigna una función definida por el usuario. Se llama automáticamente a los controladores cuando se recibe la información. Para el objeto XML, utilice `XML.onLoad()` o `XML.onData()`. Para el objeto XMLSocket, utilice `XMLSocket.onConnect()`.

Para más información, consulte "Utilización de la clase XML" en la página 692 y "Utilización de la clase XMLSocket" en la página 698. Para más información sobre la utilización de LoadVars para el envío o la carga de datos que pueden procesarse tras la recepción de éstos, consulte "Utilización de la clase LoadVars" en la página 678.

## Creación de una barra de progreso para mostrar el progreso de la carga de datos

En el siguiente ejercicio se crea dinámicamente un precargador simple mediante la interfaz de programación de aplicaciones (API) de dibujo y se muestra el progreso de la carga para un documento XML.

| SUGERENCIA | Si el archivo XML remoto se carga demasiado rápido para poder ver el efecto de la precarga, intente cargar un archivo XML de mayor tamaño en Internet y cargar, a continuación, dicho archivo. |
|---|---|

### Para crear una barra de progreso mediante la interfaz API de dibujo:

1. Cree un nuevo documento de Flash y guárdelo como **drawapi.fla**.
2. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
var barWidth:Number = 200;
var barHeight:Number = 6;

this.createEmptyMovieClip("pBar_mc", 9999);
var bar:MovieClip = pBar_mc.createEmptyMovieClip("bar_mc", 10);
bar.beginFill(0xFF0000, 100);
bar.moveTo(0, 0);
bar.lineTo(barWidth, 0);
bar.lineTo(barWidth, barHeight);
bar.lineTo(0, barHeight);
bar.lineTo(0, 0);
bar.endFill();
bar._xscale = 0;

var stroke:MovieClip = pBar_mc.createEmptyMovieClip("stroke_mc", 20);
stroke.lineStyle(0, 0x000000);
stroke.moveTo(0, 0);
stroke.lineTo(barWidth, 0);
stroke.lineTo(barWidth, barHeight);
stroke.lineTo(0, barHeight);
stroke.lineTo(0, 0);
```

```
pBar_mc.createTextField("label_txt", 30, 0, barHeight, 100, 21);
pBar_mc.label_txt.autoSize = "left";
pBar_mc.label_txt.selectable = false;

pBar_mc._x = (Stage.width - pBar_mc._width) / 2;
pBar_mc._y = (Stage.height - pBar_mc._height) / 2;

var my_xml:XML = new XML();
my_xml.ignoreWhite = true;
my_xml.onLoad = function(success:Boolean) {
  pBar_mc.onEnterFrame = undefined;
  if (success) {
    trace("XML cargado correctamente");
  } else {
    trace("No se puede cargar XML");
  }
};
my_xml.load("http://www.helpexamples.com/flash/xml/ds.xml");

pBar_mc.onEnterFrame = function() {
  var pctLoaded:Number = Math.floor(my_xml.getBytesLoaded() /
  my_xml.getBytesTotal() * 100);
  if (!isNaN(pctLoaded)) {
    pBar_mc.bar_mc._xscale = pctLoaded;
    pBar_mc.label_txt.text = pctLoaded + "% loaded";
    if (pctLoaded >= 100) {
      pBar_mc.onEnterFrame = undefined;
    }
  }
};
```

El código anterior se divide en siete secciones. En la primera sección se definen la anchura y altura de la barra de progreso cuando ésta se dibuja en el escenario. La barra de progreso se centrará en el escenario en una sección posterior. En la siguiente sección de código se crean dos clips de película:`pBar_mc` y `bar_mc`. El clip de película `bar_mc` está anidado dentro de `pBar_mc` y dibuja un rectángulo rojo en el escenario. La instancia `bar_mc` modifica la propiedad `_xscale` cuando el archivo XML se carga desde el sitio Web remoto.

A continuación, se anida un segundo clip de película dentro del clip de película `pBar_mc`: `stroke_mc`. El clip de película `stroke_mc` dibuja un contorno en el escenario que se ajusta a las dimensiones especificadas por las variables `barHeight` y `barWidth` definidas en la primera sección. En la cuarta sección de código se crea, dentro del clip de película `pBar_mc`, un campo de texto que se utiliza para mostrar el porcentaje del archivo XML que ya se ha cargado, similar a la etiqueta del componente ProgressBar. A continuación, el clip de película `pBar_mc` (que contiene las instancias anidadas `bar_mc`, `stroke_mc` y `label_txt`) se centra en el escenario.

En la sexta sección de código se define una nueva instancia de objeto XML, que se emplea para cargar un archivo XML externo. Se define un controlador de eventos onLoad, que traza un mensaje en el panel Salida. El controlador de eventos onLoad también elimina el controlador de eventos `onEnterFrame` (que se define en la siguiente sección) para el clip de película `pBar_mc`. En la última sección de código se define un controlador de eventos `onEnterFrame` para el clip de película `pBar_mc`. Este controlador de eventos supervisa la cantidad de archivo XML externo que se ha cargado y modifica la propiedad `_xscale` para el clip de película `bar_mc`. En primer lugar, el controlador de eventos `onEnterFrame` calcula qué porcentaje del archivo se ha descargado. Siempre que el porcentaje del archivo cargado sea un número válido, se establece la propiedad `_xscale` para `bar_mc` y en el campo de texto dentro de `pBar_mc` se muestra dicho porcentaje. Cuando se completa la carga del archivo, (el porcentaje cargado alcanza el 100%), se elimina el controlador de eventos `onEnterFrame` y deja de supervisarse el progreso de descarga.

3. Seleccione Control > Probar película para probar el documento de Flash.

   A medida que se carga el archivo XML externo, el clip de película anidado `bar_mc` cambia de tamaño para mostrar el progreso de la descarga. Una vez completada la carga, se elimina el controlador de eventos `onEnterFrame` a fin de que no continúe calculando el progreso de la descarga. Según la velocidad con la que se complete la descarga, podrá ver cómo crece lentamente la barra hasta que `bar_mc` alcanza la misma anchura que el clip de película `stroke_mc`. Si la descarga se realiza demasiado rápido, es posible que la barra de progreso pase de 0% a 100% demasiado pronto y resulte más difícil ver el efecto; en este caso, puede que sea necesario intentar descargar un archivo XML de mayor tamaño.

# Utilización del protocolo HTTP para conectar con scripts de servidor

Las funciones %{loadVariables function}%, %{loadVariablesNum function}%, %{getURL function}%, %{loadMovie function}%, %{loadMovieNum function}% y los métodos %{loadVariables (método MovieClip.loadVariables)}%, %{loadMovie (método MovieClip.loadMovie)}% y %{getURL (método MovieClip.getURL)}% pueden comunicarse con scripts de servidor a través de los protocolos HTTP o HTTPS.Estos métodos y funciones envían todas las variables desde la línea de tiempo asociada a la función. Cuando se utilizan como métodos del objeto MovieClip, `loadVariables()`, `getURL()` y `loadMovie()` envían todas las variables del clip de película especificado; cada función (o método) gestiona su respuesta del modo siguiente:

- La función `getURL()` devuelve la información a una ventana del navegador, no a Flash Player.
- El método `loadVariables()` carga variables en una línea de tiempo o un nivel especificado de Flash Player.

- El método `loadMovie()` carga un archivo SWF en un nivel o un clip de película especificado de Flash Player.

Cuando utilice `loadVariables()`, `getURL()` o `loadMovie()`, puede especificar varios parámetros:

- *`URL`* es el archivo en el que residen las variables remotas.
- *`Ubicación`* es el nivel o destino del archivo SWF que recibe las variables (La función `getURL()` no toma este parámetro.)

  Para más información sobre niveles y destinos, consulte el Capítulo 1, "Varias líneas de tiempo y niveles" en *Utilización de Flash*.
- *`Variables`* establece el método HTTP, ya sea `GET` (que añade las variables al final del URL) o `POST` (que envía las variables en un encabezado HTTP independiente), mediante el cual se envían las variables. Si se omite este parámetro, Flash Player se establece de forma predeterminada como `GET`, pero no se envía ninguna variable.

Por ejemplo, si desea realizar el seguimiento de las puntuaciones más altas de un juego, puede almacenarlas en un servidor y utilizar `loadVariables()` para cargarlas en el archivo SWF cada vez que alguien juegue una partida. La llamada de función puede ser similar a la del siguiente ejemplo:

```
this.createEmptyMovieClip("highscore_mc", 10);
loadVariables("http://www.helpexamples.com/flash/highscore.php",
  highscore_mc, "GET");
```

Este ejemplo carga las variables del script ColdFusion llamado high_score.cfm en la instancia del clip de película `scoreClip` mediante el método HTTP `GET`.

Cualquiera de las variables cargadas con la función `loadVariables()` debe tener el formato MIME estándar *application/x-www-form-urlencoded* (formato estándar que se utiliza en los scripts CFM y CGI). El archivo que especifique en el parámetro *`URL`* de la función `loadVariables()` debe escribir los pares de variable y valor en este formato para que Flash pueda leerlos. Este archivo puede especificar cualquier número de variables; los pares de variable y valor se deben separar con un ampersand (`&`) y las palabras dentro de un valor deben hacerlo con un signo más (+). Por ejemplo, la siguiente frase define varias variables:

```
highScore1=54000&playerName1=RGoulet&highScore2=53455&playerName2=
  WNewton&highScore3=42885&playerName3=TJones
```

**NOTA** Es posible que tenga que codificar como URL determinados caracteres, como el signo más (+) o el ampersand (&). Para más información, consulte www.macromedia.com/go/tn_14143.

Para más información, consulte el tema siguiente: "Utilización de la clase LoadVars" en la página 678. Consulte también %{loadVariables function}%, %{getURL function}%, %{loadMovie function}% y la entrada de %{LoadVars}% en *Referencia del lenguaje ActionScript 2.0*.

# Utilización de la clase LoadVars

Si está publicando en Flash Player 6 o posterior y desea una mayor flexibilidad de la que ofrece `loadVariables()`, puede utilizar la clase LoadVars para transferir variables entre un archivo SWF y un servidor.

La clase LoadVars se introdujo en Flash Player 6 para proporcionar una interfaz más limpia y orientada a objetos para tarea común de intercambiar datos CGI con un servidor Web. Entre las ventajas que ofrece la clase LoadVars, figuran las siguientes:

- No es necesario que cree clips de película de contenedor para dar cabida a datos o ordenar clips de película con variables específicas para la comunicación cliente/servidor.
- La interfaz de clase es similar a la del objeto XML, lo que proporciona cierta coherencia en ActionScript. Utiliza los métodos `load()`, `send()` y `sendAndLoad()` para iniciar la comunicación con un servidor. La principal diferencia entre las clases LoadVars y XML consiste en que los datos de LoadVars son una propiedad del objeto LoadVars, en lugar de serlo de un árbol XML DOM (Modelo de Objetos de Documento, Document Object Model) almacenado en el objeto XML.
- La interfaz de clase es más simple (ya que incluye métodos denominados `load`, `send`, `sendAndLoad`, que la antigua interfaz loadVariables.
- Puede obtener información adicional sobre la comunicación a través de los métodos `getBytesLoaded` y `getBytesTotal`.
- Puede obtener información sobre el progreso de la descarga de datos (aunque no puede acceder a los datos hasta que se han descargado completamente).
- La interfaz de callback se logra a través de métodos de ActionScript (`onLoad`) en lugar de a través del enfoque ya obsoleto y desfasado `onClipEvent` (data) que exigía loadVariables.
- Se producen notificaciones de errores.
- Puede añadir encabezados de solicitudes HTTP personalizadas.

Para llamar a los métodos del objeto LoadVars, debe crearlo primero. Este objeto es un contenedor que almacena los datos cargados.

El siguiente procedimiento muestra cómo utilizar ColdFusion y la clase LoadVars para enviar un mensaje de correo electrónico desde un archivo SWF.

| NOTA | Debe tener ColdFusion instalado en su servidor Web para realizar este ejemplo. |
|---|---|

### Para cargar datos con el objeto LoadVars:

1. Cree un archivo CFM en Macromedia Dreamweaver o en su editor de texto preferido. Añada el siguiente texto al archivo:

```
<cfif StructKeyExists(Form, "emailTo")>
<cfmail to="#Form.emailTo#" from="#Form.emailFrom#"
  subject="#Form.emailSubject#">#Form.emailBody#</cfmail>
&result=true
<cfelse>
&result=false
</cfif>
```

2. Guarde el archivo como **email.cfm** y cárguelo en el sitio Web.
3. En Flash, cree un nuevo documento.
4. Cree cuatro campos de entrada de texto en el escenario y asígneles los siguientes nombres de instancias: **emailFrom_txt**, **emailTo_txt**, **emailSubject_txt** y **emailBody_txt**.
5. Cree un campo de texto dinámico en el escenario con el nombre de instancia **debug_txt**.
6. Cree un símbolo de botón, arrastre una instancia del mismo hasta el escenario y asígnele el nombre de **submit_btn**.
7. Seleccione el fotograma 1 en la línea de tiempo y abra el panel Acciones (Ventana > Acciones) si todavía no está abierto.
8. Introduzca los códigos siguientes en el panel Acciones:

```
this.submit_btn.onRelease = function() {
  var emailResponse:LoadVars = new LoadVars();
  emailResponse.onLoad = function(success:Boolean) {
  if (success){
       debug_txt.text = this.result;
  } else {
       debug_txt.text = "error downloading content";
  }
  };
  var email:LoadVars = new LoadVars();
  email.emailFrom = emailFrom_txt.text;
  email.emailTo = emailTo_txt.text;
  email.emailSubject = emailSubject_txt.text;
  email.emailBody = emailBody_txt.text;
  email.sendAndLoad("http://www.yoursite.com/email.cfm", emailResponse,
  "POST");
};
```

Este código ActionScript crea una nueva instancia de objeto LoadVars, copia los valores de los campos de texto en la instancia y luego envía los datos al servidor. El archivo CFM envía el correo electrónico y devuelve una variable (`true` o `false`) al archivo SWF denominado `result`, que aparece en el campo de texto `debug_txt`.

| NOTA | No olvide cambiar el URL www.yoursite.com para que refleje el de su propio dominio. |
|---|---|

9. Guarde el documento como **sendEmail.fla** y seleccione Archivo > Publicar para publicarlo.
10. Cargue sendEmail.swf en el mismo directorio en el que se encuentra email.cfm (el archivo de ColdFusion guardado y cargado en el paso 2).
11. Vea y compruebe el archivo SWF en un navegador.

Para más información, consulte la entrada %{LoadVars}% de *Referencia del lenguaje ActionScript 2.0*.

Flash Player 8 introdujo el controlador de eventos `onHTTPStatus` para las clases LoadVars, XML y MovieClipLoader a fin de que los usuarios puedan acceder al código de estado desde una solicitud HTTP. Esto permite a los desarrolladores determinar *por qué* no se ha podido realizar una operación de carga específica, en lugar de determinar únicamente que dicha operación ha fallado.

En el siguiente ejemplo se muestra cómo puede utilizar el controlador de eventos `onHTTPStatus` de la clase LoadVars para comprobar si un archivo de texto se ha descargado correctamente desde el servidor y cuál fue el código de estado devuelto desde la solicitud HTTP.

### Para comprobar el estado HTTP con el objeto LoadVars:

1. Cree un nuevo documento de Flash y guárdelo como **loadvars.fla**.
2. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
this.createTextField("params_txt", 10, 10, 10, 100, 21);
params_txt.autoSize = "left";

var my_lv:LoadVars = new LoadVars();
my_lv.onHTTPStatus = function(httpStatus:Number) {
  trace("HTTP status is: " + httpStatus);
};
my_lv.onLoad = function(success:Boolean) {
  if (success) {
    trace("text file successfully loaded");
    params_txt.text = my_lv.dayNames;
  } else {
    params_txt.text = "unable to load text file";
  }
};
my_lv.load("http://www.helpexamples.com/flash/404.txt");
/* output:
  Error opening URL "http://www.helpexamples.com/flash/404.txt"
  HTTP status is: 404
*/
```

El código anterior crea un nuevo campo de texto en el escenario y activa la asignación de tamaño automática de los campos de texto. A continuación, se crean un objeto LoadVars y dos controladores de eventos: `onHTTPStatus` y `onLoad`. El controlador de eventos `onHTTPStatus` es una característica nueva de Flash Player 8 y se invoca cuando se ha completado una operación `LoadVars.load()` o `LoadVars.sendAndLoad()`. El valor pasado a la función de controlador de eventos `onHTTPStatus` (`httpStatus` en el código anterior) contiene la definición de código de estado HTTP para la operación de carga actual. Si el archivo SWF ha podido cargar correctamente el archivo de texto, el valor de `httpStatus` se establece en 200 (el código de estado en HTTP para "OK"). Si el archivo no existía en el servidor, el valor de `httpStatus` se establece en 404 (el código de estado en HTTP para "No encontrado"). Una vez finalizada la carga del archivo, se llama al segundo controlador de eventos: `LoadVars.onLoad()`. Si el archivo se ha cargado correctamente, el valor del parámetro `success` se establece en `true`; en caso contrario, `success` se establece en `false`. Por último, el archivo externo se carga mediante el método `LoadVars.load()`.

3. Seleccione Control > Probar película para probar el documento de Flash.

   Flash muestra un mensaje de error en el panel Salida que indica que no ha podido cargar la imagen debido a que ésta no existe en el servidor. El controlador de eventos `onHTTPStatus` realiza un seguimiento del código de estado 404 ya que no se pudo encontrar el archivo en el servidor, mientras que el controlador de eventos `onLoad` establece la propiedad de texto del campo de texto `params_txt` como "no se puede cargar el archivo de texto".

| ATENCIÓN | Si un servidor Web no devuelve un código de estado a Flash Player, se devolverá el número 0 al controlador de eventos `onHTTPStatus`. |
|---|---|

# Carga y descarga de archivos

La clase FileReference permite añadir la capacidad de carga y descarga de archivos entre un cliente y un servidor. Los usuarios pueden cargar o descargar archivos entre sus equipos y un servidor. Mediante un cuadro de diálogo (como el cuadro de diálogo Abrir del sistema operativo Windows), se solicita a los usuarios que seleccionen un archivo para cargar o una ubicación para la descarga.

Todos los objetos FileReference que se crean con ActionScript hacen referencia a un único archivo en el disco duro del usuario. Estos objetos tienen propiedades que contienen información sobre el tamaño, tipo, nombre, fecha de creación y fecha de modificación del archivo. En Macintosh, también existe una propiedad para el tipo de creador del archivo.

Puede crear una instancia de la clase FileReference de dos formas distintas. Puede utilizar el siguiente operador `new`:

```
import flash.net.FileReference;
var myFileReference:FileReference = new FileReference();
```

O bien, llamar al método `FileReferenceList.browse()`, que abre un cuadro de diálogo en el sistema del usuario en el que se le solicita que seleccione un archivo para cargar y, a continuación, crea una matriz de objetos FileReference si el usuario selecciona uno o varios archivos correctamente. Cada objeto FileReference representa un archivo seleccionado por el usuario en el cuadro de diálogo. Un objeto FileReference no contiene datos de las propiedades FileReference (como `name`, `size` o `modificationDate`) hasta que se llama a los métodos `FileReference.browse()` o `FileReferenceList.browse()` y el usuario selecciona un archivo en el selector de archivos o se utiliza el método `FileReference.download()` para seleccionar un archivo en el selector de archivos.

**NOTA**

`FileReference.browse()` permite al usuario seleccionar un único archivo.
`FileReferenceList.browse()` permite al usuario seleccionar varios archivos.

Después de llamar con éxito al método `browse()`, se llama a `FileReference.upload()` para cargar un archivo cada vez.

También puede añadir funciones de descarga a la aplicación de Flash. El método `FileReference.download()` solicita a los usuarios finales que seleccionen una ubicación en el disco duro para guardar un archivo de un servidor. Este método también inicia la descarga desde una URL remota. Cuando se utiliza el método `download()`, sólo está accesible la propiedad `FileReference.name` cuando se distribuye el evento `onSelect`. Las demás propiedades no están accesibles hasta que se distribuye el evento `onComplete`.

| NOTA | Cuando aparece un cuadro de diálogo en el equipo del usuario final, la ubicación predeterminada que se muestra es la carpeta a la que se ha accedido más recientemente (si se puede determinar esa ubicación) o el escritorio (si no se puede determinar dicha carpeta). Las API FileReference y FileReferenceList no permiten establecer la ubicación de archivos predeterminada. |
|---|---|

Para obtener información sobre la funcionalidad y seguridad de la API FileReference, consulte "Funcionalidad y seguridad de la API FileReference" en la página 683. Para ver un ejemplo de una aplicación que utiliza la API FileReference, consulte "Adición de funcionalidad de carga de archivos a una aplicación" en la página 684. Encontrará el archivo de origen de muestra de este ejemplo, FileUpload.fla, en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\FileUpload.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/FileUpload.

Para obtener información sobre los métodos, propiedades y eventos de la API FileReference, consulte `%{FileReference (flash.net.FileReference)}%` y `%{FileReferenceList (flash.net.FileReferenceList)}%` en *Referencia del lenguaje ActionScript 2.0*.

## Funcionalidad y seguridad de la API FileReference

Flash Player y la API FileReference (consulte "Carga y descarga de archivos" en la página 682) admiten la carga y descarga de archivos de hasta 100 MB. La API FileReference *no* permite que la aplicación Flash que inicia la transferencia de archivos realice las siguientes acciones:

- Acceder al archivo cargado o descargado
- Acceder a la ruta del archivo en el equipo del usuario

Cuando un servidor requiere autenticación, la única operación que puede llevarse a cabo es la descarga de archivos con el plug-in del navegador de Flash Player. La carga en todos los Flash Players, o la descarga a través del reproductor Flash Player independiente o externo, falla en los servidores que requieren autenticación. Utilice detectores de eventos FileReference para determinar si las operaciones se realizan correctamente o para gestionar errores.

La carga y descarga de archivos están limitadas al dominio del archivo SWF, incluidos los dominios que se especifican mediante un archivo de política entre dominios. Deberá incluir un archivo de política en el servidor si el archivo SWF que inicia la carga o descarga no procede del mismo dominio que el servidor. Para más información sobre archivos de política de varios dominios y seguridad, consulte "Dominios, seguridad entre dominios y archivos SWF" en la página 735.

Mientras se ejecutan las llamadas a `FileReference.browse()`, `FileReferenceList.browse()` o `FileReference.download()`, la reproducción del archivo SWF realiza una pausa en las siguientes plataformas: plug-ins de navegador de Flash Player en Mac OS X, Flash Player externo de Macintosh y el reproductor independiente de Macintosh en Mac OS X 10.1 y versiones anteriores. El archivo SWF continúa ejecutándose en todos los reproductores de Windows y en el reproductor Flash Player independiente en Mac OS X 10.2 y versiones posteriores.

| ADVERTENCIA | Cuando permita a los usuarios cargar archivos en un servidor, deberá asegurarse de comprobar el tipo de archivo antes de guardarlo en el disco duro. Por ejemplo, no deseará permitir que un usuario cargue un script de servidor que se pueda utilizar para eliminar carpetas o archivos en el servidor. Si sólo desea permitir a los usuarios la carga de archivos de imágenes, asegúrese de que el script de servidor que carga los archivos compruebe que sean imágenes válidas. |
|---|---|

Para ver un ejemplo de una aplicación que utiliza la API FileReference, consulte "Adición de funcionalidad de carga de archivos a una aplicación" en la página 684.

# Adición de funcionalidad de carga de archivos a una aplicación

En el siguiente procedimiento se muestra cómo crear una aplicación que permite cargar archivos de imagen en un servidor. Esta aplicación permite a los usuarios seleccionar una imagen en el disco duro para cargarla y, a continuación, enviarla a un servidor. La imagen que cargan aparece después en el archivo SWF que utilizaron para cargarla.

A continuación del ejemplo en el que se crea la aplicación de Flash, se muestra un ejemplo en el que se detalla el código de servidor. Recuerde que el tamaño de los archivos de imagen está limitado: sólo se pueden cargar imágenes de hasta 200 KB.

### Para crear una aplicación de FLA mediante la API FileReference:

1. Cree un nuevo documento de Flash y guárdelo como **fileref.fla**.
2. Abra el panel Componentes, arrastre un componente ScrollPane al escenario y asígnele el nombre de instancia **imagePane**. (El tamaño y la posición de la instancia de ScrollPane se modifican más adelante mediante ActionScript.)
3. Arrastre un componente Button al escenario y asígnele el nombre de instancia **uploadBtn**.
4. Arrastre dos componentes Label al escenario y asígneles los nombres de instancia **imageLbl** y **statusLbl**.
5. Arrastre un componente ComboBox al escenario y asígnele el nombre de instancia **imagesCb**.

6. Arrastre un componente TextArea al escenario y asígnele el nombre de instancia **statusArea**.
7. Cree un nuevo símbolo de clip de película en el escenario y ábralo para editarlo (haga doble clic en la instancia para abrirla en modo de edición de símbolos).
8. Cree un nuevo campo de texto estático dentro del clip de película y añada el texto siguiente:

   **El archivo que ha intentado descargar no se encuentra en el servidor.**

   En la aplicación final, esta advertencia podría aparecer por uno de los siguientes motivos, entre otros:

   - La imagen se eliminó de la cola en el servidor cuando se cargaron otras imágenes.
   - El servidor no copió la imagen porque el tamaño del archivo superaba los 200 KB.
   - El tipo de archivo no era un archivo JPEG, GIF o PNG válido.

   **NOTA** La anchura del campo de texto debe ser inferior a la de la instancia de ScrollPane (400 píxeles) ya que, de lo contrario, los usuarios deberán desplazarse horizontalmente para ver el mensaje de error
9. Haga clic con el botón derecho del ratón en el símbolo en la Biblioteca y seleccione Vinculación del menú contextual.
10. Seleccione las casillas de verificación Exportar para ActionScript y Exportar en primer fotograma y escriba **Mensaje** en el cuadro de texto Identificador. Haga clic en Aceptar.
11. Añada el código ActionScript siguiente al fotograma 1 de la línea de tiempo:

    **NOTA** En los comentarios de código se incluyen detalles sobre la funcionalidad. La información general de un código sigue este ejemplo.

```
import flash.net.FileReference;

imagePane.setSize(400, 350);
imagePane.move(75, 25);
uploadBtn.move(75, 390);
uploadBtn.label = "Upload Image";
imageLbl.move(75, 430);
imageLbl.text = "Select Image";
statusLbl.move(210, 390);
statusLbl.text = "Status";
imagesCb.move(75, 450);
statusArea.setSize(250, 100);
statusArea.move(210, 410);
```

```
/* El objeto detector detecta eventos FileReference. */
var listener:Object = new Object();

/* Cuando el usuario selecciona un archivo, se llama al método onSelect()
  y se pasa una referencia al objeto FileReference. */
listener.onSelect = function(selectedFile:FileReference):Void {
  /* Actualizar el componente TextArea para notificar al usuario que
  Flash está intentando cargar la imagen. */
  statusArea.text += "Attempting to upload " + selectedFile.name + "\n";
  /* Cargar el archivo en el script PHP del servidor. */
  selectedFile.upload("http://www.helpexamples.com/flash/file_io/
  uploadFile.php");
};

/* Cuando el archivo comienza a cargarse, se llama al método onOpen()
  para notificar al usuario de que el archivo se está comenzando a
  cargar. */
listener.onOpen = function(selectedFile:FileReference):Void {
  statusArea.text += "Opening " + selectedFile.name + "\n";
};

/* Una vez cargado el archivo, se llama al método onComplete(). */
listener.onComplete = function(selectedFile:FileReference):Void {
  /* Notificar al usuario de que Flash está comenzando a descargar la
  imagen. */
  statusArea.text += "Downloading " + selectedFile.name + " to
  player\n";
  /* Añadir la imagen al componente ComboBox. */
  imagesCb.addItem(selectedFile.name);
  /* Establecer el índice seleccionado del componente ComboBox en la
  imagen que se ha añadido más recientemente. */
  imagesCb.selectedIndex = imagesCb.length - 1;
  /* Llamar a la función personalizada downloadImage(). */
  downloadImage();
};

var imageFile:FileReference = new FileReference();
imageFile.addListener(listener);

imagePane.addEventListener("complete", imageDownloaded);
imagesCb.addEventListener("change", downloadImage);
uploadBtn.addEventListener("click", uploadImage);

/* Si la imagen no se descarga, la propiedad Total del objeto Event será
  igual a -1. En ese caso, mostrar un mensaje al usuario. */
function imageDownloaded(event:Object):Void {
  if (event.total == -1) {
    imagePane.contentPath = "Message";
  }
}
```

```
/* Cuando el usuario selecciona una imagen del componente ComboBox, o
  cuando se llama a la función downloadImage() directamente desde el
  método listener.onComplete(), la función downloadImage() establece el
  parámetro contentPath de ScrollPane para iniciar la descarga de la
  imagen en el reproductor. */
function downloadImage(event:Object):Void {
  imagePane.contentPath = "http://www.helpexamples.com/flash/file_io/
  images/" + imagesCb.value;
}

/* Cuando el usuario hace clic en el botón, Flash llama a la función
  uploadImage() y abre un cuadro de diálogo para buscar archivos. */
function uploadImage(event:Object):Void {
  imageFile.browse([{description: "Image Files", extension:
  "*.jpg;*.gif;*.png"}]);
}
```

En primer lugar, este código ActionScript importa la clase FileReference e inicializa, coloca y cambia el tamaño de los componentes en el escenario. A continuación, se define un objeto detector y tres controladores de eventos: `onSelect`, `onOpen` y `onComplete`. Después, se añade el objeto detector a un nuevo objeto FileReference denominado `imageFile`. A continuación, se añaden detectores de eventos a las instancias `imagePane` de ScrollPane, `imagesCb` de ComboBox y `uploadBtn` de Button. Cada una de las funciones de detector de eventos se define en el código que se muestra a continuación de esta sección de código.

La primera función, `imageDownloaded()`, comprueba si la cantidad de bytes totales de las imágenes descargadas es -1 y, en ese caso, establece la propiedad `contentPath` para la instancia de ScrollPane en el clip de película con el identificador de vinculación de Mensaje, que se creó en un paso anterior. La segunda función, `downloadImage()`, intenta descargar la imagen que se ha cargado recientemente en la instancia de ScrollPane. Una vez descargada la imagen, se activa la función `imageDownloaded()` definida anteriormente, que comprueba si se ha descargado la imagen correctamente. La última función, `uploadImage()`, abre un cuadro de diálogo para buscar archivos, que filtra todas las imágenes JPEG, GIF y PNG.

12. Guarde los cambios en el documento.
13. Seleccione Archivo > Configuración de publicación, elija la ficha Formatos y asegúrese de que Flash y HTML estén seleccionados.
14. (Opcional) En el cuadro de diálogo Configuración de publicación, seleccione la ficha Formatos y elija Acceder sólo a la red en el menú emergente Seguridad de reproducción local.

    Si completa este paso, no encontrará restricciones de seguridad al probar el documento en un navegador local.

15. En el cuadro de diálogo Configuración de publicación, haga clic en Publicar para crear los archivos HTML y SWF.

    Cuando haya terminado, continúe con el siguiente procedimiento, en el que creará el contenedor del archivo SWF.

Encontrará el archivo de origen de muestra de este ejemplo, FileUpload.fla, en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\FileUpload.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/FileUpload.

El siguiente procedimiento requiere que PHP esté instalado en el servidor Web. Asimismo, deberá disponer de permisos de escritura para las subcarpetas denominadas Imágenes y Temporal. Primero debe completar el procedimiento anterior o utilizar el archivo SWF acabado disponible en las carpetas anteriormente indicadas.

### Para crear un script de servidor para la aplicación de carga de imagen:

1. Cree un nuevo documento PHP con un editor de texto como Dreamweaver o Bloc de notas.
2. Añada el siguiente código PHP al documento. (La información general de un código sigue este script.)

```
<?php

$MAXIMUM_FILESIZE = 1024 * 200; // 200 KB
$MAXIMUM_FILE_COUNT = 10; // mantener 10 archivos como máximo en el
  servidor
echo exif_imagetype($_FILES['Filedata']);
if ($_FILES['Filedata']['size'] <= $MAXIMUM_FILESIZE) {
  move_uploaded_file($_FILES['Filedata']['tmp_name'], "./temporary/
  ".$_FILES['Filedata']['name']);
  $type = exif_imagetype("./temporary/".$_FILES['Filedata']['name']);
  if ($type == 1 || $type == 2 || $type == 3) {
    rename("./temporary/".$_FILES['Filedata']['name'], "./images/
  ".$_FILES['Filedata']['name']);
  } else {
    unlink("./temporary/".$_FILES['Filedata']['name']);
  }
}
$directory = opendir('./images/');
$files = array();
while ($file = readdir($directory)) {
  array_push($files, array('./images/'.$file, filectime('./images/
  '.$file')));
}
```

```
usort($files, sorter);
if (count($files) > $MAXIMUM_FILE_COUNT) {
  $files_to_delete = array_splice($files, 0, count($files) -
  $MAXIMUM_FILE_COUNT);
  for ($i = 0; $i < count($files_to_delete); $i++) {
    unlink($files_to_delete[$i][0]);
  }
}
print_r($files);
closedir($directory);

function sorter($a, $b) {
  if ($a[1] == $b[1]) {
    return 0;
  } else {
    return ($a[1] < $b[1]) ? -1 : 1;
  }
}
?>
```

Este código PHP define, en primer lugar, dos variables de tipo Constant: `$MAXIMUM_FILESIZE` y `$MAXIMUM_FILE_COUNT`. Estas variables determinan el tamaño máximo (en kilobytes) de la imagen que se carga en el servidor (200 KB), así como el número de archivos cargados recientemente que se pueden mantener en la carpeta Imágenes (10). Si el tamaño de archivo de la imagen que se está cargando es inferior o igual al valor de `$MAXIMUM_FILESIZE`, la imagen se mueve a la carpeta Temporal.

A continuación, se comprueba el tipo de archivo cargado para asegurarse de que la imagen está en formato JPEG, GIF o PNG. Si es un tipo de imagen compatible, se copia de la carpeta Temporal a la carpeta Imágenes. Si el archivo cargado no era un tipo de imagen permitido se eliminará del sistema de archivos.

Después, se crea una lista de directorio de la carpeta Imágenes y se reproduce indefinidamente mediante un bucle while. Cada archivo en la carpeta Imágenes se añade a una matriz y, a continuación, se ordena. Si el número actual de archivos en la carpeta Imágenes es superior al valor de `$MAXIMUM_FILE_COUNT`, se eliminarán archivos hasta que sólo quede un número de imágenes igual al valor de `$MAXIMUM_FILE_COUNT`. De este modo, se evita que la carpeta Imágenes aumente hasta un tamaño que no se pueda manejar, ya que sólo puede haber 10 imágenes a la vez en la carpeta y cada imagen debe tener un tamaño máximo de 200 KB (aproximadamente 2 MB de imágenes como máximo).

3. Guarde los cambios en el documento PHP.
4. Cargue los archivos SWF, HTML y PHP en el servidor Web.
5. Vea el documento HTML remoto en un navegador Web y haga clic en el botón para cargar imagen en el archivo SWF.

6. Busque un archivo de imagen en el disco duro y, en el cuadro de diálogo, seleccione Abrir.

   El archivo SWF carga el archivo de imagen en el documento PHP remoto y lo muestra en ScrollPane (que añade barras de desplazamiento si fuera necesario). Si desea ver una imagen que se ha cargado anteriormente, puede seleccionar el nombre de archivo desde la instancia de ComboBox en el escenario. Si el usuario intenta cargar un tipo de imagen no permitido (sólo se permiten las imágenes JPEG, GIF o PNG) o si el tamaño de archivo es demasiado grande (más de 200 KB), Flash muestra el mensaje de error del clip de película Mensaje en la Biblioteca.

Puede encontrar el archivo de origen de muestra de este ejemplo, FileUpload.fla, en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\FileUpload.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/FileUpload.

Para más información sobre la seguridad de archivos locales, consulte "Seguridad de archivos local y Flash Player" en la página 717.

Para más información sobre la escritura de PHP, visite www.php.net/.

# Lenguaje XML

*XML (Lenguaje extensible de marcado, Extensible Markup Language)* se está convirtiendo en el estándar para el intercambio de datos estructurados en aplicaciones de Internet. Puede integrar datos en Flash con servidores que utilizan tecnología XML para crear aplicaciones complejas, como las aplicaciones de chat o los sistemas de cotización de valores.

En XML, al igual que en HTML, se utilizan etiquetas para especificar o *marcar* un cuerpo de texto. En HTML se utilizan etiquetas predefinidas para indicar cómo debe aparecer el texto en un navegador Web (por ejemplo, la etiqueta `<b>` indica que el texto debe aparecer en negrita). En XML, debe definir las etiquetas que identifican el tipo de una parte de datos (por ejemplo, `<password>VerySecret</password>`). XML separa la estructura de la información del modo en el que ésta se muestra, lo que permite que el mismo documento XML se pueda utilizar y reutilizar en diferentes entornos.

Cada etiqueta XML se denomina *nodo* o elemento. Cada nodo tiene un tipo (1, que indica un elemento XML o 3, que indica un nodo de texto) y los elementos también pueden tener atributos. Un nodo anidado en otro nodo se denomina *nodo secundario*. La estructura jerárquica de árbol de los nodos se llama XML DOM (parecida a JavaScript DOM, que es la estructura de los elementos de un navegador Web).

En el ejemplo siguiente, `<portfolio>` es el nodo principal, no tiene atributos y contiene el nodo secundario `<holding>` que tiene los atributos `symbol`, `qty`, `price` y `value`:

```
<portfolio>
  <holding symbol="rich"
    qty="75"
    price="245.50"
    value="18412.50" />
</portfolio>
```

Para más información, consulte los temas siguientes:

- "Utilización de la clase XML" en la página 692
- "Utilización de la clase XMLSocket" en la página 698

Para más información sobre XML, consulte www.w3.org/XML.

Existen varios archivos de ejemplo en el disco duro que cargan XML en un archivo SWF durante la ejecución. Un ejemplo muestra cómo crear un rastreador de registros Web mediante la carga, el análisis y la manipulación de datos XML. Encontrará el archivo de origen de muestra, xml_blogTracker.fla, en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\XML_BlogTracker.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/XML_BlogTracker.

Un segundo ejemplo muestra cómo utilizar XML y las matrices anidadas a fin de seleccionar cadenas en diferentes idiomas para llenar campos de texto. El archivo de origen de muestra, xml_languagePicker.fla, se puede encontrar en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\XML_LanguagePicker.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/XML_LanguagePicker.

Un tercer ejemplo muestra cómo crear un menú dinámico con datos XML. El ejemplo llama al constructor `XmlMenu()` de ActionScript y le pasa dos parámetros: la ruta al archivo de menú XML y una referencia a la línea de tiempo actual. El resto de la funcionalidad reside en un archivo de clase personalizado, XmlMenu.as.

El archivo de origen de muestra, xmlmenu.fla, se puede encontrar en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\XML_Menu.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/XML_Menu.

# Utilización de la clase XML

Los métodos de la clase XML de ActionScript (por ejemplo, `appendChild()`, `removeNode()` e `insertBefore()`) permiten estructurar los datos XML en Flash para enviarlos a un servidor y manipular e interpretar los datos XML descargados.

Los métodos de clase XML siguientes envían datos XML a un servidor y los cargan en dicho servidor con el método `POST` de HTTP:

- El método `load()` descarga XML de un URL y lo coloca en un objeto XML de ActionScript.
- El método `send()` codifica el objeto XML en un documento XML y lo envía a un URL determinado utilizando el método `POST`. Si se especifica, los datos devueltos se mostrarán en una ventana de navegador.
- El método `sendAndLoad()` envía un objeto XML a una URL. La información devuelta se coloca en un objeto XML de ActionScript.

Por ejemplo, puede crear un sistema de cotización de valores que almacene toda la información (nombres de usuario, contraseñas, ID de sesión, valores de la cartera e información de transacciones) en una base de datos.

El script de servidor que pasa la información entre Flash y la base de datos lee y graba los datos en formato XML. Puede utilizar ActionScript para convertir la información recopilada en el archivo SWF (por ejemplo, un nombre de usuario y una contraseña) en un objeto XML y después enviar los datos al script de servidor como un documento XML. También puede utilizar ActionScript para cargar el documento XML que el servidor devuelve a un objeto XML para utilizarlo en el archivo SWF.

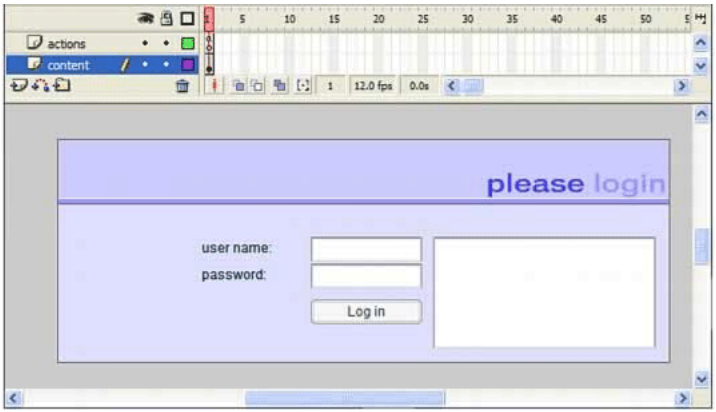

*Flujo y conversión de datos entre un archivo SWF, un script de servidor y una base de datos*

La validación de la contraseña para el sistema de cotización de valores requiere dos scripts: una función definida en el fotograma 1 y un script que crea y envía los objetos XML creados en el documento.

Cuando un usuario introduce información en los campos de texto del archivo SWF con las variables `username` y `password`, las variables deben convertirse a XML antes de pasarlas al servidor. La primera sección del script carga las variables en un objeto XML recién creado llamado `loginXML`. Cuando un usuario hace clic en un botón para iniciar una sesión, el objeto `loginXML` se convierte en una cadena de XML y se envía al servidor.

El siguiente código ActionScript se sitúa en la línea de tiempo y se utiliza para enviar al servidor datos con formato XML. Para entender este script, lea las líneas de comentarios (indicadas por los caracteres `//`):

```
//se ignoran los espacios en blanco de XML
XML.prototype.ignoreWhite = true;
// Construir un objeto XML para albergar la respuesta del servidor
var loginReplyXML:XML = new XML();
// esta función se activa cuando se recibe un paquete XML desde el servidor.
loginReplyXML.onLoad = function(success:Boolean) {
  if (success){
    //(opcional) Crear dos campos de texto para estado/depuración
    //status_txt.text = this.firstChild.attributes.status;
    // debug_txt.text = this.firstChild;
    switch (this.firstChild.attributes.STATUS) {
    case 'OK' :
      _global.session = this.firstChild.attributes.SESSION;
      trace(_global.session);
      gotoAndStop("welcome");
      break;
    case 'FAILURE' :
      gotoAndStop("loginfailure");
      break;
    default :
      // esto no debe ocurrir nunca
      trace("Unexpected value received for STATUS.");
    }
  } else {
    trace("an error occurred.");
  }
};
// esta función se activa cuando se hace clic en login_btn
login_btn.onRelease = function() {
  var loginXML:XML = new XML();
  //crear datos con formato XML para enviarlos al servidor
  var loginElement:XMLNode = loginXML.createElement("login");
  loginElement.attributes.username = username_txt.text;
  loginElement.attributes.password = password_txt.text;
  loginXML.appendChild(loginElement);
  // enviar los datos con formato XML al servidor
  loginXML.sendAndLoad("http://www.flash-mx.com/mm/main.cfm",
  loginReplyXML);
};
```




Puede comprobar este código utilizando el nombre de usuario `JeanSmith` y la contraseña `VerySecret`. La primera sección del script genera el siguiente código XML cuando el usuario hace clic en el botón login:

```
<login username="JeanSmith" password="VerySecret" />
```

El servidor recibe el XML, genera una respuesta XML y la envía al archivo SWF. Si se acepta la contraseña, el servidor responde con el mensaje que se muestra a continuación:

```
<LOGINREPLY STATUS="OK" SESSION="4D968511" />
```

Este XML incluye un atributo `session` que contiene un ID de sesión exclusivo generado de forma aleatoria que se utiliza en todas las comunicaciones entre el cliente y el servidor durante el resto de la sesión. Si se rechaza la contraseña, el servidor responde con el mensaje que se muestra a continuación:

```
<LOGINREPLY STATUS="FAILURE" />
```

El nodo XML `loginreply` debe cargarse en un objeto XML vacío en el archivo SWF. La sentencia siguiente crea el objeto XML `loginreplyXML` para recibir el nodo XML:

```
// Construir un objeto XML para albergar la respuesta del servidor
var loginReplyXML:XML = new XML();
loginReplyXML.onLoad = function(success:Boolean) {
```

La segunda sentencia de este código ActionScript define una función anónima (en línea), a la que se llama cuando se activa el evento `onLoad`.

El botón login (instancia de `login_btn`) se utiliza para enviar el nombre de usuario y la contraseña como XML al servidor y cargar una respuesta XML en el archivo SWF. Puede hacerlo utilizando el método `sendAndLoad()`, como se muestra en el siguiente ejemplo:

```
loginXML.sendAndLoad("http://www.flash-mx.com.com/mm/main.cfm",
  loginReplyXML);
```

En primer lugar, se crean los datos con formato XML a partir de los valores introducidos por el usuario en el archivo SWF, tras lo cual ese objeto XML se envía empleando el método `sendAndLoad`. De forma similar a los datos procedentes de una función `loadVariables()`, el elemento XML `loginreply` llega de forma asíncrona (es decir, no espera a que haya resultados para realizar la respuesta) y se carga en el objeto `loginReplyXML`. Cuando llegan los datos, se llama al controlador `onLoad` del objeto `loginReplyXML`. Debe definir la función `loginReplyXML`, a la que se llama cuando se desencadena el controlador `onLoad`, para que pueda procesar el elemento `loginreply`.

| NOTA | Esta función siempre debe encontrarse en el fotograma que contiene el código ActionScript para el botón login. |
|---|---|

Si el inicio de sesión es correcto, el archivo SWF pasa a la etiqueta de fotograma `welcome`. Si el inicio de sesión no es correcto, la cabeza lectora se desplaza a la etiqueta de fotograma `loginfailure`. Esto se procesa mediante una condición y una sentencia case. Para más información sobre las sentencias case y break, consulte %{case statement}% y %{break statement}% en *Referencia del lenguaje ActionScript 2.0*. Para más información sobre las condiciones, consulte %{if statement}% y %{else statement}% en *Referencia del lenguaje ActionScript 2.0*.

| NOTA | Este diseño es solamente un ejemplo y Macromedia no garantiza el nivel de seguridad que proporciona. Si está implementando un sistema de seguridad protegido mediante contraseña, asegúrese de que comprende perfectamente la seguridad de la red. |
|---|---|

Para más información, consulte Integración de XML y Flash en una aplicación Web en la dirección www.macromedia.com/support/flash/interactivity/xml/ y la entrada %{XML}% en *Referencia del lenguaje ActionScript 2.0*. Para más información sobre la seguridad de archivos locales, consulte "Seguridad de archivos local y Flash Player" en la página 717.

Encontrará un archivo de origen de muestra, login.fla, en la carpeta Samples del disco duro. Este ejemplo muestra cómo añadir funcionalidad sencilla de inicio de sesión a los sitios Web con ActionScript 2.0. El ejemplo utiliza ActionScript y componentes para crear un pequeño formulario en el que se introducen un nombre de usuario y una contraseña y, a continuación, se hace clic en un botón para entrar en un sitio.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\Login.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/Login.

Flash Player 8 introdujo el controlador de eventos `onHTTPStatus` para las clases XML, LoadVars y MovieClipLoader a fin de que los usuarios puedan acceder al código de estado desde una solicitud HTTP. Esto permite a los desarrolladores determinar *por qué* no se ha podido realizar una operación de carga específica, en lugar de determinar únicamente que dicha operación ha fallado.

En el siguiente ejemplo se muestra cómo puede utilizar el controlador de eventos `onHTTPStatus` de la clase XML para comprobar si un archivo XML se ha descargado correctamente desde el servidor y cuál fue el código de estado devuelto desde la solicitud HTTP.

### Para comprobar los códigos de estado HTTP mediante la clase XML:

1. Cree un nuevo documento de Flash y guárdelo como **xmlhttp.fla**.
2. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
var my_xml:XML = new XML();
my_xml.ignoreWhite = true;
my_xml.onHTTPStatus = function(httpStatus:Number) {
  trace("HTTP status is: " + httpStatus);
};
my_xml.onLoad = function(success:Boolean) {
  if (success) {
    trace("XML successfully loaded");
    // 0 (No error; parse was completed successfully.)
    trace("XML status is: " + my_xml.status);
  } else {
    trace("Unable to load XML");
  }
};
my_xml.load("http://www.helpexamples.com/crossdomain.xml");
```

En el código anterior se define un nuevo objeto XML con el nombre de variable `my_xml`, se definen dos controladores de eventos (`onHTTPStatus` y `onLoad`) y se carga un archivo XML externo. El controlador de eventos `onLoad` comprueba si el archivo XML se cargó correctamente y, en ese caso, envía un mensaje al panel Salida. También realiza un seguimiento de la propiedad de estado del objeto XML. Se debe recordar que el detector de eventos `onHTTPStatus` devuelve el código de estado devuelto desde el servidor Web, mientras que la propiedad `XML.status` contiene un valor numérico que indica si el objeto XML se ha podido analizar correctamente.

3. Seleccione Control > Probar película para probar el documento de Flash.

**SUGERENCIA**

El controlador de eventos `XML.onHTTPStatus` es una característica nueva de Flash Player 8.

**ADVERTENCIA**

No se deben confundir los códigos de HTTP httpStatus con la propiedad `status` de la clase XML. El controlador de eventos `onHTTPStatus` devuelve el código de estado del servidor desde una solicitud HTTP, mientras que la propiedad `status` establece y devuelve automáticamente un valor numérico que indica si un documento XML se ha analizado correctamente en un objeto XML.

**ATENCIÓN**

Si un servidor Web no devuelve un código de `estado` a Flash Player, se devolverá el número 0 al controlador de eventos `onHTTPStatus`.

Existen varios archivos de ejemplo en el disco duro que cargan XML en un archivo SWF durante la ejecución. Un ejemplo muestra cómo crear un rastreador de registros Web mediante la carga, el análisis y la manipulación de datos XML. Encontrará el archivo de origen de muestra, xml_blogTracker.fla, en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\XML_BlogTracker.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/XML_BlogTracker.

Un segundo ejemplo muestra cómo utilizar XML y las matrices anidadas a fin de seleccionar cadenas en diferentes idiomas para llenar campos de texto. El archivo de origen de muestra, xml_languagePicker.fla, se puede encontrar en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\XML_LanguagePicker.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/XML_LanguagePicker.

Un tercer ejemplo muestra cómo crear un menú dinámico con datos XML. El ejemplo llama al constructor `XmlMenu()` de ActionScript y le pasa dos parámetros: la ruta al archivo de menú XML y una referencia a la línea de tiempo actual. El resto de la funcionalidad reside en un archivo de clase personalizado, XmlMenu.as.

El archivo de origen de muestra, xmlmenu.fla, se puede encontrar en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\XML_Menu.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/XML_Menu.

## Utilización de la clase XMLSocket

ActionScript proporciona una clase XMLSocket incorporada que permite abrir una conexión continua con un servidor. Una conexión de socket permite al servidor publicar o *enviar* la información al cliente tan pronto como ésta se encuentre disponible. Sin una conexión continua, el servidor debe esperar una solicitud HTTP. Esta conexión abierta elimina los problemas de latencia y se usa con frecuencia para las aplicaciones en tiempo real como los chats. Los datos se envían por la conexión de socket como una cadena y deberán estar en formato XML. Puede utilizar la clase XML para estructurar los datos.

Para crear una conexión de socket debe crear una aplicación de servidor que espere la solicitud de conexión de socket y envíe una respuesta al archivo SWF. Este tipo de aplicación de servidor puede escribirse en un lenguaje de programación como Java.

| NOTA | La clase XMLSocket no puede atravesar cortafuegos automáticamente, ya que, a diferencia del protocolo RTMP (Protocolo de mensajería en tiempo real, Real-Time Messaging Protocol), XMLSocket no dispone de prestaciones de tunelación HTTP. Si necesita utilizar tunelación HTTP, puede que le interese utilizar Flash Remoting o Flash Communication Server (que admite RTMP). |
|---|---|

Puede utilizar los métodos `connect()` y `send()` de la clase XMLSocket para transferir XML desde y hacia un servidor a través de una conexión de socket. El método `connect()` establece una conexión de socket con un puerto de servidor Web. El método `send()` pasa un objeto XML al servidor especificado en la conexión de socket.

Cuando se invoca el método `connect()`, Flash Player abre una conexión TCP/IP con el servidor y la mantiene abierta hasta que se produce uno de los siguientes eventos:

- Se llama al método `close()` de la clase XMLSocket.
- No existen más referencias al objeto XMLSocket.
- Se sale de Flash Player.
- Se interrumpe la conexión (por ejemplo, se desconecta el módem).

En el ejemplo siguiente se crea una conexión de socket XML y se envían los datos desde el objeto XML `myXML`. Para comprender el script, lea las líneas de comentarios (indicadas por los caracteres `//`):

```
// Crear un objeto XMLSocket
var theSocket:XMLSocket = new XMLSocket();
// Conectar con un sitio a través de un puerto libre superior a 1024
  mediante el método connect().
// Introducir localhost o 127.0.0.1 para comprobación local.
// Para un servidor en funcionamiento, introduzca su dominio
  www.yourdomain.com
theSocket.connect("localhost", 12345);
// muestra texto relativo a la conexión
theSocket.onConnect = function(myStatus) {
  if (myStatus) {
    conn_txt.text = "connection successful";
  } else {
    conn_txt.text = "no connection made";
  }
};
// datos para enviar
function sendData() {
  var myXML:XML = new XML();
  var mySend = myXML.createElement("thenode");
  mySend.attributes.myData = "someData";
  myXML.appendChild(mySend);
  theSocket.send(myXML);
}
// el botón envía los datos
sendButton.onRelease = function() {
  sendData();
};
// realiza un seguimiento de los datos devueltos de la conexión de socket
theSocket.onData = function(msg:String):Void {
  trace(msg);
};
```

Para más información, consulte la entrada %{MLSocket}% de *Referencia del lenguaje ActionScript 2.0*.

Para más información sobre la seguridad de archivos locales, consulte "Seguridad de archivos local y Flash Player" en la página 717.

# Envío de mensajes hacia y desde Flash Player

Para enviar mensajes desde un archivo SWF al entorno host correspondiente (por ejemplo, un navegador Web, una película de Macromedia Director o el reproductor Flash Player independiente), puede utilizar la función `fscommand()`.Esta función permite ampliar el archivo SWF mediante las funciones del host. Por ejemplo, puede pasar una función `fscommand()` a una función de JavaScript en una página HTML que abra una nueva ventana del navegador con propiedades específicas.

Para controlar un archivo SWF en Flash Player desde los lenguajes para scripts de los navegadores Web como JavaScript, VBScript y Microsoft JScript, puede utilizar los diferentes métodos de Flash Player (funciones que envían mensajes desde un entorno host al archivo SWF). Por ejemplo, puede tener un vínculo en una página HTML que envíe el archivo SWF a un fotograma específico.

Para más información, consulte los siguientes temas:

- "Utilización de la función fscommand()" en la página 700
- "Utilización de JavaScript para controlar aplicaciones Flash" en la página 703
- "Métodos de Flash Player" en la página 704

## Utilización de la función fscommand()

**NOTA**

La API externa sustituye al método `fscommand()` en Flash 8 para interoperar con una página HTML o una aplicación de contenedor. La API externa ofrece una funcionalidad más sólida que `fscommand()` en esta situación. Para más información, consulte "Interfaz API externa" en la página 704.

Utilice la función `fscommand()` para enviar un mensaje al programa que aloja Flash Player, como un navegador Web, por ejemplo.

**NOTA**

El uso de `fscommand()` para llamar a código JavaScript no funciona en los navegadores Safari o Internet Explorer para Macintosh.

La función `fscommand()` tiene dos parámetros: *`comando`* y *`argumentos`*. Para enviar un mensaje a la versión independiente de Flash Player, debe utilizar comandos y argumentos predefinidos. Por ejemplo, el siguiente controlador de evento establece que el reproductor independiente ajuste la escala del archivo SWF al tamaño de pantalla completa al soltar el botón:

```
my_btn.onRelease = function() {
  fscommand("fullscreen", true);
};
```

La tabla siguiente muestra los valores que puede especificar para los parámetros *`command`* y *`arguments`* de `fscommand()` a fin de controlar la reproducción y el aspecto de un archivo SWF que se ejecuta en el reproductor independiente, incluidos los proyectores.

**NOTA** Un *proyector* es un archivo SWF guardado en un formato que se puede ejecutar como una aplicación independiente, es decir, mediante la incorporación de Flash Player al contenido de un archivo ejecutable.

| Comando | Argumentos | Propósito |
|---|---|---|
| `quit` | Ninguno | Cierra el proyector. |
| `fullscreen` | `true` o `false` | Si se especifica `true`, Flash Player se establece en el modo de pantalla completa. Si se especifica `false`, el reproductor vuelve a la vista de menú normal. |
| `allowscale` | `true` o `false` | Si se especifica `false`, el reproductor se establece para que el archivo SWF se dibuje siempre con su tamaño original y nunca se cambie su escala. Si se especifica `true`, se obliga al archivo SWF a cambiar su escala al 100% del reproductor. |
| `showmenu` | `true` o `false` | Si se especifica `true`, se activa el conjunto completo de elementos de menú contextual. Si se especifica `false`, se atenúan todos los elementos de menú contextual excepto Configuración y Acerca de Flash Player. |
| `exec` | Ruta de acceso a la aplicación | Ejecuta una aplicación desde el proyector. |

Para utilizar `fscommand()` a fin de enviar un mensaje a un lenguaje de creación de scripts como JavaScript en un navegador Web, puede pasar los dos parámetros que desee a *`comando`* y *`argumentos`*. Estos parámetros pueden ser cadenas o expresiones y se utilizan en una función de JavaScript que "captura" o gestiona la función `fscommand()`.

Una función `fscommand()` invoca la función de JavaScript *`moviename`*`_DoFSCommand` en la página HTML que incorpora el archivo SWF, donde *`moviename`* es el nombre de Flash Player asignado mediante el atributo `name` de la etiqueta `embed` o el atributo `id` de la etiqueta `object`. Si el archivo SWF tiene asignado el nombre `myMovie`, la función de JavaScript invocada es `myMovie_DoFSCommand`.

### Para utilizar `fscommand()` a fin de abrir un cuadro de mensaje desde un archivo SWF en la página HTML a través de JavaScript, siga estos pasos:

1. Cree un archivo FLA nuevo y guárdelo como **myMovie.fla**.
2. Arrastre dos instancias del componente Button al escenario y asígneles los nombres de instancia **window_btn** y **alert_btn**, respectivamente, y las etiquetas **Open Window** y **Alert**.
3. Inserte una nueva capa en la línea de tiempo y cambie su nombre a **Actions**.
4. Seleccione el fotograma 1 de la capa Actions y añada el siguiente código ActionScript en el panel Acciones:

```
window_btn.onRelease = function() {
  fscommand("popup", "http://www.macromedia.com/");
};
alert_btn.onRelease = function() {
  fscommand("alert", "You clicked the button.");
};
```

5. Seleccione Archivo > Configuración de publicación y asegúrese de que se ha seleccionado Flash con FSCommand en el menú Plantilla de la ficha HTML.
6. Seleccione Archivo > Publicar para generar los archivos SWF y HTML.
7. En un editor de HTML o texto, abra el archivo HTML generado en el paso 6 y examine el código. Al publicar el archivo SWF con la plantilla Flash con FSCommand en la ficha HTML del cuadro de diálogo Configuración de publicación, se insertó código adicional en el archivo HTML. Los atributos `NAME` e `ID` del archivo SWF constituyen el nombre del archivo. Por ejemplo, para el archivo myMovie.fla, los atributos se establecerían en `myMovie`.
8. En el archivo HTML, añada el siguiente código JavaScript donde pone `// Place your code here.`:

```
if (command == "alert") {
  alert(args);
} else if (command == "popup") {
  window.open(args, "mmwin", "width=500,height=300");
}
```

(Para más información sobre la publicación, consulte el Capítulo 17, “Publicación” en *Utilización de Flash*.)

En las aplicaciones de Microsoft Internet Explorer, otra alternativa es asociar un controlador de eventos directamente a la etiqueta `<SCRIPT>`, como se muestra en el ejemplo siguiente:

```
<script Language="JavaScript" event="FSCommand (command, args)"
  for="theMovie">
...
</script>
```

9. Guarde y cierre el archivo HTML.

   Cuando edite archivos HTML fuera de Flash de este modo, recuerde que debe anular la selección de la casilla de verificación HTML en Archivo > Configuración de publicación, ya que, si no lo hace, Flash sobrescribirá el código HTML al volver a publicar.

10. En un navegador Web, abra el archivo HTML para verlo. Haga clic en el botón Open Window; se abrirá una ventana con el sitio Web de Macromedia. Haga clic en el botón Alert; aparecerá una ventana de alerta.

La función `fscommand()` puede enviar mensajes a Macromedia Director que Lingo interpreta como cadenas, eventos o código Lingo ejecutable. Si el mensaje es una cadena o un evento, debe escribir el código Lingo para recibirlo de la función `fscommand()` y llevar a cabo una acción en Director. Para más información, consulte el Centro de servicio técnico de Director en www.macromedia.com/support/director.

En Visual Basic, Visual C++ y otros programas que pueden albergar controles ActiveX, `fscommand()` envía un evento VB con dos cadenas que pueden gestionarse en el lenguaje de programación del entorno. Para más información, utilice las palabras clave *Flash method* para realizar búsquedas en el Centro de servicio técnico de Flash en www.macromedia.com/support/flash.

## Utilización de JavaScript para controlar aplicaciones Flash

Flash Player 6 (6.0.40.0) y versiones posteriores admiten determinados métodos JavaScript específicos de aplicaciones Flash, así como `FSCommand` en Netscape 6.2 y versiones posteriores. Las versiones anteriores no admiten ni los métodos JavaScript ni `FSCommand` en Netscape 6.2 o posteriores. Para más información, consulte el artículo del Centro de servicio técnico de Macromedia titulado “Scripting With Flash” (Creación de scripts con Flash) en www.macromedia.com/support/flash/publishexport/scriptingwithflash/.

En Netscape 6.2 y posteriores, no es necesario establecer el atributo `swliveconnect` con el valor `true`. Sin embargo, el establecimiento de `swLiveConnect` con el valor `true` no tiene efectos negativos para el archivo SWF. Para más información, consulte el atributo `swLiveConnect` en “Parámetros y atributos” en la página 541 de *Utilización de Flash*.

## Métodos de Flash Player

Puede utilizar los métodos de Flash Player para controlar un archivo SWF en Flash Player mediante los lenguajes de creación de scripts de navegador Web como JavaScript y VBScript. Al igual que con otros métodos, puede utilizar los métodos de Flash Player para enviar llamadas a los archivos SWF desde un entorno de creación de scripts que no sea ActionScript. Cada método tiene un nombre y la mayoría de los métodos aceptan parámetros. Un parámetro especifica un valor sobre el que opera el método. El cálculo realizado por algunos de los métodos devuelve un valor que puede ser utilizado por el entorno de scripts.

Dos tecnologías permiten la comunicación entre el navegador y Flash Player: LiveConnect (Netscape Navigator 3.0 o posteriores en Windows 95/98/2000/NT/XP o Power Macintosh) y ActiveX (Internet Explorer 3.0 y posteriores en Windows 95/98/2000/NT/XP). Aunque las técnicas de creación de scripts son similares para todos los navegadores y lenguajes, los controles ActiveX cuentan con propiedades y eventos adicionales.

Para más información, incluida una lista completa de los métodos de creación de scripts de Flash Player, utilice las palabras clave *Flash method* para realizar una búsqueda en el Centro de servicio técnico de Flash en www.macromedia.com/es/support/.

# Interfaz API externa

La clase ExternalInterface, también denominada *API externa*, es un nuevo subsistema que permite comunicar fácilmente desde ActionScript y el contenedor de Flash Player a una página HTML con JavaScript o a una aplicación de escritorio que incorpore Flash Player.

**NOTA**

Esta función sustituye a la antigua `fscommand()` para interoperar con una página HTML o una aplicación de contenedor. La API externa ofrece una funcionalidad más sólida que `fscommand()` en esta situación. Para más información, consulte "Interfaz API externa" en la página 704.

La clase ExternalInterface sólo está disponible en las siguientes circunstancias:

- En todas las versiones compatibles de Internet Explorer para Windows (5.0 y versiones posteriores).
- En un contenedor ActiveX incorporado personalizado, como una aplicación de escritorio que incorpore el control ActiveX de Flash Player.

- En cualquier navegador que admita la interfaz NPRuntime, entre los que se incluyen, actualmente, los siguientes:
  - Firefox 1.0 o posterior
  - Mozilla 1.7.5 o posterior
  - Netscape 8.0 o posterior
  - Safari 1.3 o posterior

En todas las demás situaciones, la propiedad `ExternalInterface.available` devuelve el valor false.

Desde ActionScript puede llamar a una función JavaScript en la página HTML. La API externa ofrece las siguientes funciones mejoradas en comparación con `fscommand()`:

- Se puede utilizar cualquier función JavaScript, no sólo las funciones que se usan con `%{fscommand function}%`.
- Se puede pasar cualquier número de argumentos, con cualquier nombre; no se limita a pasar un comando y argumentos.
- Se pueden pasar diversos tipos de datos (como Boolean, Number y String); ya no se está limitado a los parámetros de tipo String.
- Ahora se puede recibir el valor de una llamada y ese valor vuelve inmediatamente a ActionScript (como valor devuelto de la llamada que se realiza).

Se puede llamar a una función ActionScript desde JavaScript en una página HTML. Para más información, consulte `%{ExternalInterface (flash.external.ExternalInterface)}%`. Para más información sobre la seguridad de archivos locales, consulte "Seguridad de archivos local y Flash Player" en la página 717.

Las siguientes secciones contienen ejemplos que utilizan la API externa:

- "Creación de interacción con la API externa" en la página 705
- "Control de Flash Video con la API externa" en la página 709

## Creación de interacción con la API externa

Puede crear interacción entre el navegador y un archivo SWF incorporado en una página Web. El siguiente procedimiento envía texto a la página HTML que contiene el archivo SWF y el HTML envía un mensaje al archivo SWF en tiempo de ejecución.

### Para crear la aplicación de Flash:

1. Cree un nuevo documento de Flash y guárdelo como **extint.fla**.
2. Arrastre dos componentes TextInput al escenario y asígneles los nombres de instancia **in_ti** y **out_ti**.
3. Arrastre un componente Label al escenario, asígnele el nombre de instancia **out_lbl**, colóquelo sobre la instancia `out_ti` de TextInput y establezca la propiedad de texto en la ficha Parámetros del inspector de propiedades en **Sending to JS:**.
4. Arrastre un componente Button al escenario, colóquelo junto a la etiqueta `out_lbl` y asígnele el nombre de instancia **send_button**.
5. Arrastre un componente Label al escenario, asígnele el nombre de instancia **in_lbl**, colóquelo sobre la instancia `in_ti` de TextInput y establezca la propiedad de texto en la ficha Parámetros en **Receiving from JS:**.
6. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
import flash.external.ExternalInterface;

ExternalInterface.addCallback("asFunc", this, asFunc);
function asFunc(str:String):Void {
  in_ti.text = "JS > Hello " + str;
}

send_button.addEventListener("click", clickListener);
function clickListener(eventObj:Object):Void {
  trace("click > " + out_ti.text);
  ExternalInterface.call("jsFunc", out_ti.text);
}
```

   El código ActionScript anterior se divide en tres secciones. La primera sección importa la clase ExternalInterface, por lo que no necesita utilizar el nombre de clase completo. La segunda sección define una función callback, `asFunc()`, a la que se llama desde JavaScript en un documento HTML que se crea en un ejemplo siguiente. Esta función establece el texto de un componente TextInput en el escenario. La tercera sección de código define una función y le asigna un detector de eventos para cuando el usuario haga clic en la instancia del componente Button en el escenario. Cada vez que se hace clic en el botón, el archivo SWF llama a la función `jsFunc()` de JavaScript en la página HTML y pasa la propiedad de texto a la instancia de introducción de texto `out_ti`.
7. Seleccione Archivo > Configuración de publicación, elija la ficha Formatos y asegúrese de que Flash y HTML estén seleccionados.
8. Haga clic en Publicar para crear los archivos HTML y SWF.

   Cuando haya terminado, continúe con el siguiente procedimiento para crear el contenedor del archivo SWF.

Antes de probar el documento de Flash anterior, deberá modificar el código HTML generado y añadir algo de código HTML y JavaScript. El siguiente procedimiento modifica el contenedor HTML para el archivo SWF a fin de que los dos archivos puedan interactuar cuando se ejecutan en un navegador.

### Para crear el contenedor HTML del archivo SWF:

1. Complete el procedimiento anterior.
2. Abra el archivo extint.html que Flash crea cuando se publica la aplicación.

   Éste se encuentra en la misma carpeta que el documento de Flash.
3. Añada el siguiente código JavaScript entre las etiquetas `head` de apertura y cierre:

```
<script language="JavaScript">
<!--
  function thisMovie(movieName) {
    var isIE = navigator.appName.indexOf("Microsoft") != -1;
    return (isIE) ? window[movieName] : document[movieName];
  }

  function makeCall(str) {
    thisMovie("extint").asFunc(str);
  }

  function jsFunc(str) {
    document.inForm.inField.value = "AS > Hello " + str;
  }
// -->
</script>
```

   Este código JavaScript define tres métodos. El primer método devuelve una referencia a un archivo SWF incorporado en función de cuál sea el navegador del usuario: Microsoft Internet Explorer (IE) o Mozilla. La segunda función, `makeCall()`, llama al método `asFunc()` que se definió en el documento de Flash en el ejemplo anterior. El parámetro `"extint"` de la llamada de función `thisMovie()` se refiere al ID de objeto y al nombre embed del archivo SWF incorporado. Si ha guardado el documento de Flash con un nombre diferente, deberá cambiar esta cadena para que coincida con los valores de las etiquetas object y embed. La tercera función, `jsFunc()`, establece el valor del campo de texto `inField` en el documento HTML. La llamada a esta función se realiza desde el documento de Flash cuando un usuario hace clic en el componente Button `send_button`.

4. Añada el siguiente código HTML antes de la etiqueta de cierre `</body>`:

```
<form name="outForm" method="POST"
  action="javascript:makeCall(document.outForm.outField.value);">
  Sending to AS:<br />
  <input type="text" name="outField" value="" /><br />
  <input type="submit" value="Send" />
</form>

<form name="inForm" method="POST" action="">
  Receiving from AS:<br />
  <input type="text" name="inField">
</form>
```

   Este código crea dos formularios HTML similares a los creados en el entorno de Flash en el ejercicio anterior. El primer formulario envía el valor del campo de texto `outField` a la función `makeCall()` de JavaScript definida en un paso anterior. El segundo formulario se utiliza para mostrar un valor que se envía desde el archivo SWF cuando el usuario hace clic en la instancia `send_button`.

5. Guarde el documento HTML y cargue los archivos HTML y SWF en un servidor Web.
6. Visualice el archivo HTML en un navegador Web, introduzca una cadena en la instancia `out_ti` de TextInput y haga clic en el botón Enviar.

   Flash llama a la función `jsFunc()` de JavaScript y pasa el contenido del campo de texto `out_ti`, que muestra el contenido del campo de texto de entrada `inForm.inField` del formulario HTML.
7. Escriba un valor en el campo de texto HTML `outField` y haga clic en el botón Enviar.

   Flash llama a la función `asFunc()`, que muestra la cadena de la instancia `in_ti` de TextInput.

Encontrará el archivo de origen de muestra, ExtInt.fla, en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\ExternalAPI\simple example.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/ExternalAPI/simple example.

Para ver un ejemplo más complejo en el que se utiliza la API externa, consulte "Control de Flash Video con la API externa" en la página 709. Para más información sobre la seguridad de archivos locales, consulte "Seguridad de archivos local y Flash Player" en la página 717.

**NOTA**

Evite utilizar otros métodos de acceso al objeto plug-in como, por ejemplo, `document.getElementById("pluginName")` o `document.all.pluginName`, ya que estos otros métodos no funcionan de forma coherente en todos los navegadores.

# Control de Flash Video con la API externa

El siguiente procedimiento muestra cómo controlar archivos Flash Video (FLV) mediante controles en una página HTML y ofrece información sobre el vídeo en un campo de texto HTML. Este procedimiento utiliza la API externa para conseguir esta funcionalidad.

### Para crear una aplicación de Flash mediante la API externa:

1. Cree un nuevo documento de Flash y guárdelo como **video.fla**.
2. Añada un nuevo símbolo de vídeo a la biblioteca; para ello, seleccione Nuevo vídeo en el menú emergente del panel Biblioteca.
3. Arrastre el símbolo de vídeo al escenario y asígnele el nombre de instancia **selected_video**.
4. Seleccione la instancia `selected_video` y, a continuación, el inspector de propiedades para cambiar el tamaño de la instancia a **320** píxeles de anchura y **240** de altura.
5. Establezca las coordenadas *x* e *y* para la posición del vídeo en **0**.
6. Seleccione el escenario y utilice el inspector de propiedades para cambiar el tamaño de las dimensiones a **320** por **240** píxeles.

   Ahora el escenario coincide con las dimensiones de la instancia de vídeo.
7. Añada el siguiente código ActionScript al fotograma 1 de la línea de tiempo principal:

```
import flash.external.ExternalInterface;

/* Registrar playVideo() y pauseResume() para que se las pueda
llamar desde JavaScript en la página HTML del contenedor. */
ExternalInterface.addCallback("playVideo", null, playVideo);
ExternalInterface.addCallback("pauseResume", null, pauseResume);

/* El vídeo requiere un objeto NetConnection y NetStream. */
var server_nc:NetConnection = new NetConnection();
server_nc.connect(null);
var video_ns:NetStream = new NetStream(server_nc);

/* Asociar el objeto NetStream al objeto de vídeo en el escenario para
que los datos de NetStream se muestren en el objeto de vídeo. */
selected_video.attachVideo(video_ns);

/* Se llama automáticamente al método onStatus() cuando se actualiza el
  estado del
objeto NetStream (se inicia la reproducción del vídeo, por ejemplo).
Cuando eso ocurra, enviar el valor de la propiedad code a la página HTML
  mediante
una llamada a la función updateStatus() de JavaScript a través de
  ExternalInterface. */
video_ns.onStatus = function(obj:Object):Void {
  ExternalInterface.call("updateStatus", "  " + obj.code);
};
```

```
function playVideo(url:String):Void {
  video_ns.play(url);
}

function pauseResume():Void {
  video_ns.pause();
}
```

La primera parte de este código ActionScript define dos funciones callback de ExternalInterface: `playVideo()` y `pauseResume()`. Estas funciones se llaman desde JavaScript en el siguiente procedimiento. La segunda parte del código crea un nuevo objeto NetConnection y NetStream, que se utiliza con la instancia de vídeo para reproducir dinámicamente archivos FLV.

El código del siguiente procedimiento define un controlador de eventos `onStatus` para el objeto `video_ns` de NetStream. Cuando el objeto NetStream cambia su estado, Flash emplea el método `ExternalInterface.call()` para activar la función personalizada, `updateStatus()`. Las dos últimas funciones, `playVideo()` y `pauseResume()`, controlan la reproducción de la instancia de vídeo en el escenario. Estas dos funciones se llaman desde el código JavaScript escrito en el siguiente procedimiento.

8. Guarde el documento de Flash.
9. Seleccione Archivo > Configuración de publicación, elija la ficha Formatos y asegúrese de que HTML y Flash estén seleccionados.
10. Haga clic en Publicar para publicar los archivos SWF y HTML en el disco duro.

    Cuando haya terminado, continúe con el siguiente procedimiento para crear el contenedor del archivo SWF.

Encontrará el archivo de origen de muestra, external.fla, en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\ExternalAPI.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/ExternalAPI.

En el siguiente procedimiento, se modifica el código HTML generado por Flash en el procedimiento anterior. Este procedimiento crea el código JavaScript y HTML que se requiere para que los archivos FLV se reproduzcan en el archivo SWF.

### Para crear el contenedor del archivo SWF:

1. Complete el procedimiento anterior.
2. Abra el documento video.html que publicó en el último paso del procedimiento anterior.
3. Modifique el código existente para que coincida con el código siguiente:

| NOTA | Lea los comentarios de código del ejemplo siguiente. La información general de un código sigue este ejemplo de código. |
|---|---|

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
  www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"
  />
<title>ExternalInterface</title>

<script language="JavaScript">
  // Utilizar una variable para hacer referencia al archivo SWF
  incorporado.
  var flashVideoPlayer;

  /* Cuando se carga la página HTML (mediante el evento onLoad de la
  etiqueta <body>), ésta llama a la función initialize(). */
  function initialize() {
    /* Comprobar si el navegador es IE. Si es así, flashVideoPlayer es
window.videoPlayer. De lo contrario, será document.videoPlayer.
videoPlayer es el ID asignado a las etiquetas <object> y <embed>. */
    var isIE = navigator.appName.indexOf("Microsoft") != -1;
    flashVideoPlayer = (isIE) ? window['videoPlayer'] :
  document['videoPlayer'];
  }

  /* Cuando el usuario haga clic en el botón de reproducción del
  formulario, actualizar el área de texto videoStatus, llamar a la
  función playVideo() en el archivo SWF y pasarle la URL del archivo
  FLV. */
  function callFlashPlayVideo() {
    var comboBox = document.forms['videoForm'].videos;
    var video = comboBox.options[comboBox.selectedIndex].value;
    updateStatus("____" + video + "____");
    flashVideoPlayer.playVideo("http://www.helpexamples.com/flash/
  video/" + video);
  }

  // Llamar a la función pauseResume() del archivo SWF.
  function callFlashPlayPauseVideo() {
    flashVideoPlayer.pauseResume();
  }
```

```
    /* La función updateStatus() se llama desde el archivo SWF mediante el
    método onStatus() del objeto NetStream. */
    function updateStatus(message) {
      document.forms['videoForm'].videoStatus.value += message + "\n";
    }
</script>
</head>
<body bgcolor="#ffffff" onLoad="initialize();">

<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
  codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/
  swflash.cab#version=8,0,0,0" width="320" height="240" id="videoPlayer"
  align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="video.swf" />
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<embed src="video.swf" quality="high" bgcolor="#ffffff" width="320"
  height="240" name="videoPlayer" align="middle"
  allowScriptAccess="sameDomain" type="application/x-shockwave-flash"
  pluginspage="http://www.macromedia.com/go/getflashplayer" />
</object>

<form name="videoForm">
  Select a video:<br />
  <select name="videos">
    <option value="lights_long.flv">lights_long.flv</option>
    <option value="clouds.flv">clouds.flv</option>
    <option value="typing_long.flv">typing_long.flv</option>
    <option value="water.flv">water.flv</option>
  </select>
  <input type="button" name="selectVideo" value="play"
  onClick="callFlashPlayVideo();" />
  <br /><br />

  Playback <input type="button" name="playPause" value="play/pause"
  onClick="callFlashPlayPauseVideo();" />

  <br /><br />
  Video status messages <br />
  <textarea name="videoStatus" cols="50" rows="10"></textarea>
</form>

</body>
</html>
```

Este código HTML define cuatro funciones de JavaScript: `initialize()`, `callFlashPlayVideo()`, `callFlashPlayPauseVideo()` y `updateStatus()`. La función `initialize()` se llama dentro de la etiqueta `body` del evento `onLoad`. Las funciones `callFlashPlayVideo()` y `callFlashPlayPauseVideo()` se llaman cuando el usuario hace clic en el botón Reproducir o Reproducir/Pausar dentro del documento HTML y se activan las funciones `playVideo()` y `pauseResume()` del archivo SWF.

La llamada a la última función, `updateStatus()`, la realiza el archivo SWF cuando se activa el controlador de eventos `onStatus` del objeto de NetStream `video_ns`. Este código HTML también define un formulario que contiene un cuadro combinado de vídeos entre los que el usuario puede elegir. Cuando el usuario selecciona un vídeo y hace clic en el botón de reproducción, se llama a la función `callFlashPlayVideo()` de JavaScript que, a continuación, llama a la función `playVideo()` del archivo SWF. Esta función pasa la URL al archivo SWF para que se cargue en la instancia de vídeo. Según se reproduce el vídeo y se cambia el estado del objeto NetStream, se actualiza el contenido del área de texto HTML en el escenario.

4. Guarde los cambios del documento HTML y cargue los archivos HTML y SWF en un sitio Web.
5. Abra el documento remoto video.html desde el sitio Web, seleccione un vídeo del cuadro combinado y haga clic en el botón de reproducción.

   Flash reproduce el archivo FLV seleccionado y actualiza el contenido del área de texto videoStatus en el documento HTML.

Encontrará el archivo de origen de muestra, external.fla, en la carpeta Samples del disco duro.

- En Windows, desplácese a *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ActionScript\ExternalAPI.
- En Macintosh, desplácese a *Disco duro de Macintosh*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ActionScript/ExternalAPI.

Para más información sobre la API externa, consulte `%{ExternalInterface (flash.external.ExternalInterface)}%` en *Referencia del lenguaje ActionScript 2.0*.

Para más información sobre la seguridad de archivos locales, consulte "Seguridad de archivos local y Flash Player" en la página 717.

| NOTA | Evite utilizar otros métodos de acceso al objeto plug-in como, por ejemplo, document.getElementById("pluginName") o document.all.pluginName, ya que estos otros métodos no funcionan de forma coherente en todos los navegadores. |
|---|---|

CAPÍTULO 17

# Aspectos básicos de la seguridad

17

En Macromedia Flash Basic 8 y Macromedia Flash Professional 8, puede utilizar ActionScript para cargar datos de fuentes externas en un archivo SWF o enviar datos a un servidor. Al cargar datos en un archivo SWF, es necesario entender y adaptarse al modelo de seguridad de Flash 8. Cuando abra un archivo SWF en el disco duro, es posible que deba realizar configuraciones específicas para probar el archivo de forma local.

Para obtener información sobre la seguridad de archivos local, consulte "Seguridad de archivos local y Flash Player" en la página 717. Para obtener información sobre las diferencias entre el modelo de seguridad de Flash Player 7 y Flash Player 8, consulte "Compatibilidad con modelos de seguridad de Flash Player anteriores" en la página 716. Para obtener información sobre cómo cargar y analizar datos desde un servidor, lea el Capítulo 16, "Trabajo con datos externos", en la página 671. Para más información sobre la seguridad, consulte www.macromedia.com/devnet/security y www.macromedia.com/software/flashplayer/security/.

Para más información sobre la seguridad en Flash 8, consulte los temas siguientes:

# Compatibilidad con modelos de seguridad de Flash Player anteriores

Como resultado de los cambios de seguridad introducidos en Flash Player 7, el contenido que se ejecuta de la forma esperada en Flash Player 6 o versiones anteriores podría no hacerlo de la misma manera en las versiones posteriores de Flash Player. Por ejemplo, en Flash Player 6, un archivo SWF que reside en www.macromedia.com podría leer datos de un servidor situado en data.macromedia.com. Es decir, Flash Player 6 permitiría que un archivo SWF de un dominio cargase datos de un dominio similar.

En Flash Player 7 y versiones posteriores, si un archivo SWF de la versión 6 (o anterior) intenta cargar datos de un servidor que reside en otro dominio, y ese servidor no proporciona un archivo de política que permita la lectura desde el dominio de dicho archivo SWF, aparecerá el cuadro de diálogo Configuración de Macromedia Flash Player. El cuadro de diálogo solicita al usuario que permita o deniegue el acceso a datos de varios dominios.

Si el usuario hace clic en Permitir, el archivo SWF puede acceder a los datos solicitados; si hace clic en Denegar, el archivo SWF no podrá acceder a los datos solicitados.

Para impedir que aparezca este cuadro de diálogo, deberá crear un archivo de política de seguridad en el servidor que proporciona los datos. Para más información, consulte "Permiso de carga de datos de varios dominios" en la página 744.

Flash Player 7 y las versiones posteriores no permiten el acceso a varios dominios sin un archivo de política de seguridad.

Flash Player 8 ha cambiado la forma de gestionar `System.security.allowDomain`. Un archivo SWF de Flash 8 que llama a `System.security.allowDomain` con cualquier argumento o cualquier otro archivo SWF que utilice el valor de comodín (*), sólo permitirá el acceso a sí mismo. Ahora se admite un valor de comodín (*), por ejemplo: `System.security.allowDomain("*")` y `System.security.allowInsecureDomain("*")`. Si un archivo SWF de la versión 7 o anterior llama a `System.security.allowDomain` o `System.security.allowInsecureDomain` con un argumento que no sea un comodín (*), afectará a todos los archivos SWF de la versión 7 o inferior al llamar al dominio del archivo SWF, como se hizo en Flash Player 7. No obstante, este tipo de llamada no afecta a archivos SWF de Flash Player 8 (o versiones posteriores) al llamar al dominio del archivo SWF. Este procedimiento ayuda a reducir la separación de contenido antiguo en Flash Player.

Para más información, consulte "Dominios, seguridad entre dominios y archivos SWF" en la página 735, %{allowDomain (método security.allowDomain)}% y %{allowInsecureDomain (método security.allowInsecureDomain)}%.

Flash Player 8 no permite la comunicación de archivos SWF locales con Internet sin que se haya introducido previamente una configuración específica en el equipo. Supongamos que tiene contenido antiguo publicado antes de que comenzaran a aplicarse estas restricciones. Si dicho contenido intenta comunicarse con la red o el sistema de archivos local, o ambos, Flash Player 8 detiene la operación y deberá proporcionar permiso de manera explícita para que la aplicación funcione correctamente. Para más información, consulte "Seguridad de archivos local y Flash Player" en la página 717.

# Seguridad de archivos local y Flash Player

Flash Player 8 ha aplicado mejoras al modelo de seguridad, en el que no se permite a las aplicaciones Flash y archivos SWF de un equipo local comunicarse *a la vez* con Internet y con el sistema de archivos local de forma predeterminada. Un *archivo SWF local* es un archivo SWF instalado localmente en el equipo de un usuario, no un archivo disponible en un sitio Web, y no incluye archivos de proyectores (EXE).

| NOTA | Las restricciones que se describen en esta sección no afectan a los archivos SWF que están en Internet. |
|---|---|

Al crear un archivo FLA, puede indicar si se permite un archivo SWF para comunicarse con una red o con un sistema de archivos local. En versiones anteriores de Flash Player, los archivos SWF locales podían interactuar con otros archivos SWF y cargar datos desde cualquier ubicación local o remota. En Flash Player 8, el archivo SWF no puede realizar conexiones con el sistema de archivos local *e* Internet. Se trata de un cambio realizado en la seguridad, de modo que un archivo SWF no puede leer archivos en el disco duro y enviar el contenido de estos archivos a través de Internet.

Esta restricción de la seguridad afecta al contenido implementado localmente, ya sea contenido antiguo (un archivo FLA creado en una versión anterior de Flash) o creado en Flash 8. Supongamos que implementa una aplicación Flash, con Flash MX 2004 o anterior, que se ejecuta de forma local y también tiene acceso a Internet. En Flash Player 8, esta aplicación solicita ahora al usuario que dé su permiso para que pueda comunicarse con Internet.

Al probar un archivo en el disco duro, es necesario realizar una serie de pasos para determinar si el archivo está en un documento local de confianza o no. Si crea el archivo en el entorno de edición de Flash (por ejemplo, al seleccionar Control > Probar película), el archivo es de confianza porque se encuentra en un entorno de prueba.

En Flash Player 7 y versiones anteriores, los archivos SWF locales tenían permiso para leer de un sistema de archivos local y la red (como Internet). En Flash Player 8, los archivos SWF locales tienen los niveles de permiso siguientes:

**Acceso sólo al sistema de archivos local (nivel predeterminado)** Un archivo SWF local puede leer de las rutas de red de la convención de asignación de nombres universal (UNC) y del sistema de archivos local pero no se puede comunicar con Internet. Para más información sobre archivos SWF de acceso a archivos locales, consulte "Acceso sólo a archivos locales (nivel predeterminado)" en la página 726.

**Acceso sólo a la red** Un archivo SWF puede acceder a la red (como Internet) pero no al sistema de archivos local donde está instalado. Para más información sobre archivos SWF sólo de red, consulte "Acceso sólo a la red" en la página 726.

**Acceso al sistema de archivos local y a la red** Un archivo SWF local puede leer del sistema de archivos local en el que está instalado, leer y escribir en y desde servidores, así como usar scripts entre películas de otros archivos SWF de la red o el sistema de archivos local. Estos archivos son de confianza y se comportan de la misma manera que en Flash Player 7. Para más información sobre archivos SWF de acceso local y a la red, consulte "Acceso a la red y al sistema de archivos" en la página 727.

Para más información sobre la seguridad de los archivos locales en Flash 8 en lo que respecta a la herramienta de edición, consulte las secciones siguientes:

- "Aspectos básicos de los entornos limitados de seguridad local" en la página 719
- "Configuración de seguridad de Flash Player" en la página 720
- "Seguridad de archivo local y archivos de proyectores" en la página 722
- "Solución de problemas con archivos SWF antiguos" en la página 723
- "Reparación de contenido antiguo implementado en equipos locales" en la página 723
- "Publicación de archivos para implementación local" en la página 724

Para obtener información sobre la seguridad de archivos locales para los usuarios, consulte "Configuración de seguridad de Flash Player" en la página 720. Para más información sobre la seguridad, consulte www.macromedia.com/devnet/security/ y www.macromedia.com/software/flashplayer/security/.

# Aspectos básicos de los entornos limitados de seguridad local

Existen varios entornos limitados (sandboxes) de seguridad en Flash Player. Cada uno determina cómo un archivo SWF puede interactuar con el sistema de archivos local, la red o tanto el sistema de archivos local como la red al mismo tiempo. Al restringir la forma en que un archivo puede interactuar con el sistema de archivos local o la red, se mantiene la seguridad del equipo y los archivos. Conocer los aspectos básicos de los entornos limitados de seguridad contribuye a desarrollar y probar aplicaciones Flash en el equipo sin encontrar errores inesperados.

## Local con sistema de archivos

Por razones de seguridad, Flash Player 8 coloca, de forma predeterminada, todos los archivos SWF locales, incluidos todos los antiguos, en el entorno limitado local con sistema de archivos (a menos que se aplique otra configuración). En algunos archivos SWF antiguos (anteriores a Flash Player 8), podrían verse afectadas las operaciones al aplicarse restricciones en su acceso (sin acceso a la red externo), aunque así se proporciona el valor predeterminado más seguro para garantizar la protección de los usuarios.

Desde este entorno limitado, los archivos SWF pueden leer de archivos de sistemas de archivos local o rutas de red UNC (mediante el método `XML.load()`, por ejemplo), pero no pueden comunicarse con la red de ninguna forma. De este modo, el usuario tiene la seguridad de que no se filtran los datos locales a la red ni se comparten de forma indebida.

## Local con acceso a la red

Cuando se asignan archivos SWF locales al entorno limitado local con acceso a la red, estos archivos pierden el acceso al sistema de archivos local. A cambio, se les permite a acceder a la red. Sin embargo, sigue sin permitirse que un archivo SWF local con acceso a la red lea los datos derivados de la red a menos que se disponga de permisos para ello. Por lo tanto, un archivo SWF local con acceso a la red no tiene acceso local, aunque tiene la capacidad de transmitir datos a través de la red y puede leer datos de la red desde esos sitios que designan permisos de acceso específicos al sitio.

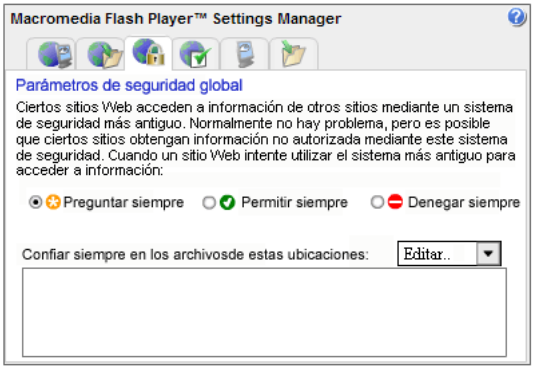
## Local de confianza

Los archivos SWF asignados al entorno limitado local de confianza pueden interactuar con el resto de archivos SWF y cargar datos desde cualquier lugar (remoto o local).

# Configuración de seguridad de Flash Player

Macromedia ha diseñado Flash Player para proporcionar una configuración de seguridad en la que no es preciso permitir o denegar el acceso en la mayoría de las situaciones. En ocasiones, podría encontrar contenido Flash antiguo que se creó mediante reglas de seguridad más antiguas para Flash Player 7 o versiones anteriores. En estos casos, Flash Player puede permitir que el contenido funcione como determinó el desarrollador, con reglas de seguridad más antiguas, o bien aplicar reglas más nuevas, más estrictas. La última opción garantiza que sólo verá o reproducirá contenido que cumpla las últimas normas de seguridad, pero en ocasiones puede dificultar el funcionamiento del contenido de Flash más antiguo.

Todos los usuarios que vean archivos SWF (incluidos desarrolladores que no sean de Flash) pueden establecer permisos globalmente a través del panel Parámetros de seguridad global del Administrador de configuración de Flash Player (como se muestra en la siguiente figura).




Cuando se utiliza contenido antiguo en una versión más reciente del reproductor, y Flash Player debe tomar una decisión sobre la aplicación de reglas más recientes, es posible que aparezca uno de los siguientes cuadros de diálogo. Estos cuadros de diálogo piden permiso antes de permitir que el contenido antiguo de Flash se comunique con otras ubicaciones de Internet:

- Es posible que aparezca un cuadro de diálogo para avisarle de que el contenido de Flash que utiliza está intentando usar reglas de seguridad anteriores para acceder a información de un sitio que no pertenece a su propio dominio y que dicha información podría compartirse en dos sitios. Flash Player le pregunta si desea permitir o denegar este acceso.

  Además de responder al cuadro de diálogo, puede utilizar el panel Parámetros de seguridad global para especificar si Flash Player debería pedir siempre permiso, a través del cuadro de diálogo, antes de permitir el acceso; denegar siempre el acceso, sin preguntar primero; o bien permitir el acceso a otros sitios o dominios sin preguntar.

- (Sólo Flash Player 8) Podría aparecer un cuadro de diálogo en el que se le avisa de que un archivo SWF está intentando comunicarse con Internet. De forma predeterminada, Flash Player 8 no permite que el contenido local de Flash se comunique con Internet.


  Haga clic en Configuración para acceder al panel Parámetros de seguridad global, donde puede especificar que algunas aplicaciones Flash de su equipo se comuniquen con Internet.

Para cambiar la configuración de seguridad u obtener más información sobre sus opciones, consulte el panel Parámetros de seguridad global. Utilice este panel para restablecer los parámetros de privacidad de Macromedia Flash Player:

- Si selecciona *Denegar siempre* y, a continuación, confirma su selección, se denegará el acceso a los sitios Web que intenten utilizar la cámara o el micrófono. No se le preguntará de nuevo si un sitio Web puede utilizar la cámara o el micrófono. Esta acción se aplica tanto a los sitios Web que ha visitado como a los que no.
- Si selecciona *Preguntar siempre* y, a continuación, confirma su selección, los sitios Web que intenten utilizar la cámara o el micrófono deberán pedir permiso. Esta acción se aplica tanto a los sitios Web que ha visitado como a los que no.

Si anteriormente ha seleccionado Recordar en el panel Privacidad (consulte la siguiente figura) para permitir o denegar de forma permanente el acceso a uno o más sitios Web, al seleccionar Preguntar siempre o Denegar siempre se anulará la selección de la opción Recordar para todos estos sitios Web. En otras palabras, la selección que efectúe aquí sustituirá a las selecciones realizadas anteriormente en el panel Configuración de privacidad.

Tras seleccionar Preguntar siempre o Denegar siempre (o en lugar de seleccionarlo), puede especificar los parámetros de privacidad de determinados sitios Web que haya visitado. Por ejemplo, puede seleccionar Denegar siempre aquí, luego utilizar el panel Configuración de privacidad de sitios y, a continuación, seleccionar Permitir siempre para los sitios Web que conozca y sean de su confianza.

Para los datos locales y el contenido implementado de forma local, los usuarios cuentan con otra opción: Podrán especificar los archivos SWF que puede que accedan a Internet a través del panel Parámetros de seguridad global. Para más información sobre la especificación de valores de configuración en el panel Parámetros de seguridad global, consulte "Especificación de archivos de confianza a través del Administrador de configuración" en la página 728. Para más información sobre el panel Parámetros de seguridad global, consulte www.macromedia.com/support/documentation/es/flashplayer/help/settings_manager04a.html.

| NOTA | Las opciones seleccionadas por los usuarios en el panel Parámetros de seguridad global tienen prioridad sobre cualquier otra decisión tomada en el cuadro de diálogo emergente de seguridad. |
|---|---|

## Seguridad de archivo local y archivos de proyectores

Los archivos de proyectores y los archivos SWF incluidos en estos o cargados en el proyecto durante el tiempo de ejecución no se ven afectados por las restricciones de seguridad de archivos locales, ya que el usuario final debe ejecutar el ejecutable para utilizar el archivo SWF. No hay cambios en la seguridad y los archivos de proyectores en Flash Player 8; el nivel de acceso y seguridad es el mismo que en las versiones anteriores de Flash Player.

Recuerde que los usuarios se muestran a menudo cautos a la hora de ejecutar archivos de proyectores. Un archivo de proyector es un ejecutable EXE o una aplicación Macintosh, por lo que los usuarios deben tener especial cuidado al ejecutar estos archivos en sus equipos. Si distribuye una aplicación mediante archivos de proyectores, puede que algunos usuarios no la instalen.

Además, un archivo de proyector incorpora una versión específica de Flash Player dentro del proyector, que podría ser anterior a la última versión de Flash Player disponible para descarga en el sitio Web de Macromedia. El Flash Player incorporado con el archivo de proyector podría ser una versión antigua si el proyector se creó con una versión anterior de Flash, o si se lanzó una edición de Flash Player tras la versión actual de la herramienta de edición de Flash. Por estos motivos, se recomienda distribuir aplicaciones mediante archivos SWF siempre que sea posible.

## Solución de problemas con archivos SWF antiguos

Algunos archivos FLA y SWF antiguos (creados con Flash MX 2004 y versiones anteriores) podrían no funcionar al probar o implementarlos de forma local (en un disco duro) debido a los cambios de seguridad en Flash 8. Esto podría ocurrir cuando un archivo SWF intenta acceder a sitios Web fuera de su dominio, caso en el que se hace necesaria la implementación de un archivo de política de varios dominios.

Podría tener archivos FLA o SWF creados en Flash MX 2004 o versiones anteriores que se han distribuido a usuarios que no utilizan la herramienta de edición de Flash 8 pero que han actualizado a Flash Player 8. Si el contenido antiguo implementado o probado localmente (un archivo SWF antiguo en un disco duro del usuario) no funciona porque intenta comunicarse con Internet cuando reproduce en Flash Player 8, deberá confiar en que los usuarios se fíen de forma explícita de su contenido para que éste se reproduzca correctamente (haciendo clic en un botón de un cuadro de diálogo).

Para aprender a reparar el contenido antiguo a fin de reproducirlo en un equipo local, consulte "Reparación de contenido antiguo implementado en equipos locales" en la página 723.

## Reparación de contenido antiguo implementado en equipos locales

Si ha publicado archivos SWF para Flash Player 7 o versiones anteriores que se han implementado en equipos locales y que se comunican con Internet, los usuarios deberán permitir expresamente la comunicación con Internet. Los usuarios pueden evitar que el contenido deje de funcionar al añadir la ubicación del archivo SWF en su equipo local al entorno limitado local de confianza en el Administrador de configuración.

### Para reparar archivos SWF de forma que se puedan reproducir localmente, utilice una de las siguientes opciones:

**Volver a implementar** Ejecute el Actualizador de contenido local. Este actualizador vuelve a configurar el archivo SWF para hacerlo compatible con el modelo de seguridad de Flash Player 8. El archivo SWF local se reconfigura de forma que sólo pueda acceder a la red o al sistema de archivos local. Para más información y descargar el Actualizador de contenido local, consulte www.macromedia.com/support/flashplayer/downloads.html.

**Volver a publicar e implementar** Vuelva a publicar el archivo con Flash Basic 8 o Flash Professional 8. La herramienta de edición requiere que se especifique en el cuadro de diálogo Configuración de publicación si un archivo SWF local puede acceder a la red o al sistema de archivos local, pero no a ambos. Si especifica que un archivo SWF local pueda acceder a la red, también deberá activar los permisos de dicho archivo (y de todos los archivos SWF locales) en los archivos SWF, HTML, de datos y/o de servidor a los que acceda. Para más información, consulte "Publicación de archivos para implementación local" en la página 724.

**Implementar nuevo contenido** Utilice un archivo de configuración (.cfg) de la carpeta #Security/FlashPlayerTrust. Puede utilizar este archivo para establecer los permisos de acceso local y a la red. Para más información, consulte "Creación de archivos de configuración para el desarrollo en Flash" en la página 730.

| NOTA | Estas opciones exigen que se vuelva a publicar o implementar el archivo SWF. |
|---|---|

# Publicación de archivos para implementación local

Es posible que envíe sus archivos SWF o FLA de Flash 8 a un usuario para que los pruebe o apruebe y, por ello, necesita que la aplicación tenga acceso a Internet. Si el documento se reproduce en un sistema local pero accede a archivos de Internet (por ejemplo, cargando XML o enviando variables), el usuario podría necesitar un archivo de configuración para que el contenido funcione correctamente, o bien podría necesitar configurar el archivo FLA de forma que el archivo SWF que publica pueda acceder a la red. Otra alternativa consistiría en establecer un archivo de configuración en el directorio FlashPlayerTrust. Para más información sobre el establecimiento de archivos de configuración, consulte "Creación de archivos de configuración para el desarrollo en Flash" en la página 730.

Utilice Flash Basic 8 o Flash Professional 8 para crear contenido de implementación local que funcione con la seguridad de archivos locales de Flash Player 8. En Configuración de publicación de Flash 8, deberá especificar si el contenido local puede acceder a la red o al sistema de archivos local, pero no a ambos.

Puede establecer niveles de permiso para un archivo FLA en el cuadro de diálogo Configuración de publicación. Estos niveles afectan a la reproducción local del archivo FLA, cuando se reproduce localmente en un disco duro.

| NOTA | Si especifica el acceso a la red de un archivo local, también debe activar los permisos en los archivos SWF, HTML, de datos y de servidor a los que acceda el archivo SWF local. |
|---|---|

**Archivos SWF de red** Los archivos SWF que se descargan de una red (como un servidor en línea) se colocan en un *entorno limitado* independiente que corresponde a los dominios exclusivos de origen de su sitio Web. Los archivos SWF locales que especifican que tienen acceso a la red se ubican en el entorno limitado *local con acceso a la red*. De forma predeterminada, estos archivos pueden leer datos desde sólo el sitio en el que se originaron. La coincidencia de dominio exacto se aplica a estos archivos. Los archivos SWF de red pueden acceder a los datos de otros dominios si disponen de los permisos adecuados. Para más información sobre archivos SWF de red, consulte "Acceso sólo a la red" en la página 726.

**Archivos SWF locales** Los archivos SWF que operan con sistemas de archivos locales o rutas de red UNC se colocan en uno de los tres entornos limitados en Flash Player 8. De forma predeterminada, los archivos SWF locales se ubican en el entorno limitado *local con sistema de archivos*. Los archivos SWF locales que se registran como de confianza (a través de un archivo de configuración) se colocan en el entorno limitado *local de confianza*. Para obtener información sobre los tres entornos limitados, consulte "Acceso sólo a archivos locales (nivel predeterminado)" en la página 726.

Para más información sobre el entorno limitado de seguridad, consulte "Aspectos básicos de los entornos limitados de seguridad local" en la página 719.

Los dos primeros niveles de permiso se establecen en el entorno de edición de Flash y el tercero mediante el panel Parámetros de seguridad global o el archivo FlashAuthor.cfg. El siguiente ejemplo muestra las opciones que están disponibles cuando se publica un archivo para probarlo en el disco duro local.

### Para publicar un documento con un nivel de permiso especificado:

1. Abra el archivo FLA para el que desea especificar un nivel de permiso.
2. Seleccione Archivo > Configuración de publicación > Flash.
3. Busque el cuadro de diálogo Seguridad de reproducción local y seleccione una de las siguientes opciones del menú emergente:
   - Acceder sólo a archivos locales (consulte "Acceso sólo a archivos locales (nivel predeterminado)")
   - Acceder sólo a la red (consulte "Acceso sólo a la red")
4. Haga clic en Aceptar para continuar editando el archivo FLA o en Publicar para crear el archivo SWF.

Para más información sobre los niveles de permiso que puede establecer para sus aplicaciones, consulte "Acceso sólo a archivos locales (nivel predeterminado)" en la página 726, "Acceso sólo a la red" en la página 726 y "Acceso a la red y al sistema de archivos" en la página 727.

## Acceso sólo a archivos locales (nivel predeterminado)

Para establecer este nivel de permiso, seleccione Configuración de publicación > Flash y, a continuación, elija Acceder sólo a archivos locales desde el menú emergente Seguridad de reproducción local. Este nivel permite a un archivo SWF local acceder sólo al sistema de archivos local donde se ejecuta. El archivo SWF puede leer de archivos conocidos del disco local sin restricción alguna. Sin embargo, las siguientes restricciones se aplican a la aplicación que accede a la red:

- El archivo SWF no puede acceder a la red de ninguna forma. El archivo SWF no puede usar scripts de otros archivos SWF de la red ni permitir que los archivos SWF de la red usen sus scripts.
- El archivo SWF no puede comunicarse con archivos SWF locales que sólo tengan permiso para acceder a la red ni con páginas HTML. No obstante, en algunos casos se permite la comunicación, por ejemplo si el HTML es de confianza y `allowScriptAccess` se ha establecido en `always` o si `allowScriptAccess` no se ha establecido y el archivo SWF es de Flash Player 7 o versiones anteriores.

## Acceso sólo a la red

Para establecer este nivel de permiso, seleccione Configuración de publicación > Flash y, a continuación, seleccione Acceder sólo a la red en el menú emergente Seguridad de reproducción local. Los archivos SWF locales con acceso a la red pueden leer de un servidor si dicho servidor contiene un archivo de política de varios dominios con `<allow-access-from-domain= “*”>`. Los archivos SWF locales con acceso a la red pueden usar scripts de otros archivos SWF si los otros archivos SWF, aquellos a los que se accede, contienen `System.Security.allowDomain(“*”)`. Los archivos SWF de red pueden usar scripts de un archivo SWF local con acceso a la red si este archivo contiene `allowDomain(“*”)`. El archivo SWF nunca podrá leer de archivos locales. En algunos casos, el tipo de archivo SWF afecta al acceso. Para obtener información, consulte %{allowDomain (`método security.allowDomain`)}% en *Referencia del lenguaje ActionScript 2.0*.

El valor de comodín (*) indica que se permite el acceso a *todos* los dominios, incluidos los servidores locales. Asegúrese de que desea permitir este amplio nivel de acceso antes de utilizar el argumento de comodín.

Sin estos permisos, los archivos SWF locales con acceso a la red sólo se pueden comunicar con otros archivos SWF locales con acceso a la red y pueden enviar datos a los servidores (mediante `XML.send()`, por ejemplo). En algunos casos, se permite el acceso si el archivo HTML es de confianza.

### Acceso a la red y al sistema de archivos

Se trata del nivel de permiso más alto. Un archivo SWF local que tenga estos permisos es un *archivo SWF local de confianza*. Los archivos SWF locales de confianza pueden leer de otros archivos SWF locales, interactuar con cualquier servidor y escribir código ActionScript para otros archivos SWF o archivos HTML que no han prohibido de forma explícita el permiso para archivos (por ejemplo, con `allowScriptAccess="none"`). El usuario o el desarrollador de Flash puede conceder este nivel de permiso de las siguientes maneras:

- Mediante el panel Parámetros de seguridad global en el Administrador de configuración
- Mediante un archivo de configuración global

Se puede instalar un archivo de configuración con el archivo SWF, creado por un desarrollador de Flash o añadido por un administrador (para todos los usuarios o el usuario actual) o cualquier desarrollador de Flash (para el usuario actual).

Para más información sobre archivos de configuración y el panel Parámetros de seguridad global, consulte "Configuración de seguridad de Flash Player" en la página 720 y "Especificación de archivos de confianza a través del Administrador de configuración" en la página 728 y "Creación de archivos de configuración para el desarrollo en Flash" en la página 730.

## Comprobación local de contenido con restricciones de seguridad de archivos locales de Flash 8

Como desarrollador de Flash, suele probar aplicaciones de Flash localmente, por lo que podría ver un mensaje de cuadro de diálogo que aparece cuando la aplicación de Flash local intenta comunicarse con Internet. Este cuadro de diálogo podría aparecer al probar un archivo SWF en Flash Player si el archivo SWF no dispone de acceso a la red. Para más información sobre la publicación de archivos SWF con niveles de permiso especificados, consulte "Publicación de archivos para implementación local" en la página 724. La publicación de un archivo SWF con una de estas opciones significa que se puede comunicar con la red *o* el sistema de archivos local.

En algunas ocasiones, podría ser necesario comunicarse con el sistema de archivos local y la red al probar un documento. Dado que el nuevo modelo de seguridad podría interrumpir el flujo de trabajo al editar aplicaciones Flash, puede utilizar el panel Parámetros de seguridad global del Administrador de configuración de Flash Player para especificar las aplicaciones Flash del equipo que se pueden comunicar siempre tanto con Internet como con el sistema de archivos local. También se puede modificar el archivo de configuración para especificar directorios de confianza en el disco duro.

Para más información, consulte las siguientes secciones:

- "Especificación de archivos de confianza a través del Administrador de configuración" en la página 728
- "Creación de archivos de configuración para el desarrollo en Flash" en la página 730

## Especificación de archivos de confianza a través del Administrador de configuración

Puede especificar qué contenido de Flash instalado en su equipo puede utilizar siempre las reglas de seguridad antiguas añadiendo la ubicación correspondiente al panel Parámetros de seguridad global del Administrador de configuración de Flash Player, Tras añadir una ubicación de su equipo al panel Seguridad, el contenido de esa ubicación pasa a ser *de confianza*. Flash Player no pedirá permiso y está autorizado a utilizar siempre las reglas de seguridad antiguas, aunque esté seleccionado Denegar siempre en el panel Seguridad. La lista Confiar siempre en los archivos de estas ubicaciones tiene prioridad sobre las opciones del panel Configuración. Es decir, aunque haya seleccionado denegar siempre al contenido local y Web el derecho a utilizar las reglas de seguridad antiguas, los archivos locales de su lista de confianza podrán utilizar siempre las reglas antiguas.

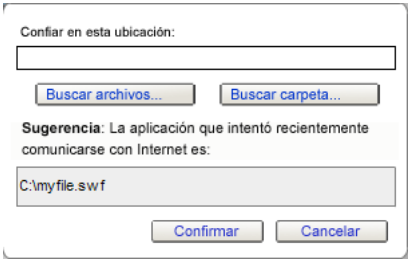
La lista Confiar siempre en los archivos de la parte inferior del panel se aplica específicamente al contenido de Flash que ha descargado en el equipo; no al contenido que utiliza mientras visita un sitio Web.

En el siguiente ejemplo se muestra cómo especificar que un archivo SWF local se pueda comunicar con Internet. Cuando se prueba localmente un archivo en un navegador (Archivo > Vista previa de publicación > HTML), podría aparecer un cuadro de diálogo de seguridad. Si hace clic en Configuración, aparecerá el panel Parámetros de seguridad global del Administrador de configuración.

### Para especificar que un archivo SWF local se pueda comunicar con Internet y el sistema de archivos local:

1. En el panel Parámetros de seguridad global, haga clic en el menú emergente y seleccione Agregar.

   Se abre un cuadro de diálogo para añadir una ubicación.




   Si ha llegado al Administrador de configuración desde el botón Configuración en un cuadro de diálogo, el cuadro Agregar contiene una ruta parecida a C:*nombredirectorio**nombrearchivo.swf* o */Usuarios/nombredirectorio/nombrearchivo.swf*; esta ruta le indica el archivo que ha intentado comunicarse con Internet y que la seguridad de Flash Player ha bloqueado. Si la ruta incluye el contenido que desea que se comunique con Internet, copie y péguela en el cuadro Confiar en esta ubicación. También puede hacer clic en los botones Examinar y localizar el contenido.

   Puede añadir un archivo o un directorio completo. Si añade un directorio completo, todos los archivos y subdirectorios de ese directorio serán de confianza. En ocasiones, el contenido de Flash se compone de varios archivos relacionados y es posible que deba confiar en el directorio completo donde están ubicados todos los archivos relacionados. En general, evite confiar en directorios principales.

2. Haga clic en Confirmar.

   La ubicación se añadirá al panel de configuración de seguridad. Las ubicaciones que aparecen pueden utilizar siempre reglas de seguridad antiguas, aunque estén seleccionadas las opciones Preguntar siempre o Denegar siempre en la parte superior del panel de seguridad.

   Una vez haya añadido ubicaciones de confianza, reinicie el contenido de Flash local actualizando el navegador o reiniciando el reproductor.

Si hace clic en Preguntar siempre, sólo se aplica esa configuración para permitir siempre contenido antiguo (Flash Player 7 y versiones anteriores). La configuración no “siempre permite” contenido de Flash Player 8. Se aconseja especificar las aplicaciones Flash y los directorios del equipo que pueden comunicarse con Internet y el sistema de archivos local.

## Creación de archivos de configuración para el desarrollo en Flash

La herramienta de edición de Flash 8 establece un indicador en el disco duro que lo identifica como desarrollador y lo dirige a una versión específica para desarrolladores del panel Parámetros de configuración global en lugar de este mismo panel pero dirigido al usuario. Este indicador se encuentra en el archivo FlashAuthor.cfg del disco duro, que se instala automáticamente durante la instalación de la herramienta de edición de Flash Basic 8 y Flash Professional 8.

El archivo FlashAuthor.cfg se ubica en los siguientes directorios aproximados:

**Windows** *disco de inicio*\Documents and Settings\<*Nombredeusuario*>\Application Data\Macromedia\Flash Player\#Security

**Macintosh** /Users/<*Nombredeusuario*>/Library/Preferences/Macromedia/Flash Player/#Security/

De forma predeterminada, este archivo se establece en `LocalSecurityPrompt=Author`, lo que significa que las advertencias que ve en el equipo lo consideran un desarrollador de Flash a diferencia de un usuario que no tiene la herramienta de edición instalada.

Puede probar aplicaciones locales como usuario final y ver los cuadros de diálogo de advertencia que podría encontrar un usuario final. Para ello, abra FlashAuthor.cfg en un editor de texto y cambie LocalSecurityPrompt en el archivo FlashAuthor.cfg para que coincida con lo siguiente:

```
LocalSecurityPrompt=User
```

Es posible que desee proporcionar un archivo FlashAuthor.cfg, con `LocalSecurityPrompt` establecido en `Autor`, a otros desarrolladores durante el proceso de diseño o de desarrollo, o bien a los usuarios que prueban aplicaciones Flash en el disco duro local y no tienen instalada la herramienta de edición de Flash 8. De este modo, podrá reproducir la experiencia del usuario final con el contenido implementado de forma local.

| NOTA | Si se elimina el archivo FlashAuthor.cfg, se vuelve a crear el archivo al iniciar la herramienta de edición de Flash 8. |
|---|---|

En el directorio #Security del disco duro, se puede crear un directorio FlashPlayerTrust en el que se pueden almacenar archivos de configuración exclusiva. Dentro de estos archivos, puede especificar directorios o aplicaciones que confíen en el disco duro. Este directorio no requiere acceso administrativo, por lo que los usuarios sin permisos administrativos pueden establecer permisos para archivos SWF y probar aplicaciones.

Si no especifica un directorio, el contenido podría no funcionar de la forma esperada. Los archivos de configuración de un directorio FlashPlayerTrust contienen rutas de directorio. El archivo puede incluir una lista de distintos directorios y se puede añadir nuevas rutas al archivo. Flash Player espera una ruta por línea en los archivos de configuración. Cualquier línea que comience por el signo # (sin espacio delante del mismo) se tratará como un comentario.

### Para crear un archivo de configuración para confiar en un directorio:

1. Busque la carpeta #Security en el disco duro.
2. Cree una carpeta denominada **FlashPlayerTrust** en la carpeta #Security.
3. Cree un nuevo archivo en el directorio FlashPlayerTrust con un editor de texto y guárdelo como **myTrustFiles.cfg**.

   Puede utilizar cualquier nombre exclusivo para el archivo de configuración.
4. Busque el directorio en el que prueba las aplicaciones Flash.
5. Escriba o pegue cada ruta de directorio (cualquier ruta de directorio en el disco duro) en una nueva línea del archivo. Puede pegar varias rutas de directorio en distintas líneas. Al terminar, el archivo presentará un aspecto similar al siguiente ejemplo:

   ```
   C:\Documents and Settings\<sunombre>\Mis documentos\files\
   C:\Documents and Settings\<sunombre>\Mis documentos\testapps\
   ```
6. Guarde los cambios en myTrustFiles.cfg.
7. Pruebe un documento que tenga acceso a los archivos de red y locales del directorio que ha añadido al archivo.

   Las aplicaciones Flash guardadas en este directorio pueden acceder ahora a los archivos locales y a la red.

Puede haber numerosas rutas de directorio guardadas en cada archivo de configuración y numerosos archivos *.cfg guardados en el directorio FlashPlayerTrust.

Si crea aplicaciones que se instalan en un disco duro de usuario final, podría ser necesario crear un archivo de configuración en FlashPlayerTrust para especificar un directorio de confianza para la aplicación. Puede crear archivos de configuración en el directorio FlashPlayerTrust que especifiquen la ubicación de la aplicación de confianza. Consulte el procedimiento anterior para obtener información sobre este directorio y crear archivos de configuración.

| NOTA | Un usuario ejecuta un instalador con un permiso administrativo en un equipo. |
|---|---|

Debe desarrollar un esquema de designación de nombres exclusivo para evitar conflictos con otras aplicaciones que podrían instalar archivos en este directorio. Por ejemplo, podría ser conveniente utilizar el nombre del software y de la empresa en el nombre de archivo para evitar conflictos.

| SUGERENCIA | Si no desea utilizar archivos de configuración, podría publicar sus aplicaciones Flash en un servidor de prueba independiente en lugar de proporcionar a los clientes u otros desarrolladores archivos SWF que se ejecuten en sus discos duros locales. |
|---|---|

Para más información sobre los archivos de configuración, consulte www.macromedia.com/go/flashauthorcfg. También puede crear un archivo de configuración exclusiva para confiar en uno o más directorios. Para obtener información detallada sobre seguridad, consulte www.macromedia.com/devnet/security/ y www.macromedia.com/software/flashplayer/security/.

## Propiedad sandboxType

La propiedad `System.security.sandboxType` de Flash Player 8 devuelve el tipo de entorno local de seguridad en el que opera el archivo SWF que realiza la llamada.

La propiedad `sandboxType` presenta uno de los cuatro valores siguientes:

**remote** El archivo SWF se aloja en Internet y opera bajo reglas de entorno local basadas en dominio.

**localTrusted** El archivo SWF es un archivo local al que el usuario ha dado su confianza, ya sea mediante el panel Parámetros de seguridad global del Administrador de configuración o mediante un archivo de configuración FlashPlayerTrust. El archivo SWF puede leer tanto de fuentes de datos locales como comunicarse con la red (como Internet).

**localWithFile** El archivo SWF es un archivo local para el que el usuario no ha dado su confianza y no se ha publicado con una designación de acceso a la red. El archivo SWF puede leer de fuentes de datos locales pero no puede comunicarse con la red (como Internet).

**localWithNetwork** El archivo SWF es un archivo local para el que el usuario no ha dado su confianza y se ha publicado con la opción Acceder sólo a la red seleccionada en el cuadro de diálogo Configuración de publicación (ficha Flash). El archivo SWF se puede comunicar con la red pero no puede leer de fuentes de datos locales.

Se puede comprobar la propiedad `sandboxType` desde cualquier archivo SWF, aunque sólo se devolverá un valor en archivos publicados para Flash Player 8. Esto significa que cuando se publica para Flash Player 7 o versiones anteriores, no se sabrá si se admite la propiedad `sandboxType` durante la ejecución. En el caso de que no se admita, el valor es `undefined`, lo que tiene lugar cuando la versión de Flash Player (indicada por la propiedad `System.capabilities.version`) es inferior a 8. Si el valor es `undefined`, se puede determinar el tipo de entorno local acorde independientemente de que la URL del archivo SWF sea un archivo local o no. Si el archivo SWF es un archivo local, Flash Player lo clasifica como `localTrusted` (que es como se trató todo el contenido local antes de Flash Player 8); de lo contrario Flash Player clasifica el archivo SWF como `remote`.

## Restricciones de archivos locales con sistema de archivos

No se ha registrado un archivo local con sistema de archivos a través del archivo de configuración del directorio FlashPlayerTrust, en el panel Parámetros de seguridad global del Administrador de configuración, o no se ha otorgado permiso de red en el cuadro de diálogo Configuración de publicación del entorno de edición de Flash.

NOTA

Para obtener información sobre los entornos limitados de seguridad, consulte "Aspectos básicos de los entornos limitados de seguridad local" en la página 719.

Estos archivos incluyen contenido antiguo que se reproduce en Flash Player 8. Si está desarrollando contenido en Flash 8, o tiene contenido que entra dentro de una de las siguientes categorías, usted (o sus usuarios) deberían registrar el archivo como de confianza. Para obtener información sobre el registro de un archivo como de confianza, consulte "Especificación de archivos de confianza a través del Administrador de configuración" en la página 728. Para obtener información sobre la concesión de permisos para la reproducción de archivos locales a través de archivos de configuración, consulte "Creación de archivos de configuración para el desarrollo en Flash" en la página 730.

Los archivos SWF locales con sistema de archivos presentan las siguientes restricciones:

- No se puede acceder a la red, lo que incluye lo siguiente:
  - Cargar otros archivos SWF desde la red (*excepto* mediante rutas UNC que no sean de Internet)
  - Enviar peticiones HTTP
  - Realizar conexiones mediante XMLSocket, Flash Remoting o NetConnection
  - Llamar a `getURL()` *excepto* si se utiliza `getURL("file:...")` o `getURL("mailto:...")`

- Se puede interactuar con otros archivos locales con sistema de archivos, aunque existen las siguientes restricciones:
  - Utilizar scripts entre películas (como el acceso de ActionScript a objetos en otros archivos SWF).
  - Llamar a `System.security.allowDomain`
  - Utilizar `LocalConnection` como remitente o detector e independientemente de los controladores `LocalConnection.allowDomain`.

    **NOTA** Los archivos SWF locales con sistema de archivos pueden interactuar con otros archivos SWF locales con sistema de archivos que no sean de red. Sin embargo, no pueden interactuar con archivos SWF locales con acceso a la red.

  Los archivos SWF locales con sistema de archivos tienen acceso de lectura a archivos conocidos del sistema de archivos local. Por ejemplo, puede utilizar `XML.load()` en un archivo SWF local con sistema de archivos siempre que cargue del sistema de archivos local y no de Internet.
- Los archivos SWF locales con sistema de archivos no se pueden comunicar con páginas HTML, lo que incluye lo siguiente:
  - Creación de scripts entrantes (como ExternalInterface API, ActiveX, LiveConnect y XPConnect)
  - Creación de scripts salientes (como llamadas `fscommand` personalizadas y `getURL("javascript:...")`)

    **NOTA** Una excepción a esto es si la página HTML es de confianza.

# Dominios, seguridad entre dominios y archivos SWF

De forma predeterminada, Flash Player 7 o versiones posteriores impiden que un archivo SWF que se encuentre disponible en un dominio pueda leer los datos, objetos o variables de archivos SWF que se encuentran disponibles en otro dominio. Además, el contenido que se carga mediante protocolos que no son seguros (protocolos que no son HTTPS) no puede leer el contenido cargado mediante un protocolo seguro (HTTPS), aunque ambos contenidos se encuentren exactamente en el mismo dominio. Por ejemplo, un archivo SWF situado en http://www.macromedia.com/main.swf no puede cargar datos de https://www.macromedia.com/data.txt sin permiso explícito, de la misma forma que un archivo SWF proporcionado por un dominio no puede cargar datos (mediante `loadVars()`, por ejemplo) de otro dominio.

Las direcciones IP numéricas idénticas son compatibles. Sin embargo, los nombres de dominio no son compatibles con las direcciones IP, aunque el nombre de dominio se resuelva en la misma dirección IP.

En las tablas siguientes se muestran ejemplos de dominios compatibles:

| | |
|---|---|
| www.macromedia.com | www.macromedia.com |
| data.macromedia.com | data.macromedia.com |
| 65.57.83.12 | 65.57.83.12 |

En la tabla siguiente se muestran ejemplos de dominios incompatibles:

| | |
|---|---|
| www.macromedia.com | data.macromedia.com |
| macromedia.com | www.macromedia.com |
| www.macromedia.com | macromedia.com |
| 65.57.83.12 | www.macromedia.com (incluso si este dominio se resuelve en 65.57.83.12 ) |
| www.macromedia.com | 65.57.83.12 (incluso si www.macromedia.com se resuelve en esta dirección IP) |

Flash Player 8 no permite que los archivos SWF locales se comuniquen con Internet sin una configuración adecuada. Para obtener información sobre la configuración de un archivo de configuración para probar contenido de forma local, consulte “Creación de archivos de configuración para el desarrollo en Flash” en la página 730.

Para más información sobre seguridad, consulte www.macromedia.com/devnet/security/ y www.macromedia.com/software/flashplayer/security/.

Para más información, consulte los siguientes temas:

- "Reglas de nombres de dominio para la configuración y los datos locales" en la página 736
- "Acceso a varios dominios y subdominios entre archivos SWF" en la página 736
- "Permiso de carga de datos de varios dominios" en la página 744

## Reglas de nombres de dominio para la configuración y los datos locales

En Flash Player 6, se utilizan de forma predeterminada las reglas de coincidencia de superdominio al acceder a la configuración local (como los permisos de acceso a la cámara o al micrófono) o a los datos localmente persistentes (objetos compartidos). Es decir, la configuración y los datos para los archivos SWF albergados en here.xyz.com, there.xyz.com y xyz.com están compartidos y se almacenan todos en xyz.com.

En Flash Player 7, se utilizan de forma predeterminada las reglas de coincidencia de dominio exacto. Es decir, la configuración y los datos para un archivo albergado en here.xyz.com se almacenan en here.xyz.com, la configuración y los datos para un archivo albergado en there.xyz.com se almacenan en there.xyz.com, y así sucesivamente. La nueva propiedad `System.exactSettings` permite especificar las reglas que se van a utilizar. Esta propiedad se admite para los archivos publicados en Flash Player 6 o una versión posterior. Para los archivos publicados en Flash Player 6, el valor predeterminado es `false`, lo cual significa que se utilizarán las reglas de coincidencia de superdominio. Para los archivos publicados en Flash Player 7, el valor predeterminado es `true`, lo que significa que se utilizarán las reglas de coincidencia de dominio exacto. Si utiliza la configuración o los datos locales persistentes y desea publicar un archivo SWF de Flash Player 6 en Flash Player 7 u 8, es posible que tenga que establecer este valor en `false` en el archivo transferido. Para más información, consulte `%{exactSettings (propiedad System.exactSettings)}%` en la *Referencia del lenguaje ActionScript 2.0*.

## Acceso a varios dominios y subdominios entre archivos SWF

Al desarrollar una serie de archivos SWF que se comunican entre sí en línea (por ejemplo, cuando utiliza `loadMovie()`, `MovieClip.loadMovie()`, `MovieClipLoader.LoadClip()` u objetos LocalConnection), puede albergar los archivos SWF en diferentes dominios o en diferentes subdominios de un mismo superdominio.

En los archivos publicados en Flash Player 5 o una versión anterior, no hay restricciones en el acceso a varios dominios o subdominios.

En los archivos publicados en Flash Player 6, se podía utilizar el controlador `LocalConnection.allowDomain` o el método `System.security.allowDomain()` para especificar que se permitía el acceso a varios dominios (por ejemplo, para permitir que un archivo en someOtherSite.com pudiera acceder a un archivo en someSite.com) y no se necesitaba ningún comando para permitir el acceso a subdominios (por ejemplo, un archivo en store.someSite.com podía acceder a un archivo en www.someSite.com).

Los archivos publicados en Flash Player 7 implementan el acceso entre archivos SWF de forma distinta a como lo hacían en versiones anteriores, de dos formas. En primer lugar, Flash Player 7 implementa las reglas de coincidencia de dominio exacto en lugar de las reglas de coincidencia de superdominio. Por lo tanto, el archivo al cual se accede (aunque se haya publicado en una versión de Flash Player anterior a Flash Player 7) debe permitir explícitamente el acceso a varios dominios o subdominios; este tema se trata en esta sección. En segundo lugar, un archivo albergado en un sitio que utilice un protocolo seguro (HTTPS) debe permitir explícitamente el acceso de un archivo albergado en un sitio que utilice un protocolo inseguro (HTTP o FTP); este tema se trata en la siguiente sección ("Acceso de protocolo HTTP a HTTPS entre archivos SWF" en la página 749).

Por lo general, se llama a `System.security.allowDomain` en las aplicaciones. Sin embargo, cuando el receptor de LocalConnection es un archivo SWF de HTTPS y el remitente no lo es, se llama en su lugar a `allowInsecureDomain`.

El siguiente caso sólo afecta a archivos SWF publicados para Flash Player 7. Cuando el receptor es HTTPS y el remitente es un archivo SWF local, se llama a `allowDomain()`, aun cuando se debería llamar a `allowInsecureDomain()`. Sin embargo, en Flash Player 8, cuando un objeto LocalConnection receptor HTTPS es Flash Player 8 y el remitente es un archivo local, se llama a `allowInsecureDomain()`.

Los archivos que se ejecutan en Flash Player 8 están sujetos a cambios con respecto a la forma en que se ejecutan en Flash Player 7. Al llamar a `System.security.allowDomain`, se permiten operaciones de uso de scripts entre películas sólo cuando el archivo SWF al que se accede es el que se denomina `System.security.allowDomain`. En otras palabras, un archivo SWF que llama a `System.security.allowDomain` sólo permite ahora el acceso a sí mismo. En versiones anteriores, al llamar a `System.security.allowDomain` se permitían operaciones de uso de scripts entre películas en las que el archivo SWF al que se accedía podía ser cualquier archivo SWF del mismo dominio que el denominado `System.security.allowDomain`. Al proceder de ese modo, se abría todo el dominio del archivo SWF que estaba llamando.

Se ha añadido compatibilidad para el valor de comodín (*) en `System.security.allowDomain("*")` y `System.security.allowInsecureDomain("*")`. El valor de comodín (*) permite operaciones de uso de scripts entre películas en las que el archivo que accede es cualquier archivo y se puede cargar desde cualquier ubicación (como el permiso global). Los permisos del comodín pueden ser de gran utilidad, aunque deben adherirse a las nuevas reglas de seguridad de archivos locales en Flash Player 8. De forma específica, los archivos locales no proceden de un dominio, por lo que se debe utilizar el valor de comodín. Sin embargo, hay que tener especial cuidado al utilizar el valor de comodín ya que cualquier dominio tendrá acceso al archivo. Para más información, consulte `%{allowInsecureDomain (método security.allowInsecureDomain)}%`.

Se podría dar la situación en la que se carga un archivo SWF secundario desde un dominio diferente al que llama. Podría ser aconsejable permitir el uso de scripts de ese archivo en el archivo SWF principal, aunque no se sabrá el dominio final del que procederá el archivo SWF secundario. Este caso puede ocurrir, por ejemplo, cuando se utilizan redirecciones de reparto de carga o servidores de terceros. En esta situación, puede utilizar la propiedad `MovieClip._url` como argumento de este método. Por ejemplo, si carga un archivo SWF en `my_mc`, podrá llamar a `System.security.allowDomain(my_mc._url)`. Si lo hace, debe esperar a que el archivo SWF empiece a cargarse en `my_mc`, ya que la propiedad `_url` no dispone aún de su valor final correcto. Para determinar si un archivo SWF secundario ha comenzado a cargarse, utilice `MovieClipLoader.onLoadStart`.

También puede darse la situación contraria; es decir, podría crear un archivo SWF secundario en el que quiera que su archivo principal use scripts, pero no conoce cuál será el dominio de su archivo SWF principal (lo que significa que se trata de un archivo SWF que podría cargarse por medio de varios dominios). En este caso, llame a `System.security.allowDomain(_parent._url)` desde el archivo SWF secundario. No tiene que esperar a que el archivo SWF principal se cargue porque se ha cargado antes de que lo haga el archivo secundario.

**NOTA**

Si el archivo SWF de Internet al que se accede se carga desde una dirección URL HTTPS, el archivo debe llamar a `System.security.allowInsecureDomain("*")`.

En la siguiente tabla se resumen las reglas de coincidencia de dominio en diferentes versiones de Flash Player:

| Archivos publicados para Flash Player | Acceso a varios dominios entre archivos SWF (se necesita allowDomain()) | Acceso a subdominios entre archivos SWF |
|---|---|---|
| 5 o anterior | Sin restricciones | Sin restricciones |
| 6 | Coincidencia de superdominio: se necesita `allowDomain()` si no coinciden los superdominios. | Sin restricciones |
| 7 y posterior | Coincidencia exacta de dominio Permiso explícito para archivos alojados en HTTPS para acceder a archivos alojados en HTTP o FTP | Coincidencia exacta de dominio Permiso explícito para archivos alojados en HTTPS para acceder a archivos alojados en HTTP o FTP |

**NOTA**

Se necesita `System.security.allowInsecureDomain` en Flash Player 7 y versiones posteriores si realiza acceso HTTP a HTTPS, incluso si tiene una coincidencia de dominio exacto.

Las versiones que controlan el comportamiento de Flash Player son las versiones de archivo SWF (versión publicada de un archivo SWF), no la versión del propio Flash Player. Por ejemplo, cuando Flash Player 8 reproduce un archivo SWF publicado para la versión 7, Flash Player presenta un comportamiento coherente con la versión 7. Esta práctica evita que las actualizaciones del reproductor modifiquen el comportamiento de `System.security.allowDomain` en los archivos SWF implementados.

Puesto que Flash Player 7 y las versiones posteriores implementan reglas de coincidencia de dominio exacto en lugar de reglas de coincidencia de superdominio, es posible que tenga que modificar los scripts existentes si quiere acceder a ellos desde archivos publicados en Flash Player 7 u 8 (los archivos modificados se podrán publicar igualmente en Flash Player 6). Si ha utilizado alguna sentencia `LocalConnection.allowDomain()` o `System.security.allowDomain()` en sus archivos y ha especificado que se permita el acceso a sitios de superdominio, deberá cambiar sus parámetros de modo que especifiquen dominios exactos. En el siguiente ejemplo se muestran los cambios que podría ser necesario hacer si se tiene el código de Flash Player 6:

```
// Comandos de Flash Player 6 en un archivo SWF en www.anyOldSite.com
// para permitir el acceso de los archivos SWF albergados en
  www.someSite.com
// o en store.someSite.com
System.security.allowDomain("someSite.com");
my_lc.allowDomain = function(sendingDomain)
  return(sendingDomain=="someSite.com");
}
// Comandos correspondientes para permitir el acceso de los archivos SWF
// publicados en Flash Player 7 o versiones posteriores
System.security.allowDomain("www.someSite.com", "store.someSite.com");
my_lc.allowDomain = function(sendingDomain)
  return(sendingDomain=="www.someSite.com" ||
    sendingDomain=="store.someSite.com");
}
```

También es posible que tenga que añadir sentencias como éstas a sus archivos si actualmente no las está utilizando. Por ejemplo, si el archivo SWF está albergado en www.someSite.com y desea permitir el acceso de un archivo SWF publicado en Flash Player 7 o versiones posteriores en store.someSite.com, deberá añadir sentencias como las del siguiente ejemplo al archivo que está en www.someSite.com (el archivo que está en www.someSite.com podrá publicarse igualmente en Flash Player 6):

```
System.security.allowDomain("store.someSite.com");
my_lc.allowDomain = function(sendingDomain)
  return(sendingDomain=="store.someSite.com");
}
```

Además, tenga en cuenta que si una aplicación Flash Player 6 que se ejecuta en Flash Player 7 intenta acceder a datos situados fuera de su dominio concreto, se aplicarán las reglas de coincidencia de dominio de Flash Player 7 y versiones posteriores, y se pedirá al usuario que conceda o deniegue el acceso.

En resumen, deberá modificar sus archivos añadiendo o cambiando sentencias `allowDomain` si quiere publicar archivos en Flash Player 7 o versiones posteriores que cumplan las siguientes condiciones:

- Ha implementado la creación de scripts entre películas del archivo SWF (consulte"Permiso de acceso a datos entre archivos SWF de varios dominios" en la página 741).
- El archivo SWF (de cualquier versión) llamado no está albergado en un sitio que utiliza un protocolo seguro (HTTPS) o bien los archivos SWF que efectúan la llamada y que reciben la llamada están albergados en sitios HTTPS. (Si sólo se utiliza con HTTPS el archivo SWF llamado, consulte "Acceso de protocolo HTTP a HTTPS entre archivos SWF" en la página 749.)

- Los archivos SWF no están en el mismo dominio (por ejemplo, un archivo está en www.domain.com y otro en store.domain.com).

Deberá realizar los siguientes cambios:

- Si el archivo SWF llamado se publica en Flash Player 7, incluya `System.security.allowDomain` o `LocalConnection.allowDomain` en el mismo, para que utilice la coincidencia de nombre de dominio exacto.
- Si el archivo SWF llamado se publica en Flash Player 6, modifíquelo añadiendo o cambiando una sentencia `System.security.allowDomain` o `LocalConnection.allowDomain`, para que utilice la coincidencia de nombre de dominio exacto, tal como se muestra en los ejemplos de código explicados en esta misma sección. Puede publicar el archivo modificado en Flash Player 6 o 7.
- Si el archivo SWF llamado se publica en Flash Player 5 o una versión anterior, transfiéralo a Flash Player 6 o 7 y añada una sentencia `System.security.allowDomain`, para que utilice la coincidencia de nombre de dominio exacto, tal como se muestra en los ejemplos de código explicados en esta misma sección. (Los objetos LocalConnection no se admiten en Flash Player 5 o una versión anterior.)

Para obtener información sobre los entornos limitados locales de seguridad, consulte "Seguridad de archivos local y Flash Player" en la página 717.

## Permiso de acceso a datos entre archivos SWF de varios dominios

Para que cada archivo pueda acceder a los datos del otro (variables y objetos), ambos tienen que haberse creado en el mismo dominio. De forma predeterminada, en Flash Player 7 y versiones posteriores, los dos dominios deben coincidir exactamente para que los dos archivos puedan compartir datos. No obstante, es posible que un archivo SWF permita acceder a varios archivos SWF que se encuentran disponibles en dominios específicos llamando a `LocalConnection.allowDomain` o `System.security.allowDomain()`.

`System.security.allowDomain()` permite archivos SWF y HTML en dominios especificados acceder a objetos y variables en el archivo SWF que contiene la llamada `allowDomain()`.

Si hay dos archivos SWF en el mismo dominio (por ejemplo, http://misitio.com/movieA.swf y http://misitio.com/movieB.swf), el archivo movieA.swf puede examinar y modificar variables, objetos, propiedades, métodos y demás en movieB.swf, y movieB puede hacer lo mismo en movieA. Esto se denomina *uso de scripts entre películas*.

Si los dos archivos SWF están en dominios diferentes (por ejemplo, http://misitio.com/movieA.swf y http://otrositio.com/movieB.swf), de forma predeterminada Flash Player no permite a movieA.swf usar scripts en movieB.swf, ni viceversa. Mediante una llamada a `System.security.allowDomain("mysite.com")`, movieB.swf concede permiso a movieA.swf para usar scripts en él. Un archivo SWF concede permiso a otros archivos SWF de dominios diferentes para usar scripts en él llamando a `System.security.allowDomain()`. Esto se denomina *creación de scripts en varios dominios*.

Para más información sobre `System.security.allowDomain()`, uso de scripts entre películas y scripts creados en varios dominios, consulte %{allowDomain (método security.allowDomain)}% en *Referencia del lenguaje ActionScript 2.0*.

Por ejemplo, suponga que main.swf está disponible en www.macromedia.com. El archivo SWF cargará otro archivo SWF (data.swf) de data.macromedia.com en una instancia de clip de película creada dinámicamente mediante `createEmptyMovieClip()`.

```
// En macromedia.swf
this.createEmptyMovieClip("target_mc", this.getNextHighestDepth());
target_mc.loadMovie("http://data.macromedia.com/data.swf");
```

Supongamos ahora que data.swf define un método llamado `getData()` en su línea de tiempo principal. De manera predeterminada, main.swf no puede llamar al método `getData()` definido en data.swf después de que dicho archivo se haya cargado, dado que los dos archivos SWF no residen en el mismo dominio. Por ejemplo, una vez que el archivo data.swf se haya cargado, la llamada de método siguiente en el archivo main.swf no se podrá realizar:

```
// En macromedia.swf, una vez que el archivo data.swf se haya cargado:
target_mc.getData(); // Esta llamada de método no se podrá realizar
```

Sin embargo, es posible que el archivo data.swf permita acceder a archivos SWF que se encuentren disponibles en www.macromedia.com mediante el uso del controlador `LocalConnection.allowDomain` y del método `System.security.allowDomain()`, según el tipo de acceso necesario. El código siguiente, añadido a data.swf, permite que un archivo SWF que se encuentre disponible en www.macromedia.com pueda acceder a sus variables y métodos:

```
// En data.swf
this._lockroot = true;
System.security.allowDomain("www.macromedia.com");
var my_lc:LocalConnection = new LocalConnection();
my_lc.allowDomain = function(sendingDomain:String):Boolean {
  return (sendingDomain == "www.macromedia.com");
};
function getData():Void {
  var timestamp:Date = new Date();
  output_txt.text += "data.swf:" + timestamp.toString() + "\n\n";
}
output_txt.text = "**INIT**:\n\n";
```

El archivo macromedia.swf podrá llamar ahora a la función `getData` cargada en el archivo SWF. Tenga en cuenta que `allowDomain` permite a cualquier archivo SWF del dominio permitido crear un script de cualquier otro archivo SWF del dominio que permita el acceso, a menos que el archivo SWF al que acceda esté albergado en un sitio que use un protocolo seguro (HTTPS).

Para más información sobre coincidencias de nombre de dominio, consulte "Acceso a varios dominios y subdominios entre archivos SWF" en la página 736.

# Archivos de política de servidor para permitir el acceso a los datos

Un documento de Flash puede cargar datos desde una fuente externa mediante una de las siguientes llamadas para cargar datos: `XML.load()`, `XML.sendAndLoad()`, `LoadVars.load()`, `LoadVars.sendAndLoad()`, `loadVariables()`, `loadVariablesNum()`, `MovieClip.loadVariables()`, `XMLSocket.connect()` y Macromedia Flash Remoting (`NetServices.createGatewayConnection`). Asimismo, un archivo SWF puede importar bibliotecas compartidas en tiempo de ejecución (RSL) o elementos definidos en otro archivo SWF durante el tiempo de ejecución. De forma predeterminada, los datos o las RSL deben residir en el mismo dominio que el archivo SWF que está cargando los medios o datos externos.

Para que los datos y los elementos de las bibliotecas compartidas en tiempo de ejecución puedan ser utilizados por los archivos SWF de diferentes dominios, utilice un *archivo de política para distintos dominios*. Un archivo de política para distintos dominios es un archivo XML que permite al servidor indicar que sus datos y documentos están disponibles para los archivos SWF que se encuentran disponibles en determinados dominios o en todos los dominios. Cualquier archivo SWF que se encuentre disponible en un dominio especificado por el archivo de política del servidor podrá acceder a los datos, activos y RSL desde dicho servidor.

Si está cargando datos externos, debe crear archivos de política aunque no tenga previsto transferir ningún archivo a Flash Player 7. Si utiliza RSL, deberá crear archivos de política si el archivo que efectúa la llamada o el archivo llamado se publica en Flash Player 7.

Para más información, consulte los siguientes temas:

- "Permiso de carga de datos de varios dominios" en la página 744
- "Ubicaciones personalizadas de archivos de política" en la página 746
- "Archivos de política XMLSocket" en la página 747

## Permiso de carga de datos de varios dominios

Cuando un documento de Flash intenta acceder a datos desde otro dominio, Flash Player intenta cargar automáticamente un archivo de política desde dicho dominio. Si el dominio del documento Flash que intenta acceder a los datos se incluye en el archivo de política, se podrá acceder a los datos de forma automática.

Los archivos de política deben denominarse crossdomain.xml y deben residir en el directorio raíz u otro directorio del servidor en el que los datos se encuentran disponibles con código ActionScript adicional (consulte "Ubicaciones personalizadas de archivos de política" en la página 746). Los archivos de política sólo funcionan en servidores que se comunican utilizando los protocolos HTTP, HTTPS o FTP. Los archivos de política son específicos del puerto y el protocolo del servidor en el que residen.

Por ejemplo, un archivo de política ubicado en https://www.macromedia.com:8080/crossdomain.xml sólo se aplica a las llamadas para cargar datos realizadas a www.macromedia.com en el puerto 8080 utilizando el protocolo HTTPS.

Una excepción a esta regla es cuando se utiliza un objeto XMLSocket para conectar con un servidor de socket de otro dominio. En ese caso, un servidor HTTP que se ejecuta en el puerto 80 del mismo dominio que el servidor de socket deberá proporcionar el archivo de política para la llamada de método.

Un archivo de política XML contiene una sola etiqueta `<cross-domain-policy>`, la cual, a su vez, no contiene ninguna o contiene varias etiquetas `<allow-access-from>`. Cada etiqueta `<allow-access-from>` contiene un atributo, `domain`, el cual especifica una dirección IP exacta, un dominio exacto o un dominio comodín (cualquier dominio). Los dominios comodín se indican mediante un solo asterisco (*), que incluye todos los dominios y todas las direcciones IP o mediante un asterisco seguido por un sufijo, que incluye sólo los dominios que terminan con el sufijo especificado. Los sufijos deben empezar por un punto. Sin embargo, los dominios comodín con sufijos pueden coincidir con los dominios formados únicamente por el sufijo sin el punto inicial. Por ejemplo, se considera que foo.com forma parte de *.foo.com. Los comodines no pueden utilizarse en las especificaciones de dominio IP.

Si especifica una dirección IP, sólo se otorgará acceso a los archivos SWF cargados desde esa dirección IP mediante la sintaxis de IP (por ejemplo, http://65.57.83.12/flashmovie.swf), no a los archivos que se hayan cargado mediante una sintaxis de nombre de dominio. Flash Player no lleva a cabo la resolución DNS.

En el siguiente ejemplo se muestra un archivo de política que permite acceder a documentos de Flash originados en foo.com, www.friendOfFoo.com, *.foo.com y 105.216.0.40, a partir de un documento de Flash en foo.com:

```
<?xml version="1.0"?>
<!-- http://www.foo.com/crossdomain.xml -->
<cross-domain-policy>
  <allow-access-from domain="www.friendOfFoo.com" />
  <allow-access-from domain="*.foo.com" />
  <allow-access-from domain="105.216.0.40" />
</cross-domain-policy>
```

También puede permitir el acceso a documentos procedentes de cualquier dominio, tal y como se muestra en el siguiente ejemplo:

```
<?xml version="1.0"?>
<!-- http://www.foo.com/crossdomain.xml -->
<cross-domain-policy>
  <allow-access-from domain="*" />
</cross-domain-policy>
```

Cada etiqueta `<allow-access-from>` también tiene el atributo opcional `secure`. El atributo `secure` adopta de manera predeterminada el valor `true`. Puede establecer el atributo con el valor `false` si el archivo de política está en un servidor HTTPS y desea permitir que los archivos SWF de un servidor HTTP carguen datos del servidor HTTPS.

La configuración del atributo `secure` con el valor `false` puede poner en peligro la seguridad proporcionada por HTTPS.

Si el archivo SWF que está descargando procede de un servidor HTTPS pero el archivo SWF que lo carga se encuentra en un servidor HTTP, deberá añadir el atributo `secure="false"` a la etiqueta `<allow-access-from>`, tal y como se muestra en el siguiente código:

```
<allow-access-from domain="www.foo.com" secure="false" />
```

Un archivo de política que no contenga ninguna etiqueta `<allow-access-from>` tiene el mismo efecto que un servidor que no tenga ninguna política.

# Ubicaciones personalizadas de archivos de política

Flash Player 7 (7.0.19.0) admite un método denominado `System.security.loadPolicyFile`. Este método le permite especificar una ubicación personalizada en un servidor en la que se encuentra un archivo de política de varios dominios de forma que no tenga que encontrarse en el directorio raíz. Flash Player 7 (7.0.14.0) sólo buscaba archivos de política en la ubicación raíz de un servidor, pero puede no resultar adecuado para un administrador de sitio situar este archivo en el directorio raíz. Para más información sobre el método `loadPolicyFile` y las conexiones XMLSocket, consulte "Archivos de política XMLSocket" en la página 747 y `%{loadPolicyFile (método security.loadPolicyFile)}%` en *Referencia del lenguaje ActionScript 2.0*.

Si utiliza el método `loadPolicyFile`, un administrador de sitio podrá situar el archivo de política en cualquier directorio, siempre y cuando los archivos SWF que necesiten utilizar el archivo de política llamen a `loadPolicyFile` para indicar a Flash Player el lugar en el que está ubicado el archivo de política. No obstante, los archivos de política que no se sitúan en el directorio raíz tienen un alcance limitado. El archivo de política sólo permite el acceso a ubicaciones situadas en el mismo nivel o en niveles inferiores de la jerarquía del servidor.

El método `loadPolicyFile` sólo está disponible en Flash Player 7 (7.0.19.0) o versiones posteriores. Los autores de archivos SWF que utilicen el método `loadPolicyFile` deben hacer lo siguiente:

- Exigir Flash Player 7 (7.0.19.0) o posteriores.
- Organizar el sitio del que proceden los datos de forma que disponga de un archivo de política en la ubicación predeterminada (el directorio raíz) y en la ubicación no predeterminada. Las versiones anteriores de Flash Player utilizarán la ubicación predeterminada.

En caso contrario, los autores deberán crear archivos SWF de manera que se implemente un error de operación de carga de varios dominios.

**ATENCIÓN**

Si el archivo SWF utiliza `loadPolicyFile`, los visitantes que dispongan de Flash Player 6 o versiones anteriores o Flash Player 7 (7.0.19.0) o versiones posteriores no experimentarán ningún problema. Sin embargo, los visitantes que tengan Flash Player 7 (7.0.14.0) no dispondrán de compatibilidad con `loadPolicyFile`.

Si desea utilizar un archivo de política situado en una ubicación personalizada del servidor, deberá realizar una llamada a `System.security.loadPolicyFile` *antes* de cualquier solicitud que dependa del archivo de política, como en el siguiente caso:

```
System.security.loadPolicyFile("http://www.foo.com/folder1/folder2/
  crossdomain.xml");
var my_xml:XML = new XML();
my_xml.load("http://www.foo.com/folder1/folder2/myData.xml");
```

Puede cargar varios archivos de política con ámbitos solapados mediante `loadPolicyFile`. Para todas las solicitudes, Flash Player intenta consultar todos los archivos en cuyo ámbito se incluya la ubicación de la solicitud. Aunque un archivo de política no conceda acceso a varios dominios, esto no impedirá que otro archivo conceda acceso a los datos. Si todos los intentos de acceso resultan infructuosos, Flash Player busca en la ubicación predeterminada del archivo crossdomain.xml (en el directorio raíz). La solicitud fallará si no se encuentra ningún archivo de política en la ubicación predeterminada.

## Archivos de política XMLSocket

Para un intento de conexión XMLSocket, Flash Player 7 (7.0.14.0) buscaba crossdomain.xml en un servidor HTTP en el puerto 80 del subdominio con el que se intentaba establecer la conexión. Flash Player 7 (7.0.14.0) y todas las versiones anteriores limitaban las conexiones XMLSocket con los puertos 1024 y superiores. Sin embargo, en Flash Player 7 (7.0.19.0) y versiones posteriores, ActionScript puede informar a Flash Player de la ubicación no predeterminada de un archivo de política empleando `System.security.loadPolicyFile`. Aun así, todas las ubicaciones personalizadas de archivos de política XMLSocket deben encontrarse en un servidor de socket XML.

En el siguiente ejemplo, Flash Player recupera un archivo de política de un URL especificado:

```
System.security.loadPolicyFile("http://www.foo.com/folder/policy.xml");
```

Los permisos otorgados por el archivo de política de dicha ubicación se aplicarán a todo el contenido del mismo nivel o de un nivel inferior de la jerarquía del servidor. Por consiguiente, si intenta cargar los siguientes datos, observará que sólo puede cargar datos procedentes de determinadas ubicaciones:

```
myLoadVars.load("http://foo.com/sub/dir/vars.txt"); // permitido
myLoadVars.load("http://foo.com/sub/dir/deep/vars2.txt"); // permitido
myLoadVars.load("http://foo.com/elsewhere/vars3.txt"); // no permitido
```

Para resolver este inconveniente, puede cargar más de un archivo de política en un mismo archivo SWF empleando `loadPolicyFile`. Flash Player siempre espera a que finalicen las descargas de archivos de política antes de denegar un solicitud que requiera un archivo de política. Flash Player consulta la ubicación predeterminada de crossdomain.xml si no se han autorizado otras políticas en el archivo SWF.

Una sintaxis especial permite recuperar los archivos de política directamente desde un servidor XMLSocket:

```
System.security.loadPolicyFile("xmlsocket://foo.com:414");
```

En este ejemplo, Flash Player intenta recuperar un archivo de política desde el host y el puerto especificados. Podrá utilizar cualquier puerto si el archivo de política no está en el directorio predeterminado (raíz); en caso contrario, el puerto se limita a 1024 y superiores (al igual que en reproductores anteriores). Cuando se establece una conexión con el puerto especificado, Flash Player envía `<policy-file-request />`, terminada por un byte nulo.

El servidor de socket XML puede estar configurado para proporcionar archivos de política de las siguientes formas:

- Para proporcionar archivos de política y conexiones de socket normales a través del mismo puerto. El servidor esperaría a la `<policy-file-request />` antes de transmitir un archivo de política.
- Para que proporcione archivos de política en un puerto distinto al de las conexiones normales, en cuyo caso podría enviar un archivo de política en cuanto se establece una conexión en el puerto del archivo de política dedicado.

El servidor debe enviar un byte nulo para terminar un archivo de política antes de que se cierre la conexión. Si el servidor no cierra la conexión, Flash Player lo hace recibiendo el byte nulo de terminación.

Un archivo de política proporcionado por un servidor de socket XML tiene la misma sintaxis que cualquier otro archivo de política, pero debe especificar también los puertos a los que se concede el acceso. Los puertos permitidos se especifican en el atributo `to-ports` de la etiqueta `<allow-access-from>`. Si un archivo de política es inferior al puerto 1024, podrá conceder acceso a cualquier puerto; cuando un archivo de política procede del puerto 1024 u otro superior, sólo puede conceder acceso a otros puertos superiores a 1024. Se permiten números de puerto individuales, rangos de puertos y comodines. El siguiente código es un ejemplo de archivo de política XMLSocket:

```
<cross-domain-policy>
<allow-access-from domain="*" to-ports="507" />
<allow-access-from domain="*.foo.com" to-ports="507,516" />
<allow-access-from domain="*.bar.com" to-ports="516-523" />
<allow-access-from domain="www.foo.com" to-ports="507,516-523" />
<allow-access-from domain="www.bar.com" to-ports="*" />
</cross-domain-policy>
```

Como en Flash Player 7 (7.0.19.0) se permite la capacidad de conectar con puertos inferiores a 1024, para autorizar esta conexión siempre es necesario un archivo de política cargado con `loadPolicyFile`, incluso cuando un archivo SWF se conecta a su propio subdominio.

# Acceso de protocolo HTTP a HTTPS entre archivos SWF

Debe utilizar un controlador o método `allowDomain` para permitir que un archivo SWF de un dominio pueda acceder a un archivo SWF de otro dominio. No obstante, si el archivo SWF al que se accede está albergado en un sitio que usa un protocolo seguro (HTTPS), el controlador o método `allowDomain` no permitirá el acceso de un archivo SWF albergado en un sitio que use un protocolo no seguro. Para permitir el acceso en esas condiciones, debe usar la sentencia `LocalConnection.allowInsecure Domain()` o `System.security.allowInsecureDomain()`. Consulte "Permiso de acceso de protocolo HTTP a HTTPS entre archivos SWF" en la página 749 para más información.

## Permiso de acceso de protocolo HTTP a HTTPS entre archivos SWF

Además de las reglas de coincidencia de dominio exacto, debe permitir explícitamente que los archivos albergados en sitios que utilizan un protocolo inseguro puedan acceder a los archivos albergados en sitios que utilizan un protocolo seguro (HTTPS). Dependiendo de si el archivo llamado se publica para Flash Player 6, 7 u 8, deberá implementar una de las sentencias `allowDomain` (consulte "Acceso a varios dominios y subdominios entre archivos SWF" en la página 736) o utilizar las sentencias `LocalConnection.allowInsecureDomain` o `System.security.allowInsecureDomain()`.

Por ejemplo, si el archivo SWF de https://www.someSite.com/data.swf debe permitir el acceso de un archivo SWF de http://www.someSite.com, el código siguiente, añadido a data.swf, permitirá el acceso:

```
// En data.swf
System.security.allowInsecureDomain("www.someSite.com");
my_lc.allowInsecureDomain = function(sendingDomain:String):Boolean {
  return (sendingDomain == "www.someSite.com");
};
```

ADVERTENCIA

Al implementar una sentencia `allowInsecureDomain()` se pone en juego la seguridad ofrecida por el protocolo HTTPS. Estos cambios sólo deben realizarse si no puede reorganizar el sitio de manera que todos los archivos SWF se encuentren disponibles en el protocolo HTTPS.

El código siguiente muestra un ejemplo de cambios que puede que tenga que realizar:

```
// Comandos en un archivo SWF de Flash Player 6 en https://www.someSite.com
// para permitir el acceso de los archivos SWF de Flash Player 7 albergados
// en http://www.someSite.com o en http://www.someOtherSite.com
System.security.allowDomain("someOtherSite.com");
my_lc.allowDomain = function(sendingDomain)
  return(sendingDomain=="someOtherSite.com");
}
// Comandos correspondientes en un archivo SWF de Flash Player 7
// para permitir el acceso de los archivos SWF de Flash Player 7 albergados
// en http://www.someSite.com o en http://www.someOtherSite.com
System.security.allowInsecureDomain("www.someSite.com",
  "www.someOtherSite.com");
my_lc.allowInsecureDomain = function(sendingDomain) {
  return(sendingDomain=="www.someSite.com" ||
    sendingDomain=="www.someOtherSite.com");
}
```

También es posible que tenga que añadir sentencias como éstas a sus archivos si actualmente no las está utilizando. Es posible que deba hacerse una modificación aunque ambos archivos se encuentren en el mismo dominio (por ejemplo, si un archivo en http://www.domain.com llama a otro archivo en https://www.domain.com).

En resumen, es posible que tenga que modificar sus archivos añadiendo o cambiando sentencias si quiere publicar archivos en Flash Player 7 o versiones posteriores que cumplan las siguientes condiciones:

- Ha implementado la creación de scripts entre películas entre archivos SWF (mediante `loadMovie()`, `MovieClip.loadMovie()`, `MovieClipLoader.LoadClip()` u objetos LocalConnection).
- El archivo que efectúa la llamada no se alberga en un sitio con protocolo HTTPS, y el archivo llamado sí utiliza HTTPS.

Deberá realizar los siguientes cambios:

- Si el archivo llamado se publica en Flash Player 7, incluya `System.security.allowInsecureDomain` o `LocalConnection.allowInsecureDomain` en el mismo, para que utilice la coincidencia de nombre de dominio exacto, tal como se muestra en los ejemplos de código explicados en esta misma sección.
- Si el archivo llamado se publica en Flash Player 6 o una versión anterior y tanto el archivo que efectúa la llamada como el archivo llamado se encuentran en el mismo dominio (por ejemplo, un archivo en http://www.domain.com llama a otro archivo en https://www.domain.com), no es necesaria ninguna modificación.
- Si el archivo llamado se publica en Flash Player 6, los archivos no se encuentran en el mismo dominio y no desea transferir el archivo llamado a Flash Player 7, modifique dicho archivo añadiendo o cambiando una sentencia `System.security.allowDomain` o `LocalConnection.allowDomain`, para que utilice la coincidencia de nombre de dominio exacto, tal como se muestra en los ejemplos de código explicados en esta misma sección.
- Si el archivo llamado se publica en Flash Player 6 y desea transferirlo a Flash Player 7, incluya `System.security.allowInsecureDomain` o `LocalConnection.allowInsecureDomain` en el mismo, para que utilice la coincidencia de nombre de dominio exacto, tal como se muestra en los ejemplos de código explicados en esta misma sección.
- Si el archivo llamado se publica en Flash Player 5 o una versión anterior y los archivos no se encuentran en el mismo dominio, puede realizar una de estas dos acciones. Puede transferir el archivo llamado a Flash Player 6 y añadir o cambiar una sentencia `System.security.allowDomain`, para que utilice la coincidencia de nombre de dominio exacto, tal como se muestra en los ejemplos de código explicados en esta misma sección, o bien puede transferir el archivo llamado a Flash Player 7 e incluir una sentencia `System.security.allowInsecureDomain` en el mismo, para que utilice la coincidencia de nombre de dominio exacto, tal como se muestra en los ejemplos de código explicados en esta misma sección.

CAPÍTULO 18

# 18 Depuración de aplicaciones

Macromedia Flash Basic 8 y Macromedia Flash Professional 8 proporcionan diversas herramientas para probar el código ActionScript en los archivos SWF. El depurador permite detectar errores en un archivo SWF mientras se ejecuta en el reproductor de depuración de Flash (consulte "Depuración de los scripts" en la página 754). Flash también proporciona las siguientes herramientas de depuración adicionales:

- El panel Salida, que muestra mensajes de error (incluidos algunos errores de tiempo de ejecución) y listas de variables y objetos (consulte "Utilización del panel Salida" en la página 768)
- La sentencia `trace`, que envía notas de programación y valores de expresiones al panel Salida (consulte "Utilización de la sentencia trace" en la página 772)
- Las sentencias `throw` y `try..catch..finally`, que sirven para probar y responder a los errores de tiempo de ejecución desde el script.

En esta sección se muestra cómo depurar los scripts y aplicaciones de Flash mediante el depurador, así como la forma de utilizar el panel Salida. Para más información, consulte los siguientes temas:

# Depuración de los scripts

El depurador de Flash 8 ayuda a detectar errores en el archivo SWF mientras se ejecuta en Flash Player. Debe visualizar el archivo SWF con una versión especial de Flash Player llamada *Reproductor de depuración de Flash*. Cuando se instala la herramienta de edición, se instala automáticamente Reproductor de depuración de Flash. Así pues, si instala Flash y explora un sitio Web con contenido Flash o utiliza la opción Probar película, estará utilizando el Reproductor de depuración de Flash. También puede ejecutar el instalador en el siguiente directorio de Windows o Macintosh: *Directorio de instalación de Flash*\Players\Debug\ directorio o iniciar la aplicación Reproductor de depuración de Flash independiente desde el mismo directorio.

Si utiliza el comando Control > Probar película para probar archivos SWF con controles de teclado implementados (tabulación, métodos abreviados de teclado creados mediante `Key.addListener()`, etc.), seleccione Control > Deshabilitar métodos abreviados de teclado. Al seleccionar esta opción impedirá que el entorno de edición "capte" las pulsaciones de teclado, permitiendo que lleguen hasta el reproductor. Por ejemplo, en el entorno de edición, Ctrl+U abre el cuadro de diálogo Preferencias. Si el script asigna Ctrl+U a una acción que subraya el texto en pantalla, cuando se utilice Probar película, al presionar Ctrl+U abrirá el cuadro de diálogo Preferencias en lugar de ejecutar la acción de subrayar texto. Para permitir que el comando Ctrl+U llegue hasta el reproductor, debe seleccionar Control > Deshabilitar métodos abreviados de teclado.

| ATENCIÓN | Si utiliza una aplicación en lengua no inglesa en un sistema en lengua inglesa, el comando Probar película dará un error si una parte cualquiera de la ruta del archivo SWF incluye caracteres que no pueden representarse con el esquema de codificación MBCS. Por ejemplo, las rutas japonesas no funcionan en un sistema inglés. Esta limitación se aplica a todas las áreas de la aplicación que utilizan el reproductor externo. |
|---|---|

El depurador muestra una lista jerárquica de clips de película cargados actualmente en Flash Player. Con el depurador, puede ver y modificar los valores de variables y de propiedades mientras se reproduce el archivo SWF; además, puede utilizar puntos de corte para detener el archivo SWF y desplazarse por el código ActionScript línea a línea.

Puede utilizar el depurador en modo de prueba con archivos locales, o bien para probar archivos de un servidor Web que se encuentren en una ubicación remota. El depurador permite establecer puntos de corte en el código de ActionScript que detienen Flash Player y desplazarse por el código a medida que se ejecuta. Después puede volver a editar los scripts de modo que produzcan los resultados correctos.

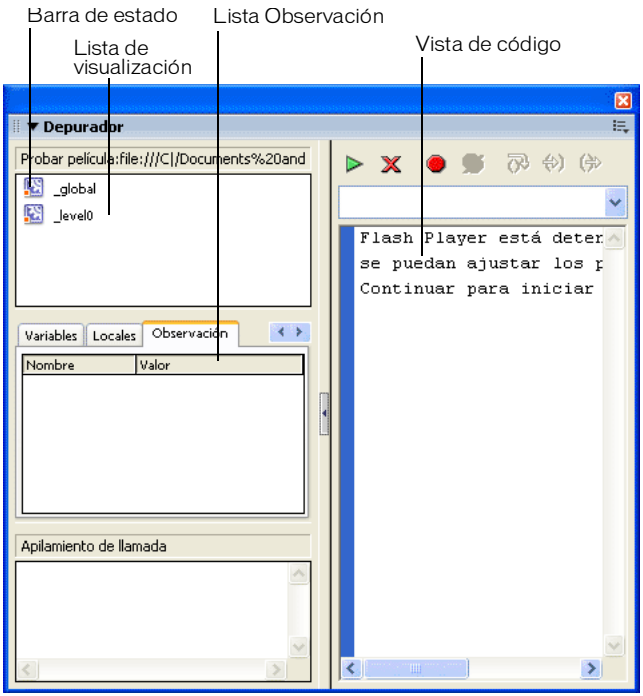
Una vez activado el depurador, su barra de estado muestra la URL o la ruta del archivo, indica si el depurador se ejecuta en modo de prueba o desde una ubicación remota y muestra una vista dinámica de la lista de visualización del clip de película. Cuando se añaden o eliminan clips de película del archivo, la lista de visualización refleja los cambios al instante. Puede cambiar el tamaño de la lista de visualización moviendo el separador horizontal.

### Para activar el Depurador en modo de prueba:

- Seleccione Control > Depurar película.

  Este comando exporta el archivo SWF con información de depuración (el archivo SWD) y permite la depuración del archivo SWF. Abre el depurador y abra el archivo SWF en modo de prueba.

| NOTA | Si es necesario, puede cambiar el tamaño de las diferentes regiones del panel Depurador. Cuando el puntero cambia entre cada región, puede arrastrarlo para cambiar el tamaño de las listas de Visualización y Observación y de la vista de código. |
|---|---|



Para más información, consulte los siguientes temas:

- “Depuración de un archivo SWF desde una ubicación remota”
- “Visualización y modificación de variables” en la página 758
- “Utilización de la lista Observación” en la página 760
- “Visualización de las propiedades de un clip de película y modificación de las propiedades editables” en la página 762
- “Establecimiento y eliminación de puntos de corte” en la página 763
- “Desplazamiento por las líneas de código” en la página 766

# Depuración de un archivo SWF desde una ubicación remota

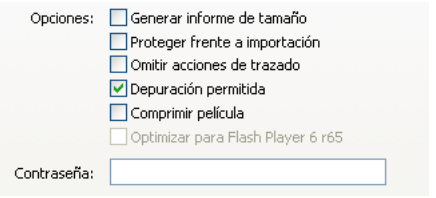
Puede depurar un archivo SWF remoto utilizando la versión independiente, ActiveX o de plugin de Flash Player. Puede encontrar estas versiones de Flash Player en el siguiente directorio de Windows o Macintosh: *Directorio de instalación de Flash*\Players\Debug\.

Cuando exporte un archivo SWF, puede activar la depuración en el archivo y crear una contraseña de depuración. Si la depuración no se activa, el depurador no se activará.

Para asegurarse de que sólo los usuarios de confianza puedan ejecutar sus archivos SWF en el reproductor de depuración de Flash, publique el archivo con una contraseña de depuración. Del mismo modo que ocurre con JavaScript o HTML, los usuarios pueden ver variables de cliente en ActionScript. Para guardar variables de un modo seguro, debe enviarlas a una aplicación de servidor en lugar de almacenarlas en su archivo. No obstante, como creador de Flash, podría tener otros secretos comerciales, como estructuras de clips de película, que no desea revelar. Puede utilizar una contraseña de depuración para proteger su trabajo.

### Para activar la depuración remota de un archivo SWF:

1. Seleccione Archivo > Configuración de publicación.
2. En la ficha Flash del cuadro de diálogo Configuración de publicación, seleccione Depuración permitida.



3. Para establecer una contraseña, especifíquela en el cuadro Contraseña.

   Una vez que haya establecido esta contraseña, nadie podrá descargar información al depurador sin ella. Sin embargo, si deja vacío el campo de contraseña, no será necesaria ninguna contraseña.

4. Cierre el cuadro de diálogo Configuración de publicación y seleccione uno de los comandos siguientes:

   - Control > Depurar película
   - Archivo > Exportar > Exportar película
   - Archivo > Publicar

   Flash crea un archivo de depuración con la extensión .swd y lo guarda en el mismo directorio que el archivo SWF. El archivo SWD se utiliza para depurar ActionScript y contiene información que le permite utilizar puntos de corte y desplazarse por el código.

5. Coloque el archivo SWD en el mismo directorio que el archivo SWF en el servidor.

   Si el archivo SWD no está en el mismo directorio que el archivo SWF, podrá igualmente depurar de forma remota, pero el depurador no dispondrá de información de puntos de corte y no podrá desplazarse paso a paso por el código.

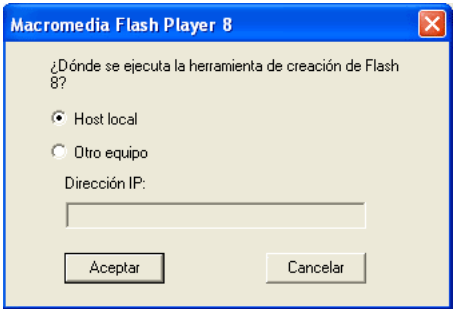
6. En Flash, seleccione Ventana > Depurador.

7. En el depurador, seleccione Habilitar depuración remota en el menú emergente (en la parte superior derecha del panel).

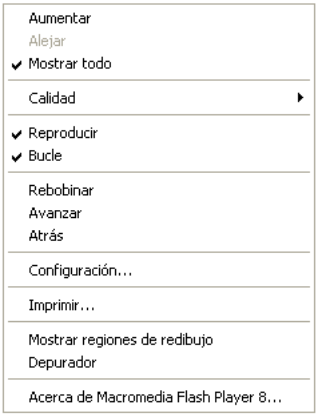
### Para activar el depurador desde una ubicación remota:

1. Abra la aplicación de edición de Flash.
2. En un navegador o en la versión de depuración del reproductor independiente, abra el archivo SWF publicado desde la ubicación remota.

   Aparecerá el cuadro de diálogo de depuración remota.



| NOTA | Si no aparece el cuadro de diálogo de depuración remota es porque Flash no ha podido encontrar el archivo SWD. En tal caso, haga clic con el botón derecho del ratón (Windows) o con la tecla Control presionada (Macintosh) en el archivo SWF para ver el menú contextual y seleccione Depurador. |
|---|---|




3. En el cuadro de diálogo de depuración remota, seleccione Host local u Otro equipo:
   - Seleccione la primera opción si el reproductor de depuración y la aplicación de edición de Flash están instalados en el mismo equipo.
   - Seleccione la segunda opción si el reproductor de depuración y la aplicación de edición de Flash no están instalados en el mismo equipo. Introduzca la dirección IP del equipo en el que se ejecuta la aplicación de edición de Flash.
4. Al establecer una conexión, aparece un mensaje de solicitud de contraseña.

   Introduzca la contraseña de depuración, si ha establecido una.

   La lista de visualización del archivo SWF aparecerá en el depurador. Si el archivo SWF no se reproduce, es posible que el depurador esté en estado de pausa, de manera que haga clic en Continuar para iniciarlo.

## Visualización y modificación de variables

La ficha Variables del depurador muestra los nombres y los valores de las variables globales y de línea de tiempo del archivo SWF que están seleccionadas en la lista de visualización. Si se modifica el valor de una variable en la ficha Variables, el cambio se reflejará en el archivo SWF mientras éste se ejecuta. Por ejemplo, para comprobar la detección de una colisión en un juego, puede introducir el valor de la variable para situar una pelota en el lugar correcto junto a una pared.

En la ficha Locales del depurador se muestran los nombres y los valores de las variables locales que están disponibles en la línea de ActionScript donde el archivo SWF está detenido actualmente, ya sea en un punto de corte o en cualquier otro lugar dentro de una función definida por el usuario.

### Para ver una variable:

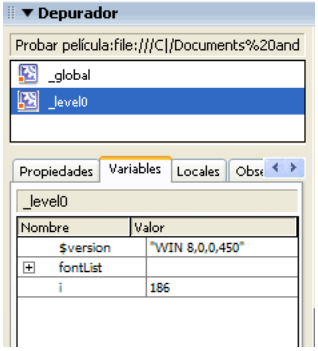
1. Seleccione el clip de película que contiene la variable en la lista de visualización.

   Para mostrar las variables globales, seleccione el clip `_global` de la lista de visualización.

| NOTA | Si es necesario, puede cambiar el tamaño de las diferentes regiones del panel Depurador. Cuando el puntero cambia entre cada región, puede arrastrarlo para cambiar el tamaño de las listas de Visualización y Observación y de la vista de código. |
|---|---|

2. Haga clic en la ficha Variables.

La lista de visualización se actualizará automáticamente según se vaya reproduciendo el archivo SWF. Si se elimina del archivo SWF de un clip de película en un fotograma determinado, dicho clip de película se eliminará también de la lista de visualización del depurador, junto con su variable y el nombre de la variable. Sin embargo, si marca una variable para la lista Observación (consulte "Utilización de la lista Observación" en la página 760), la variable se elimina de la ficha Variables, pero puede verse todavía en la ficha Observación.



**Para modificar el valor de una variable:**

- Haga doble clic en el valor y especifique otro.

El valor no puede ser una expresión. Por ejemplo, puede utilizar `"Hola"`, `3523` o `"http://www.macromedia.com"`, pero no puede utilizar `x + 2` ni `eval("name:" +i)`. El valor puede ser una cadena (cualquier valor entre comillas [""]), un número o un valor booleano (`true` o `false`).

| NOTA | Para escribir el valor de una expresión en el panel Salida en modo de prueba, utilice la sentencia `trace`. Consulte "Utilización de la sentencia trace" en la página 772. |
|---|---|

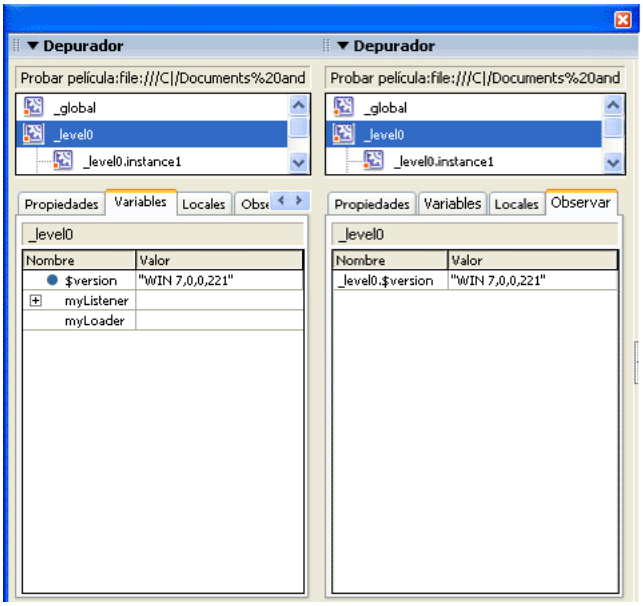
# Utilización de la lista Observación

Para controlar un conjunto de variables críticas de un modo razonable, debe marcar las variables que desee que aparezcan en la lista Observación. La lista Observación muestra la ruta absoluta de la variable, así como su valor. También puede introducir un nuevo valor de variable en la lista Observación del mismo modo que puede hacerlo en la ficha Variables. La lista Observación sólo puede mostrar variables y propiedades a las que puede acceder utilizando una ruta de destino absoluta, como `_global` o `_root`.

Si añade una variable local a la lista Observación, su valor sólo aparece cuando Flash Player se detiene en una línea de ActionScript donde la variable se encuentra dentro de su ámbito. Todas las demás variables aparecen mientras se reproduce el archivo SWF. Si el depurador no encuentra el valor de la variable, éste aparece en la lista como no definido.

| NOTA | Si es necesario, puede cambiar el tamaño de las diferentes regiones del panel Depurador. Cuando el puntero cambia entre cada región, puede arrastrarlo para cambiar el tamaño de las listas de Visualización y Observación y de la vista de código. |
|---|---|

En la lista Observación sólo pueden mostrarse variables, no propiedades ni funciones.




*Variables marcadas para la lista Observación y variables de la lista Observación*

### Utilice uno de los siguientes procedimientos para añadir variables a la lista Observación:

- En la ficha Variables o Locales, haga clic con el botón derecho del ratón (Windows) o con la tecla Control presionada (Macintosh) en una variable seleccionada y elija Observación en el menú contextual. Aparecerá un punto azul junto a la variable.
- En la ficha Observación, haga clic con el botón derecho del ratón (Windows) o mantenga presionada la tecla Control (Macintosh) y seleccione Añadir del menú contextual. Haga doble clic en la columna de nombres e introduzca la ruta de destino para el nombre de la variable que se encuentra en el campo.

### Para eliminar variables de la lista Observación:

- En las fichas Observación o Variables, haga clic con el botón derecho del ratón (Windows) o con la tecla Control presionada (Macintosh) y seleccione Quitar en el menú contextual.

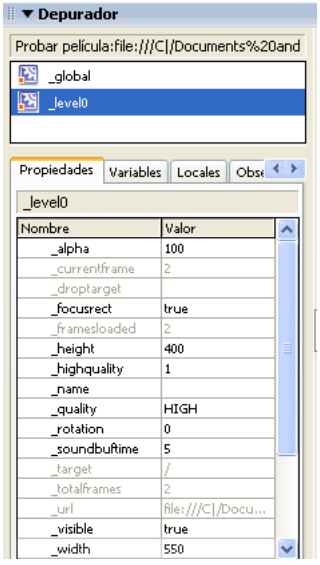
# Visualización de las propiedades de un clip de película y modificación de las propiedades editables

La ficha Propiedades del depurador muestra todos los valores de propiedades de los clips de película del escenario. Puede modificar un valor y ver su efecto en el archivo SWF mientras se ejecuta. Algunas propiedades de los clips de película son de sólo lectura y no pueden modificarse.

| NOTA | Si es necesario, puede cambiar el tamaño de las diferentes regiones del panel Depurador. Cuando el puntero cambia entre cada región, puede arrastrarlo para cambiar el tamaño de las listas de Visualización y Observación y de la vista de código. |
|---|---|

**Para ver las propiedades de un clip de película en el depurador:**

1. Seleccione un clip de película de la lista de visualización.
2. Haga clic en la ficha Propiedades del depurador.



**Para modificar el valor de una propiedad:**

- Haga doble clic en el valor y especifique otro.

  El valor no puede ser una expresión. Por ejemplo, puede introducir **50** o **"clearwater"**, pero no **x + 50**. El valor puede ser una cadena (cualquier valor entre comillas [""]), un número o un valor booleano (`true` o `false`). No se permiten valores de objeto o matriz (por ejemplo, `{id: "rogue"}` o `[1, 2, 3]`) en el depurador.

| NOTA | Para escribir el valor de una expresión en el panel Salida en modo de prueba, utilice la sentencia `trace`. Consulte "Utilización de la sentencia trace" en la página 772. |
|---|---|

# Establecimiento y eliminación de puntos de corte

Los puntos de corte permiten detener una aplicación de Flash que se está ejecutando en el Reproductor de depuración de Flash en una línea específica del código ActionScript. Puede utilizar puntos de corte para probar posibles puntos problemáticos en el código. Por ejemplo, si ha escrito un conjunto de sentencias `if..else if` y no puede determinar cuál se está ejecutando, añada un punto de corte delante de las sentencias y examínelas de una en una en el depurador.

Puede establecer puntos de corte en el panel Acciones, en la ventana Script o en el depurador. Los puntos de corte establecidos en el panel Acciones se guardan con el archivo FLA. Los puntos de corte establecidos en el depurador y en la ventana Script no se guardan en el archivo FLA y sólo son válidos para la sesión de depuración actual.

| ATENCIÓN | Si establece puntos de corte en el panel Acciones o la ventana Script y presiona el botón Formato automático, puede que observe que algunos puntos de corte ya no están en la posición correcta. Es posible que el código ActionScript se mueva a otra línea diferente al aplicar formato al código, ya que algunas veces se eliminan líneas vacías. Puede que deba comprobar y modificar los puntos de corte después de hacer clic en Formato automático, o bien aplicar formato automático a los scripts antes de establecer los puntos de corte. |
|---|---|

**Para establecer o eliminar un punto de corte del panel Acciones o la ventana Script durante una sesión de depuración, realice una de las acciones siguientes:**

- Haga clic en el margen izquierdo. Un punto rojo indica un punto de corte.
- Haga clic en el botón Opciones de depuración situado encima del panel Script.

- Haga clic con el botón derecho del ratón (Windows) o con la tecla Control presionada (Macintosh) para ver el menú contextual y seleccione Definir punto de corte, Quitar punto de corte o Quitar puntos de corte de este archivo.

| NOTA | En la ventana Script, también puede seleccionar Quitar puntos de corte de todos los archivos AS. |
|---|---|

- Presione las teclas Ctrl+Mayús+B (Windows) o Comando+Mayúsculas+B (Macintosh).

| NOTA | En algunas versiones anteriores de Flash, al hacer clic en el margen izquierdo del panel Script se seleccionaba la línea de código; ahora, se añade o elimina un punto de corte. Para seleccionar una línea de código, haga clic mientras presiona la tecla Control (Windows) o Comando (Macintosh). |
|---|---|

**Para establecer y quitar puntos de corte en el depurador, realice una de las acciones siguientes:**

- Haga clic en el margen izquierdo. Un punto rojo indica un punto de corte.
- Haga clic en el botón Alternar punto de corte o Quitar todos los puntos de corte que se encuentra encima de la vista de código.
- Haga clic con el botón derecho del ratón (Windows) o con la tecla Control presionada (Macintosh) para ver el menú contextual y seleccione Definir punto de corte, Quitar punto de corte o Quitar puntos de corte de este archivo.
- Presione las teclas Ctrl+Mayús+B (Windows) o Comando+Mayúsculas+B (Macintosh).

Una vez que Flash Player se haya detenido en un punto de corte, puede entrar, salir de esa línea de código o pasar por encima de ella. (Consulte "Desplazamiento por las líneas de código" en la página 766.)

Puede establecer puntos de corte en la ventana Script y mostrarlos en el depurador si éste tiene la misma ruta al archivo ActionScript que la que se abrió en la ventana Script. De forma similar, puede establecer puntos de corte en el depurador durante una sesión de depuración y mostrarlos en el archivo ActionScript si lo abre en la ventana Script.

| NOTA | No establezca puntos de corte en comentarios ni líneas vacías; si establece puntos de corte en comentarios o líneas vacías, se ignorarán. |
|---|---|

## Archivo XML de puntos de corte

Cuando trabaja con puntos de corte en un archivo de script externo en la ventana Script, el archivo AsBreakpoints.xml le permite almacenar información sobre los puntos de corte. El archivo AsBreakpoints.xml se escribe en el directorio de Configuración local, en las siguientes ubicaciones:

Windows:

*Disco duro*\Documents and Settings*usuario*\Configuración local\Datos de programa\Macromedia\Flash 8*idioma*\Configuration\Debugger\

Macintosh:

*Disco duro de Macintosh*/Users/*User*/Library/Application Support/Macromedia Flash 8/Configuration/Debugger/

Un ejemplo del archivo AsBreakpoints.xml es el siguiente:

```
<?xml version="1.0"?>
<flash_breakpoints version="1.0">
  <file name="c:\tmp\myscript.as">
    <breakpoint line="10"></breakpoint>
    <breakpoint line="8"></breakpoint>
    <breakpoint line="6"></breakpoint>
  </file>
  <file name="c:\tmp\myotherscript.as">
    <breakpoint line="11"></breakpoint>
    <breakpoint line="7"></breakpoint>
    <breakpoint line="4"></breakpoint>
  </file>
</flash_breakpoints>
```

El archivo XML contiene las etiquetas siguientes:

**flash_breakpoints** Este nodo contiene un atributo, denominado `version`, que indica la versión del archivo XML. Flash 8 es la versión 1.0.

**file** Un nodo secundario de `flash_breakpoints`. Este nodo incluye un atributo llamado `name`, que indica el nombre del archivo que contiene los puntos de corte.

**breakpoint** Un nodo secundario de `file`. Este nodo incluye un atributo denominado `line`, que indica el número de línea donde se encuentra el punto de corte.

El archivo AsBreakpoints.xml se lee cuando se inicia Flash y se vuelve a generar cuando se cierra Flash. AsBreakpoints.xml se utiliza para realizar un seguimiento de los puntos de corte entre las sesiones de desarrollo de Flash. Una estructura de datos interna mantiene los puntos de corte a medida que se establecen y eliminan durante el desarrollo en Flash.

# Desplazamiento por las líneas de código

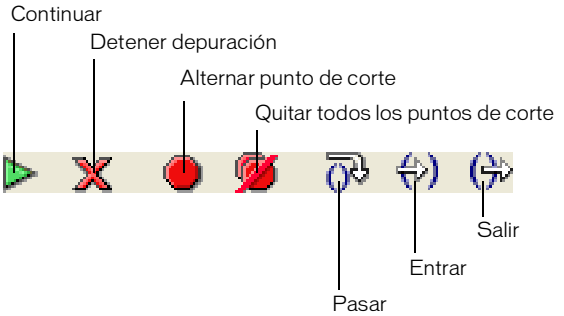
Al iniciar una sesión de depuración, se realiza una pausa en Flash Player para que pueda alternar entre los puntos de corte. Si define puntos de corte en el panel Acciones, puede hacer clic en Continuar para reproducir el archivo SWF hasta que llegue a un punto de corte. Si no se han definido puntos de corte en el panel Acciones, puede utilizar el menú de salto del depurador para seleccionar el script que desee del archivo SWF. Tras seleccionar un script, puede añadirle puntos de corte.

Tras añadir puntos de corte, deberá hacer clic en Continuar para iniciar el archivo SWF. El depurador se detiene al llegar al punto de corte. Por ejemplo, en el código siguiente, supongamos que se define un punto de corte dentro de un botón en la línea `myFunction()`:

```
on(press){
  myFunction();
}
```

Al hacer clic en el botón, se alcanza el punto de corte y Flash Player efectúa una pausa. A continuación, puede llevar el depurador a la primera línea de la función `myFunction()`, sin importar en qué punto del documento está definida. También puede continuar hasta el final de la función o salir de ésta.

A medida que se desplaza por las líneas de código, los valores de las variables y las propiedades cambian en la lista Observación y en las fichas Variables, Locales y Propiedades. La flecha amarilla en la parte izquierda de la vista de código del depurador indica la línea en la que se ha detenido el depurador. Utilice los botones siguientes situados en la parte superior de la vista de código:




**Entrar** hace que el depurador (indicado por la flecha amarilla) entre en una función. Entrar sólo funciona para las funciones definidas por el usuario.

En el ejemplo siguiente, si coloca un punto de corte en la línea 7 y hace clic en Entrar, el depurador avanza a la línea 2 y al hacer otro clic en Entrar se avanza hasta la línea 3. Al hacer clic en Entrar para líneas que no contienen funciones definidas por el usuario, el depurador pasa una línea de código. Por ejemplo, si se detiene en la línea 2 y selecciona Entrar, el depurador avanzará hasta la línea 3, como se muestra en el ejemplo siguiente:

```
1 function myFunction() {
2 x = 0;
3 y = 0;
4 }
5
6 mover = 1;
7 myFunction();
8 mover = 0;
```

| NOTA | Los números de este fragmento de código indican números de línea. No forman parte del código. |
|---|---|

**Salir** hace avanzar el depurador hasta salir de una función. Este botón sólo funciona si el depurador está detenido en una función definida por el usuario; desplaza la flecha amarilla hasta la línea posterior a la línea desde donde se llamó a la función. En el ejemplo anterior, si coloca un punto de corte en la línea 3 y hace clic en Salir, el depurador pasa a la línea 8. Cuando se hace clic en una línea que no se encuentra en una función definida por el usuario, se produce el mismo resultado que cuando se hace clic en Continuar. Por ejemplo, si se detiene en la línea 6 y hace clic en Salir, el reproductor continúa ejecutando el script hasta que encuentra un punto de corte.

**Pasar** hace avanzar el depurador pasando una línea de código. Este botón mueve la flecha amarilla a la siguiente línea del script. En el ejemplo anterior, si se detiene en la línea 7 y hace clic en Pasar, avanzará directamente a la línea 8 sin pasar por `myFunction()`, aunque continúe ejecutándose `myFunction()`.

**Continuar** deja la línea en la que se ha detenido el reproductor y prosigue la reproducción hasta llegar a un punto de corte.

**Detener depuración** desactiva el depurador, pero hace que el archivo SWF siga reproduciéndose en Flash Player.

# Utilización del panel Salida

En el modo de prueba, el panel Salida muestra información que ayuda a solucionar problemas relacionados con el archivo SWF. Cierta información (como los errores sintácticos) aparece automáticamente. Puede mostrar otro tipo de información mediante los comandos Mostrar objetos y Mostrar variables. (Consulte "Lista de los objetos de un archivo SWF" en la página 770 y "Lista de las variables de un archivo SWF" en la página 770).

Si se utiliza la sentencia `trace` en los scripts, puede enviar información específica al panel Salida durante la ejecución del archivo SWF. Por ejemplo, podrían enviarse notas acerca del estado del archivo SWF o el valor de una expresión. (Consulte "Utilización de la sentencia trace" en la página 772.)

### Para mostrar u ocultar el panel Salida, utilice uno de los siguientes procedimientos:

- Seleccione Ventana > Salida.
- Presione F2.

Para trabajar con el contenido del panel Salida, haga clic en el menú emergente situado en la esquina superior derecha para ver las opciones.

En la siguiente tabla se enumeran las opciones disponibles en el menú emergente del panel Salida:

| Elemento de menú | Lo que hace |
|---|---|
| Ajustar texto | Activa o desactiva el ajuste automático del texto de líneas largas para que el usuario no tenga que utilizar la barra de desplazamiento horizontal con el fin de ver la línea completa de caracteres. Si está seleccionada, se ajusta el texto de las líneas; en caso contrario, el texto de las líneas no se ajusta. |
| Copiar | Copia todo el contenido del panel Salida al portapapeles del equipo. Para copiar una parte seleccionada de la salida, seleccione el área que desea copiar y luego seleccione Copiar. |
| Borrar | Borra toda la salida actualmente existente en el panel Salida. |
| Buscar | Abre un cuadro de diálogo que le permite buscar una palabra clave o una frase en el contenido del panel Salida. |
| Buscar de nuevo | Intenta localizar la siguiente instancia de una palabra clave o frase en el contenido del panel Salida. |
| Guardar en archivo | Guarda el contenido actual del panel Salida en un archivo de texto externo. |
| Imprimir | Muestra el cuadro de diálogo Imprimir, que le permite imprimir el contenido actual del panel Salida en una impresora instalada o en programas instalados como Flash Paper o Acrobat. |
| Nivel de filtro | Le permite seleccionar entre dos niveles de salida posibles: Ninguno o Detalles. La selección de Ninguno suprime toda la salida enviada al navegador. |
| Maximizar panel | Maximiza el panel Salida cuando está acoplado. |
| Cerrar panel | Cierra el panel Salida y borra el contenido de dicho panel. |

Para más información sobre el panel Salida, consulte los siguientes temas:

- "Lista de los objetos de un archivo SWF" en la página 770
- "Lista de las variables de un archivo SWF" en la página 770
- "Visualización de las propiedades de un campo de texto para la depuración" en la página 772
- "Utilización de la sentencia trace" en la página 772
- "Actualización de Flash Player para realizar pruebas" en la página 773

# Lista de los objetos de un archivo SWF

En el modo de prueba, el comando Mostrar objetos muestra el nivel, el fotograma, el tipo de objeto (forma, clip de película o botón), las rutas de destino y los nombres de instancia de los clips de película, los botones y los campos de texto en una lista jerárquica. Esta opción resulta especialmente útil para encontrar el nombre de instancia y la ruta de destino correctas. A diferencia del depurador, la lista no se actualiza automáticamente a medida que se reproduce el archivo SWF; es necesario seleccionar el comando Mostrar objetos cada vez que se desee enviar la información al panel Salida.

| ATENCIÓN | La selección del comando Mostrar objetos borra toda la información que se muestra actualmente en el panel Salida. Si no desea perder la información del panel Salida, seleccione Guardar en archivo del menú emergente Opciones del panel Salida o copie y pegue la información en otro lugar antes de seleccionar el comando Mostrar objetos. |
|---|---|

El comando Mostrar objetos no enumera todos los objetos de datos de ActionScript. En este contexto, el objeto se considera como una forma o símbolo en el escenario.

### Para mostrar una lista de los objetos de un archivo SWF:

1. Si el archivo SWF no se está ejecutando en el modo de prueba, seleccione Control > Probar película.
2. Seleccione Depurar > Mostrar objetos.

   En el panel Salida se mostrará una lista de todos los objetos que se encuentren en el escenario, como en el ejemplo siguiente:

```
Level #0: Frame=1 Label="Scene_1"
  Button: Target="_level0.myButton"
    Shape:
  Movie Clip: Frame=1 Target="_level0.myMovieClip"
    Shape:
  Edit Text: Target="_level0.myTextField" Text="This is sample text."
```

# Lista de las variables de un archivo SWF

En el modo de prueba, el comando Mostrar variables muestra una lista de todas las variables existentes actualmente en el archivo SWF. Esta lista resulta especialmente útil para encontrar el nombre de la variable y la ruta de destino correctos. A diferencia del depurador, la lista no se actualiza automáticamente a medida que se reproduce el archivo SWF; es necesario seleccionar el comando Mostrar variables cada vez que se desee enviar la información al panel Salida.

El comando Mostrar variables también muestra las variables globales declaradas con el identificador `_global`. Las variables globales aparecen en la parte superior de la ventana de resultados de Mostrar variables en una sección titulada Variables globales; cada variable lleva un prefijo `_global`.

Además, el comando Mostrar variables muestra propiedades getter/setter, propiedades creadas con el método `Object.addProperty()` y que inician los métodos `get` (captador) o `set` (definidor). Aparece una propiedad getter/setter junto a las demás propiedades del objeto al que pertenecen. Para distinguir fácilmente estas propiedades de otras variables, el valor de una propiedad getter/setter aparece precedido por la cadena `[getter/setter]`. El valor que se muestra para una propiedad getter/setter se determina calculando el resultado de la función `get` de la propiedad.

**ATENCIÓN**

La selección del comando Mostrar variables borra toda la información que se muestra en el panel Salida. Si no desea perder la información del panel Salida, seleccione Guardar en archivo del menú emergente Opciones del panel Salida o copie y pegue la información en otro lugar antes de seleccionar el comando Mostrar variables.

### Para mostrar una lista de las variables de un archivo SWF:

1. Si el archivo SWF no se está ejecutando en el modo de prueba, seleccione Control > Probar película.
2. Seleccione Depurar > Mostrar variables.

   En el panel Salida aparece una lista de todas las variables existentes actualmente en el archivo SWF, como en el ejemplo siguiente:

```
Global Variables:
  Variable _global.mycolor = "lime_green"
Level #0:
Variable _level0.$version = "WIN 7,0,19,0"
Variable _level0.myArray = [object #1, class 'Array'] [
    0:"socks",
    1:"gophers",
    2:"mr.claw"
  ]
Movie Clip: Target="_level0.my_mc"
```

## Visualización de las propiedades de un campo de texto para la depuración

Para obtener información de depuración sobre los objetos TextField, utilice el comando Depurar > Mostrar variables en modo de prueba. El panel Salida utiliza las siguientes convenciones al mostrar los objetos TextField:

- Si no se encuentra una propiedad en el objeto, no se muestra.
- En una línea se muestran como máximo cuatro propiedades.
- Una propiedad con un valor de cadena se muestra en una línea aparte.
- Si hay otras propiedades definidas para el objeto una vez que se han procesado las propiedades incorporadas, éstas se añaden a la pantalla siguiendo las reglas de los puntos segundo y tercero de esta lista.
- Las propiedades de color aparecen como números hexadecimales (0x00FF00).
- Las propiedades aparecen en el orden siguiente: `variable`, `text`, `htmlText`, `html`, `textWidth`, `textHeight`, `maxChars`, `borderColor`, `backgroundColor`, `textColor`, `border`, `background`, `wordWrap`, `password`, `multiline`, `selectable`, `scroll`, `hscroll`, `maxscroll`, `maxhscroll`, `bottomScroll`, `type`, `embedFonts`, `restrict`, `length`, `tabIndex`, `autoSize`.

El comando Mostrar objetos del menú Depurar (en modo de prueba) enumera los objetos TextField. Si se especifica un nombre de instancia para un campo de texto, en el panel Salida se mostrará la ruta de destino completa, incluido el nombre de instancia, con el formato siguiente:

```
Target = "target path"
```

Para más información sobre el comando Mostrar variables o Mostrar objetos, consulte "Utilización del panel Salida" en la página 768.

## Utilización de la sentencia trace

Cuando se utiliza la sentencia `trace` en un script, puede enviarse información al panel Salida. Por ejemplo, cuando se prueba un archivo SWF o una escena, pueden enviarse notas de programación determinadas al panel o hacer que aparezcan resultados concretos cuando se presiona un botón o se reproduce un fotograma. La sentencia `trace` es similar a la sentencia `alert` de JavaScript.

El uso de la sentencia `trace` en un script permite utilizar expresiones como parámetros. El valor de una expresión aparece en el panel Salida en modo de prueba, como se muestra en el siguiente fragmento de código e imagen del panel Salida.

### Para utilizar la sentencia trace en un script:

1. Seleccione el fotograma 1 de la línea de tiempo y añada el siguiente código en el panel Acciones:

```
this.createEmptyMovieClip("img_mc", 10);
var mclListener:Object = new Object();
mclListener.onLoadInit = function(target_mc:MovieClip) {
  trace(target_mc+" loaded in "+getTimer()+" ms");
};
mclListener.onLoadError = function(target_mc:MovieClip,
  errorCode:String, httpStatus:Number) {
  trace(">> error downloading image into "+target_mc);
  trace(">>\t errorCode="+errorCode+", httpStatus="+httpStatus);
};
var img_mcl:MovieClipLoader = new MovieClipLoader();
img_mcl.addListener(mclListener);
img_mcl.loadClip("http://www.helpexamples.com/flash/images/404.jpg",
  img_mc);
```

2. Seleccione Control > Probar película para probar el archivo SWF.

   El panel Salida muestra el siguiente mensaje:


## Actualización de Flash Player para realizar pruebas

Puede descargar la última versión de Flash Player del Centro de servicio técnico de Flash de Macromedia en http://www.macromedia.com/go/flash_support_es y utilizarla para probar los archivos SWF.

CAPÍTULO 19

# Recomendaciones y convenciones de codificación para ActionScript 2.0

19

Los diseñadores y desarrolladores que utilicen Macromedia Flash deberán escribir código y estructurar las aplicaciones de forma intuitiva y útil tanto para ellos como para otras personas que trabajen en el mismo proyecto. Esto es especialmente importante en el caso de archivos FLA con muchos activos o en archivos de código grandes. Cuando se siguen las recomendaciones y convenciones de codificación, el equipo de diseño y desarrollo puede comprender la estructura de archivos y el código ActionScript funciona de forma más eficaz. Este documento ayuda a dar forma al proceso de desarrollo y codificación de Flash.

Dado que es habitual que más de un diseñador o desarrollador trabaje en un mismo proyecto de Flash, los equipos resultan beneficiados cuando todos sus miembros siguen un conjunto estándar de directrices para utilizar Flash, organizar archivos FLA y escribir código ActionScript 2.0. En las secciones de este capítulo se describen las prácticas recomendadas para escribir código ActionScript, mientras que en algunas secciones de *Utilización de Flash* se abordan las recomendaciones para el uso de la herramienta de edición de Flash.

El seguimiento de estas directrices también contribuye a que las personas que estén aprendiendo a utilizar Flash y a escribir código ActionScript logren una mayor coherencia. Las prácticas recomendadas deben adoptarlas en todo momento los diseñadores, desarrolladores o el personal que trabaje individualmente o como parte de un equipo.

- Cuando trabaja con documentos de Flash o ActionScript

  La adopción de unas prácticas coherentes y eficaces le ayudará a acelerar el flujo de trabajo. Si se siguen las convenciones de codificación establecidas, el desarrollo es más rápido y resulta más sencillo comprender y recordar la estructura del documento cuando es preciso editarlo posteriormente. Además, el código suele ser más portable y más fácil de reutilizar dentro del marco de un proyecto de mayor tamaño.

- Cuando comparte archivos FLA o AS

  Otras personas que editen el documento pueden encontrar y comprender rápidamente el código ActionScript, modificar código de forma coherente, y encontrar y editar activos.

- Cuando trabaja con aplicaciones

  Varios autores pueden trabajar en una aplicación con menos conflictos y más eficacia. Si se siguen las prácticas recomendadas y las convenciones de codificación, los administradores de sitios o de proyectos podrán administrar y estructurar proyectos o aplicaciones complejos con menos conflictos o redundancias.
- Cuando aprende o enseña Flash y ActionScript

  Aprender a crear aplicaciones utilizando las prácticas recomendadas y siguiendo las convenciones de código reduce la necesidad de tener que aprender nuevamente determinadas metodologías. Si los estudiantes que reciben formación de Flash utilizan formas coherentes y mejores para estructurar el código, su aprendizaje del lenguaje puede ser más rápido y menos frustrante.

El uso de técnicas coherentes y de las siguientes directrices sirve de ayuda a las personas que reciben formación de Flash o a las que trabajan eficazmente en entornos de equipo. La coherencia de los métodos permite que las personas que trabajan solas puedan recordar la estructura de un documento, sobre todo si no han trabajado recientemente en el archivo FLA.

Éstas son sólo algunas de las razones para aprender y seguir las prácticas recomendadas. Hay muchas otras razones que sin duda descubrirá cuando lea estas prácticas recomendadas y desarrolle sus propios buenos hábitos. Considere los siguientes temas como directrices para trabajar con Flash; puede seguir algunas de las recomendaciones o todas ellas. También puede modificar las recomendaciones y adaptarlas a su forma de trabajar. Muchas de las directrices de este capítulo le ayudarán a desarrollar una forma coherente de trabajar con Flash y escribir código ActionScript.

En este capítulo se tratan los siguientes temas sobre convenciones de codificación y prácticas recomendadas:

# Convenciones de asignación de nombre

Normalmente, el 80% del tiempo de desarrollo se emplea en depurar, solucionar problemas y realizar tareas de mantenimiento general, especialmente en proyectos de gran tamaño. Incluso cuando se trabaja en proyectos pequeños se suele emplear una cantidad de tiempo considerable en analizar y corregir el código. La legibilidad del código es importante para su propio beneficio y el de los miembros del equipo. Al seguir las convenciones de asignación de nombre, aumenta la legibilidad, lo que acelera el flujo de trabajo y le permite encontrar y resolver cualquier error existente en el código. Todos los programadores siguen una forma estandarizada de escribir código, lo cual mejora el proyecto en muchos aspectos.

La utilización de convenciones de asignación de nombre a las variables puede contribuir a funciones importantes como las siguientes:

- Mejoran la legibilidad del código, de forma que es posible identificar de inmediato el tipo de datos de una variable. Esto puede ayudar a los alumnos, a aquellas personas que estén aprendiendo a usar el código o a desarrolladores que no estén familiarizados con el código.
- Son fáciles de buscar y reemplazar si es preciso.
- Contribuyen a reducir los conflictos con palabras reservadas y construcciones del lenguaje.
- Pueden servir de ayuda para distinguir entre variables de diferentes ámbitos (variables locales, propiedades de clase, parámetros, etc.).

Las siguientes secciones contienen las directrices para la asignación de nombres al escribir código ActionScript como, por ejemplo, la asignación de nombres a archivos, variables, constantes, componentes, etc. En "Aplicación de formato a la sintaxis de ActionScript" en la página 808 se describen las convenciones de formato específicas de ActionScript y habituales en otros lenguajes de programación. En "Convenciones de codificación de ActionScript" en la página 790 se abordan las convenciones de codificación específicas para la escritura de código ActionScript y el desarrollo con Flash 8.

| NOTA | Flash Player 7 y 8 siguen la especificación del lenguaje ECMAScript (ECMA-262) edición 3, aunque no la cumplen de manera estricta. Resulta útil consultar esta especificación para saber cómo funciona el lenguaje. (Consulte www.ecma-international.org/publications/standards/Ecma-262.htm.) |
|---|---|

Esta sección contiene los siguientes temas:

## Directrices generales para la asignación de nombre

En esta sección se describen las directrices para la asignación de nombre al escribir código ActionScript. Las convenciones de asignación de nombres son importantes para conseguir que el código sea lógico. El objetivo principal es mejorar la legibilidad del código ActionScript 2.0. Recuerde que todas las variables deben tener nombres exclusivos. Flash Player 7 y versiones posteriores distinguen entre mayúsculas y minúsculas para los nombres. No utilice el mismo nombre con diferencias de mayúsculas y minúsculas, ya que ello puede confundir a los programadores que lean el código y puede generar problemas en versiones anteriores de Flash que no exigen distinción entre mayúsculas y minúsculas. Tenga presentes las siguientes directrices cuando asigne nombre a elementos tales como variables, archivos y clases de Flash:

- Limite el uso de abreviaturas.

  Utilice las abreviaturas de forma coherente. Una abreviatura debe referirse claramente a una sola cosa. Por ejemplo, la abreviatura "sec" puede representar tanto "section" (sección) como "second" (segundo).
- Concatene palabras para crear nombres.

  Utilice mayúsculas y minúsculas cuando concatene palabras para distinguir cada una de las palabras y facilitar la legibilidad. Por ejemplo, escriba `myPelican` en lugar de `mypelican`.
- Asigne nombres a los archivos de forma que se describa el proceso o elemento; por ejemplo, `addUser`.
- No utilice nombres no descriptivos para los métodos o las variables.

  Por ejemplo, si recupera un dato que es el nombre de usuario del visitante, podría utilizar el método `getUserName()` en lugar de `getData()`, que es menos descriptivo. Este ejemplo expresa lo que ocurre y no el modo en que se produce.
- Elija unos nombres tan cortos como sea posible.

  Recuerde que los nombres deberán ser descriptivos.

En las siguientes secciones se ofrecen más detalles sobre la asignación de nombre a elementos tales como variables, clases, paquetes y constantes del código.

## Cómo evitar la utilización de palabras reservadas o construcciones del lenguaje

Al asignar nombre a instancias y variables, evite utilizar palabras reservadas, ya que éstas pueden provocar errores en el código. Entre las palabras reservadas se encuentran las *palabras clave* del lenguaje ActionScript.

Tampoco deberá utilizar palabras del lenguaje ActionScript 2.0 (llamadas *construcciones del lenguaje*) como nombres de instancias o variables. Entre las construcciones de ActionScript se encuentran los nombres de clases, los nombres de clases de componentes, los nombres de métodos y propiedades y los nombres de interfaces.

**ADVERTENCIA**

No utilice nunca mayúsculas y minúsculas diferentes para evitar conflictos con las palabras reservadas. Por ejemplo, la asignación del nombre `textfield` a una instancia de la clase TextField (con la cual no entra en conflicto porque Flash distingue entre mayúsculas y minúsculas) es una práctica de codificación poco recomendable.

En la siguiente tabla se enumeran las palabras clave reservadas de ActionScript 2.0 que provocan errores en los scripts cuando se utilizan como nombres de variables:

| | | | |
|---|---|---|---|
| add | and | break | case |
| catch | class | continue | default |
| delete | do | dynamic | else |
| eq | extends | false | finally |
| for | function | ge | get |
| gt | if | ifFrameLoaded | implements |
| import | in | instanceof | interface |
| intrinsic | le | it | ne |
| new | not | null | on |
| onClipEvent | or | private | public |
| return | set | static | super |
| switch | tellTarget | this | throw |
| try | typeof | var | void |
| while | with | | |

Las siguientes palabras, tomadas del borrador de especificación del lenguaje ECMAScript (ECMA-262) edición 4, se reservan para su uso futuro en Flash. Evite utilizar estas palabras porque podrían utilizarse en futuras versiones de Flash.

| | | | |
|---|---|---|---|
| as | abstract | Boolean | bytes |
| char | const | debugger | double |
| enum | export | final | float |
| goto | is | long | namespace |
| native | package | protected | short |
| synchronized | throws | transient | use |
| volatile | | | |

## Asignación de nombre a variables

Los nombres de variables sólo pueden contener letras, números y símbolos de dólar (`$`). No empiece los nombres de variables por números. Las variables deben ser exclusivas y distinguen entre mayúsculas y minúsculas en Flash Player 7 y versiones posteriores. Evite, por ejemplo, los siguientes nombres de variables:

```
my/warthog = true;   // incluye una barra
my warthogs = true;  //incluye un espacio
my.warthogs = true; // incluye un punto
5warthogs = 55;    //empieza con un número
```

Siempre que sea posible, utilice strict data typing con las variables, pues le ayudará de las siguientes formas:

- Añade la funcionalidad de completar código, lo que permite acelerar la programación.
- Genera errores en el panel Salida, por lo que no tendrá un error sin mensaje cuando compile el archivo SWF. Estos errores le ayudan a encontrar y solucionar problemas en las aplicaciones.

Para añadir un tipo de datos a las variables, debe definir la variable con la palabra clave `var`. En el siguiente ejemplo, utilizaría strict data typing para crear un objeto LoadVars:

```
var paramsLv:LoadVars = new LoadVars();
```

Strict data typing proporciona la capacidad de completar código y garantiza que el valor de `paramsLv` contenga un objeto LoadVars. También garantiza que el objeto LoadVars no se utilice para almacenar datos numéricos o de cadena. Dado que strict typing utiliza la palabra clave `var`, no se puede añadir strict data typing a variables o propiedades globales de un objeto o matriz. Para más información sobre el uso de strict typing en las variables, consulte "Asignación de tipos de datos y "strict data typing"" en la página 337.

| NOTA | strict data typing no ralentiza un archivo SWF. La verificación de tipos se produce al compilar (cuando se crea el archivo SWF) y no en tiempo de ejecución. |
|---|---|

Utilice las siguientes directrices cuando asigne nombre a variables en el código:

- Todas las variables deben tener nombres exclusivos.
- No utilice el mismo nombre de variable con diferentes mayúsculas y minúsculas.

  No utilice, por ejemplo, `firstname` y `firstName` como variables diferentes en la aplicación. Aunque los nombres distinguen entre mayúsculas y minúsculas en Flash Player 7, no utilice el mismo nombre de variable con un uso diferente de mayúsculas y minúsculas porque puede resultar confuso para los programadores que lean el código y pueden producirse problemas en versiones anteriores de Flash que no distinguían entre mayúsculas y minúsculas.
- No utilice como nombres de variables palabras que formen parte del lenguaje ActionScript 1.0 ó 2.0.

  En concreto, no utilice nunca palabras clave como nombres de instancias, ya que provocarán errores en el código. Evite utilizar diferentes mayúsculas y minúsculas como forma de distinguir nombres para evitar conflictos y conseguir que el código funcione.
- No utilice variables que formen parte de construcciones de programación habituales.

  No utilice construcciones del lenguaje si sabe que éstas se emplean en otros lenguajes de programación, aunque Flash no incluya o admita dichas construcciones. No utilice, por ejemplo, las siguientes palabras clave como variables:

  ```
  textfield = "myTextField";
  switch = true;
  new = "funk";
  ```
- Añada siempre anotaciones de tipos de datos en el código.

  La adición de estas anotaciones de tipos en las variables, lo que también se conoce como "uso de strict data types con variables," o "comprobación de tipos al compilar variables", resulta fundamental para:

  - Generar errores durante la compilación para que la aplicación no produzca un fallo sin mensaje.
  - Activar la funcionalidad de completar código.

- Contribuir a que los usuarios comprendan el código.

Para obtener información sobre la adición de anotaciones de tipos, consulte "Asignación de tipos de datos y "strict data typing"" en la página 337.

- No haga un uso excesivo del tipo Object.

  Las anotaciones de tipos de datos deben ser precisas para mejorar el rendimiento. Sólo use un tipo Object si no hay ninguna alternativa razonable.

- Las variables deben ser claras y lo más cortas posibles.

  Asegúrese de que los nombres de las variables sean descriptivos, pero evite utilizar nombres extremadamente largos y complejos.

- Utilice nombres de variables de un solo carácter solamente para la optimización de bucles.

  Opcionalmente, puede utilizar un solo carácter como nombre de variables temporales de bucles (por ejemplo, `i`, `j`, `k`, `m` y `n`). Utilice estos nombres de variable de un solo carácter únicamente en los índices de bucles cortos o en los casos en los que es fundamental optimizar el rendimiento y la velocidad. El siguiente ejemplo muestra este uso:

```
var fontArr:Array = TextField.getFontList();
fontArr.sort();
var i:Number;
for (i = 0; i<fontArr.length; i++) {
  trace(fontArr[i]);
}
```

- Asigne a las variables nombres que empiecen por una letra minúscula.

  Los nombres con iniciales en mayúsculas están reservados para clases, interfaces, etc.

- Utilice una combinación de mayúsculas y minúsculas para las palabras concatenadas.

  Por ejemplo, utilice `myFont` en lugar de `myfont`.

- No utilice siglas ni abreviaturas.

  La excepción a esta regla es el uso de acrónimos o abreviaturas que representen la forma estándar de empleo de un término (como HTML o CFM). En el caso de siglas de uso habitual, utilice una combinación de mayúsculas y minúsculas para lograr una mayor legibilidad, como, por ejemplo, `newHtmlParser` en lugar de `newHTMLParser`.

- Utilice pares complementarios para crear un conjunto relacionado de nombres de variables.

  Por ejemplo, puede utilizar pares complementarios para indicar la puntuación mínima y máxima de un juego de la siguiente forma:

```
var minScoreNum:Number = 10; //puntuación mínima
var maxScoreNum:Number = 500; //puntuación máxima
```

## Asignación de nombre a constantes

Puede utilizar las constantes en situaciones en las que es necesario referirse a una propiedad cuyo valor es invariable. De esta forma será más sencillo encontrar errores tipográficos en el código que podrían pasar inadvertidos si utilizara literales. También podrá modificar el valor en un solo lugar.

Las variables deben utilizar una combinación de letras mayúsculas y minúsculas; no obstante, utilice las siguientes directrices de asignación de nombre a constantes estáticas (variables que no cambian):

- Las constantes deben ir en mayúsculas.
- Las palabras deben separarse mediante guiones bajos.

Puede observar el uso de estas directrices en el siguiente fragmento de código ActionScript:

```
var BASE_URL:String = "http://www.macromedia.com"; //constante
var MAX_WIDTH:Number = 10;  //constante
```

No programe directamente las constantes numéricas, a menos que la constante sea 1, 0 o -1, que es la constante que utilizaría en un bucle `for` como valor de contador.

## Asignación de nombre a variables booleanas

Asigne a las variables booleanas nombres que comiencen por la palabra “is” (ya que en inglés, por naturaleza, un valor booleano es (“is”) o no es (“is not”). Por consiguiente, podría utilizar el siguiente código para indicar si un bebé es o no niña (lo cual se representa mediante un valor booleano):

```
isGirl
```

O bien, para una variable que indique si un usuario ha iniciado una sesión o no la ha iniciado, podría utilizar lo siguiente:

```
isLoggedIn
```

## Asignación de nombre a funciones y métodos

Utilice las siguientes directrices cuando asigne nombres a funciones y métodos en el código. Para obtener información sobre la escritura de funciones y métodos, consulte el Capítulo 5, “Funciones y métodos”.

- Utilice nombres descriptivos.
- Utilice una combinación de mayúsculas y minúsculas para las palabras concatenadas. Un ejemplo de ello es `singLoud()`.
- Comience los nombres de funciones y métodos con una letra minúscula.

- En el nombre de la función describa el valor que se devuelve.

  Por ejemplo, si devuelve el nombre de un título de canción, podría denominar a la función `getCurrentSong()`.
- Establezca un estándar para relacionar funciones similares.

  ActionScript 2.0 no permite la sobrecarga. En el contexto de la programación orientada a objetos, la *sobrecarga* es la capacidad de hacer que las funciones se comporten de forma distinta según los tipos de datos que se pasen en ellas.
- Utilice verbos para denominar a los métodos.

  Puede concatenar el nombre, pero deberá contener un verbo. En la mayoría de los métodos se utilizan verbos porque ejecutan una operación en un objeto.

A continuación se incluyen algunos ejemplos de nombres de métodos:

```
sing();
boogie();
singLoud();
danceFast();
```

## Asignación de nombre a clases y objetos

Al crear un nuevo archivo de clase, siga estas directrices para asignar nombre a la clase y al archivo ActionScript. A continuación se muestran ejemplos de nombres de clase con el formato adecuado:

```
class Widget;
class PlasticWidget;
class StreamingVideo;
```

Una clase puede contener variables de miembros públicas y privadas. La clase puede contener variables para las que no desee permitir el acceso o modificación por parte de los usuarios. Convierta estas variables en privadas y permita a los usuarios acceder a los valores únicamente mediante métodos de captador/definidor.

Siga estas directrices para asignar nombre a las clases:

- Comience un nombre de clase con mayúscula.
- Escriba los nombres de clase mezclando mayúsculas y minúsculas cuando se trate de una palabra compuesta o concatenada.

  Comience con mayúscula las palabras compuestas o concatenadas. Un ejemplo de ello es `NewMember`.
- Los nombres de clase suelen ser nombres o nombres adjetivados.

  Un calificador describe el nombre o la frase. Por ejemplo, en lugar de utilizar simplemente "member", puede calificar el nombre mediante `NewMember` u `OldMember`.

- Al elegir el nombre, es más importante la claridad que la brevedad.
- No utilice siglas ni abreviaturas.

  La excepción a esta regla es el uso de acrónimos o abreviaturas que representen la forma estándar de empleo de un término (como HTML o CFM). En el caso de siglas de uso habitual, utilice una combinación de mayúsculas y minúsculas, como, por ejemplo, `newHtmlParser` en lugar de `newHTMLParser` para lograr una mayor legibilidad.
- Utilice nombres explicativos y sencillos que describan el contenido de la clase.

  Para evitar confusión o vaguedad, utilice nombres genéricos.
- A veces, el nombre de clase es una palabra compuesta.

  Un calificador podría describir el nombre o la frase. Por ejemplo, en lugar de utilizar simplemente “member”, puede calificar el nombre mediante `NewMember` u `OldMember`.
- No utilice el plural de las palabras que se usan en el nombre de clase (como `Witches` o `BaldPirates`).

  En la mayoría de los casos, es mejor dejar las palabras como nombres adjetivados. Un calificador describe el nombre o la frase. Por ejemplo, en lugar de utilizar simplemente “cat”, o "buckaneer" puede calificar el nombre mediante `BlackCat` u `OldBuckaneer`.
- No utilice un nombre de clase en las propiedades de la clase porque provocaría redundancia.

  Por ejemplo, no tiene sentido escribir `Cat.catWhiskers`. Es preferible utilizar `Cat.whiskers` en su lugar.
- No utilice nombres que puedan interpretarse como verbos.

  Por ejemplo, `Running` o `Gardening`. Estos podrían confundirse con métodos, estados u otras actividades de la aplicación.
- Utilice nombres de clase exclusivos para cada clase de una misma aplicación.
- No asigne a las clases nombres que puedan entrar en conflicto con los de las clases incorporadas de Flash.
- Intente comunicar la relación de una clase dentro de una jerarquía.

  Esto ayuda a mostrar la relación de la clase dentro de una aplicación. Por ejemplo, puede tener la interfaz Widget y la implementación de Widget puede ser PlasticWidget, SteelWidget y SmallWidget.

Para obtener información sobre interfaces, consulte el Capítulo 8, “Interfaces”.

## Asignación de nombre a paquetes

Es habitual que los nombres de paquetes utilicen una convención de denominación de "dominio inverso". Ejemplos de nombres de dominio inversos son `com.macromedia` para macromedia.com y `org.yourdomain` para yourdomain.org.

Siga estas directrices al asignar nombre a los paquetes:

- Escriba con letras minúsculas el prefijo de un nombre de paquete.

  Por ejemplo, `com`, `mx` u `org`.
- Coloque las clases relacionadas (clases con funcionalidad relacionada) en el mismo paquete.
- Asigne a los paquetes nombres con un prefijo coherente.

  Por ejemplo, podría utilizar `com.macromedia.nombreProyecto` para mantener la coherencia. Otro ejemplo sería `com.macromedia.docs.learnAS2.Users` para la *Referencia* del *Aprendizaje de ActionScript 2.0*.
- Utilice un nombre de paquete que sea claro y descriptivo.

  Es importante explicar las responsabilidades del paquete. Por ejemplo, si tiene un paquete llamado `Pentagons` cuyo cometido es utilizar la interfaz API de dibujo de Flash para dibujar diversos tipos de pentágonos en ejemplos de documentación, su nombre sería `com.macromedia.docs.as2.Pentagons`.
- Utilice una combinación de mayúsculas y minúsculas para nombres de paquetes compuestos o concatenados.

  `packageName` es un ejemplo de un nombre de paquete compuesto y concatenado. Recuerde que todas las letras del prefijo deben ser minúsculas (com, org, etc.).
- No utilice guiones bajos ni símbolos de dólar.

## Asignación de nombre a interfaces

Utilice una "I" mayúscula como inicial de los nombres de interfaces para distinguirlos de los nombres de clases. El nombre de interfaz `IEmployeeRecords` empieza con mayúscula y contiene palabras concatenadas donde se mezclan mayúsculas y minúsculas:

```
interface IEmployeeRecords{}
```

También se aplican las siguientes convenciones:

- Los nombres de interfaces empiezan por una letra mayúscula.

  Los nombres de clases también comienzan por mayúscula.
- Los nombres de interfaces suelen ser adjetivos.

  Por ejemplo, `Printable`.

Para más información sobre interfaces, consulte el Capítulo 8, "Interfaces"

## Asignación de nombre a componentes personalizados

Los nombres de componentes empiezan por mayúscula y mezclan mayúsculas y minúsculas si contienen palabras concatenadas. Por ejemplo, en los siguientes componentes de interfaz de usuario predeterminados se utilizan palabras concatenadas y se mezclan mayúsculas y minúsculas:

- CheckBox
- ComboBox
- DataGrid
- DateChooser
- DateField
- MenuBar
- NumericStepper
- ProgressBar
- RadioButton
- ScrollPane
- TextArea
- TextInput

Los componentes que no utilizan palabras concatenadas empiezan por una letra mayúscula.

Si desarrolla componentes personalizados, utilice una convención de asignación de nombres para evitar incompatibilidades con los nombres de los componentes de Macromedia. Los nombres de los componentes personalizados deben ser distintos de los predeterminados incluidos en Flash. Si adopta una convención de asignación de nombres propia que sea coherente, evitará conflictos en la asignación de nombres.

Recuerde que las convenciones de asignación de nombres de esta sección son sólo directrices. Lo más importante es que utilice de forma coherente un esquema de asignación de nombres que le sea útil.

# Utilización de comentarios en el código

En esta sección se describe cómo utilizar comentarios en el código. Los comentarios documentan las decisiones que se toman con respecto al código y responden a las preguntas *cómo* y *por qué*. Por ejemplo, podría describir una solución en los comentarios. Otro desarrollador podría encontrar fácilmente el código relacionado y actualizarlo o corregirlo. Y, finalmente, el problema podría resolverse en una futura versión de Flash o Flash Player, por lo que la solución ya no sería necesaria.

Para más información sobre la escritura de comentarios en el código ActionScript, consulte las siguientes secciones:

- "Escritura de comentarios adecuados" en la página 788
- "Adición de comentarios a clases" en la página 789

## Escritura de comentarios adecuados

La utilización coherente de comentarios en el código ActionScript 2.0 le permite describir áreas complejas del código o interacciones importantes que no estén demasiado claras. Los comentarios deben explicar claramente el propósito del código y no simplemente traducirlo. Si alguna parte del código no queda suficientemente clara, añada comentarios.

Si utiliza la herramienta Formato automático en el código, observará que los comentarios finales (consulte "Comentarios finales" en la página 98) se desplazan a la siguiente línea. Puede añadir estos comentarios después de aplicar formato al código pues, de lo contrario, deberá modificar su ubicación tras utilizar la herramienta Formato automático.

Para más información sobre el uso de comentarios en las clases, consulte "Adición de comentarios a clases" en la página 789.

Utilice las siguientes directrices cuando añada comentarios al código:

- Utilice comentarios en bloque (/* y */) para comentarios que ocupen varias líneas y comentarios de una sola línea (//) para comentarios breves.

  También puede utilizar un *comentario final* en la misma línea que el código ActionScript si es preciso.
- Asegúrese de que no utiliza comentarios para traducir el código ActionScript.

  No es necesario comentar los elementos que resultan obvios en el código ActionScript.
- Incluya comentarios para los elementos que no sean obvios de forma inmediata en el código.

  En concreto, deberá añadir comentarios si el asunto no se describe en los párrafos anterior o posterior.

- No utilice comentarios saturados.

  Las líneas de comentarios saturadas suelen contener signos igual (=) o asteriscos (*). En lugar de eso, utilice un espacio en blanco para separar los comentarios del código ActionScript.

  | NOTA | Si utiliza la herramienta Formato automático para aplicar formato al código ActionScript, se eliminará el espacio en blanco. Recuerde volver a añadirlo o utilice comentarios de una línea (//) para mantener el espaciado; estas líneas pueden eliminarse fácilmente tras aplicar formato al código. |
  |---|---|

- Elimine cualquier comentario superfluo del código antes de implementar el proyecto.

  Si observa que hay demasiados comentarios en el código ActionScript, piense si necesita volver a escribir parte de él. La necesidad de incluir muchos comentarios sobre el funcionamiento del código ActionScript denota una falta de elegancia en la redacción del código.

  | NOTA | El uso de comentarios es especialmente importante si se va a presentar el código ActionScript ante un público. Por ejemplo, añada comentarios al código si está creando aplicaciones de ejemplo para el aprendizaje de Flash o si está escribiendo tutoriales sobre código ActionScript. |
  |---|---|

## Adición de comentarios a clases

Los dos tipos de comentarios en un archivo de clase o interfaz típico son *comentarios de documentación* y *comentarios de implementación*.

| NOTA | Los comentarios de documentación e implementación no se representan formalmente en el lenguaje ActionScript. Sin embargo, los desarrolladores los utilizan habitualmente cuando escriben archivos de clase e interfaz. |
|---|---|

Los comentarios de documentación se utilizan para describir las especificaciones del código, no la implementación. Los comentarios de implementación se utilizan para convertir en comentario parte del código o para comentar la implementación de determinadas secciones del código. Los comentarios de documentación se delimitan con /** y */, y los comentarios de implementación se delimitan con /* y */.

Los comentarios de documentación se utilizan para describir interfaces, clases, métodos y constructores. Incluya un solo comentario de documentación por cada clase, interfaz o miembro y colóquelo inmediatamente antes de la declaración. Si debe proporcionar más información de la que cabe en los comentarios de documentación, utilice los comentarios de implementación (con el formato de comentarios en bloque o comentarios de una sola línea).

Inicie las clases con un comentario estándar que siga este formato:

```
/**
  User class
  version 1.2
  3/21/2004
  copyright Macromedia, Inc.
 */
```

Tras los comentarios de documentación, declare la clase. Los comentarios de implementación deben ir inmediatamente después de la declaración.

| NOTA | No incluya comentarios que no estén directamente relacionados con la clase que se está leyendo. Por ejemplo, no incluya comentarios que describan el paquete correspondiente. |
|---|---|

Utilice comentarios de bloque, de una sola línea o finales en el cuerpo de la clase para comentar el código ActionScript. Para más información sobre el uso de comentarios en clases de clases, consulte "Adición de comentarios a clases" en la página 789.

# Convenciones de codificación de ActionScript

Uno de los aspectos más importantes de la programación es la coherencia, ya sea en los esquemas de asignación de nombres de variables (descritos en "Convenciones de asignación de nombre" en la página 777), en la aplicación de formato (descrita en "Aplicación de formato a la sintaxis de ActionScript" en la página 808) o en los estándares de codificación y la colocación del código ActionScript 2.0, que se describe en esta sección. La depuración y el mantenimiento del código se simplifican enormemente si el código se organiza según unos estándares.

Para más información sobre convenciones de codificación, consulte los siguientes temas:

- "Mantenimiento del código ActionScript en un único lugar" en la página 791
- "Asociación de código a los objetos" en la página 791
- "Gestión del ámbito" en la página 793
- "Estructuración de un archivo de clase" en la página 796
- "Utilización de funciones" en la página 804

## Mantenimiento del código ActionScript en un único lugar

Siempre que sea posible, coloque el código ActionScript 2.0 en un solo lugar, por ejemplo, en uno o varios archivos ActionScript externos o en el fotograma 1 de la línea de tiempo (cuando se coloca en la línea de tiempo, el código se denomina *script de fotograma*).

Si coloca código ActionScript en un script de fotograma, sitúelo en el primer o segundo fotograma de la línea de tiempo, en una capa denominada *Acciones*, que es la primera o segunda capa de la línea de tiempo. En ocasiones, puede crear dos capas de ActionScript para separar las funciones, lo cual es aceptable. Algunas aplicaciones Flash no siempre colocan todo el código en un solo lugar (en concreto, si se utilizan pantallas o comportamientos).

A excepción de estos casos poco habituales, normalmente puede colocar todo el código en una misma ubicación. A continuación se enumeran las ventajas que ofrece colocar el código ActionScript en un solo lugar:

- Es fácil encontrar el código en un archivo de origen potencialmente complejo.
- Es fácil depurar el código.

Una de las tareas más difíciles de la depuración de un archivo FLA es encontrar todo el código. Después de haber encontrado todo el código, debe descubrir cómo interactúa con otras partes del código y con el archivo FLA. Si coloca todo el código en un solo fotograma, será mucho más sencillo depurarlo porque estará centralizado y estos problemas se producirán con menor frecuencia. Para más información sobre cómo asociar código a los objetos (y descentralizar el código), consulte "Asociación de código a los objetos" en la página 791. Para más información sobre comportamientos y código descentralizado, consulte el Capítulo 3, "Recomendaciones para el uso de comportamientos" en *Utilización de Flash*.

## Asociación de código a los objetos

Debe evitar asociar código ActionScript a objetos (como pueden ser instancias de botón o clip de película) de un archivo FLA, incluso en el caso de aplicaciones sencillas o prototipo. La asociación de código a un objeto implica seleccionar una instancia de clip de película, componente o botón, abrir el editor de ActionScript (el panel Acciones o la ventana Script) y añadir código ActionScript mediante las funciones de controlador `on()` u `onClipEvent()`.

Esta práctica no se aconseja por las siguientes razones:

- Es difícil localizar el código ActionScript asociado a los objetos y editar los archivos FLA.
- Es difícil depurar el código ActionScript asociado a los objetos.
- El código ActionScript escrito en la línea de tiempo o en las clases es más elegante, y es más sencillo programar a partir de él.

- El código ActionScript asociado a los objetos propicia el uso de un estilo de codificación poco elegante.
- El código ActionScript asociado a objetos fuerza a los alumnos y lectores a aprender sintaxis adicional, además de estilos de codificación diferentes que suelen ser deficientes y limitados.
- Los usuarios generalmente deben volver a aprender a escribir funciones y demás en una línea de tiempo.

Puede que algunos usuarios de Flash consideren que es más sencillo aprender a utilizar código ActionScript asociando código a un objeto. Otros puede que opinen que es más fácil añadir código sencillo o escribir o enseñar código ActionScript de esta forma. Sin embargo, el contraste entre dos estilos de codificación (el código colocado en objetos y los scripts de fotogramas) puede resultar confuso para los desarrolladores que están aprendiendo ActionScript y se deben evitar. Asimismo, los usuarios que aprenden a escribir código asociado a objetos con frecuencia deben volver a aprender a colocar el código equivalente como un script de fotograma. Por este motivo resulta recomendable aprender a escribir scripts de fotograma y mantener una coherencia en el proceso de aprendizaje.

La asociación de código ActionScript a un botón denominado `myBtn` presenta el siguiente aspecto. Evite este método:

```
on (release) {
  // Realizar una acción.
}
```

Sin embargo, colocar el código ActionScript equivalente en una línea de tiempo presenta este aspecto:

```
// código correcto
myBtn.onRelease = function() {
  // Realizar una acción.
};
```

Para más información sobre la sintaxis de ActionScript, consulte "Aplicación de formato a la sintaxis de ActionScript" en la página 808.

**NOTA**

La utilización de comportamientos y pantallas implica a veces la asociación de código a objetos, por lo que deben aplicarse otras prácticas al utilizar estas funciones. Para más información, consulte el Capítulo 3, "Recomendaciones para el uso de comportamientos" en *Utilización de Flash*.

## Gestión del ámbito

El ámbito es el área donde la variable es conocida y puede utilizarse en un archivo SWF, por ejemplo, en una línea de tiempo, globalmente en toda la aplicación o de forma local en una función. Por lo general, puede hacer referencia al ámbito de varias formas cuando escribe código. El uso correcto del ámbito implica que puede crear código ActionScript portátil y reutilizable sin riesgo de romper las aplicaciones al crear nuevos módulos.

Es importante entender la diferencia entre los ámbitos global y raíz. El ámbito raíz es exclusivo de cada archivo SWF cargado. El ámbito global se aplica a todas las líneas de tiempo y ámbitos de los archivos SWF. Utilice direccionamiento relativo en lugar de referencias a las líneas de tiempo raíz, ya que el direccionamiento relativo hace que el código sea reutilizable y portátil. Para más información sobre la gestión del ámbito en las aplicaciones, consulte las secciones siguientes:

"Variables y ámbito" en la página 354

"Ámbito y referencias" en la página 86

"Clases y ámbito" en la página 256.

### Cómo evitar destinos absolutos (_root)

Puede utilizar diversos métodos para hacer referencia a las instancias que permiten evitar el uso de `_root`; se describen más adelante en esta sección. Evite el uso de `_root` en ActionScript 2.0 porque los archivos SWF que se cargan en otros archivos SWF podrían no funcionar correctamente. El identificador `_root` utiliza el archivo SWF base que se está cargando y no el archivo SWF donde se utiliza el direccionamiento relativo en lugar de `_root`. Este problema limita la portabilidad del código en los archivos SWF que se cargan en otro archivo y, en concreto, en componentes y clips de película. Puede ayudar a resolver estos problemas mediante el uso de `_lockroot`, pero utilice `_lockroot` únicamente cuando sea necesario (por ejemplo, cuando cargue un archivo SWF pero no tenga acceso al archivo FLA). Para más información sobre el uso de `_lockroot`, consulte "Utilización de _lockroot" en la página 794.

Utilice las palabras clave `this`, `this._parent` o `_parent` en lugar de `_root` en función de la ubicación del código ActionScript 2.0. A continuación se muestra un ejemplo de direccionamiento relativo:

```
myClip.onRelease = function() {
  trace(this._parent.myButton._x);
};
```

Todas las variables deben incluirse en un ámbito, excepto las variables que son parámetros de función y variables locales. Siempre que sea posible, incluya en un ámbito las variables relativas a su ruta actual, mediante el direccionamiento relativo como, por ejemplo, la propiedad `this`. Para más información sobre la utilización de la propiedad `this`, consulte %{this property}% en *Referencia del lenguaje ActionScript 2.0*.

## Utilización de _lockroot

Puede utilizar `_lockroot` para hacer referencia a un contenido con el fin de resolver los problemas de ámbito que a veces conlleva el uso inadecuado de `_root`. Aunque de esta forma se solucionan muchos problemas con las aplicaciones, considere la posibilidad de utilizar `_lockroot` para solucionar los problemas debidos al uso de `_root`. Si tiene problemas al cargar contenido en un archivo SWF o una instancia de componente, intente aplicar `_lockroot` a un clip de película que cargue el contenido. Por ejemplo, si tiene un clip de película denominado `myClip` que carga contenido y deja de funcionar después de cargarse, intente utilizar el siguiente código, que se sitúa en una línea de tiempo:

```
this._lockroot = true;
```

## Utilización de la palabra clave this

Siempre que sea posible, utilice la palabra clave `this` como prefijo en lugar de omitirla, aunque el código funcione sin ella. Utilice la palabra clave `this` para saber si un método o propiedad pertenece a una determinada clase. Por ejemplo, para una función en una línea de tiempo, escriba el código ActionScript 2.0 con el siguiente formato:

```
circleClip.onPress = function() {
  this.startDrag();
};
circleClip.onRelease = function() {
  this.stopDrag();
};
```

Para una clase, utilice el siguiente formato para escribir código:

```
class User {
  private var username:String;
  private var password:String;
  function User(username:String, password:String) {
    this.username = username;
    this.password = password;
  }
  public function get username():String {
    return this.username;
  }
  public function set username(username:String):Void {
    this.username = username;
  }
}
```

Si en estas situaciones añade de forma coherente la palabra clave `this`, será mucho más sencillo leer y entender el código ActionScript 2.0.

## El ámbito en las clases

Si transfiere código a clases de ActionScript 2.0, es posible que deba cambiar la manera en que utiliza la palabra clave `this`. Por ejemplo, si tiene un método de clase que utiliza una función callback (como el método `onLoad` de la clase LoadVars), puede ser difícil saber si la palabra clave `this` hace referencia a la clase o al objeto LoadVars. En esta situación, puede que deba crear un puntero a la clase actual, como se muestra en el siguiente ejemplo:

```
class Product {
  private var m_products_xml:XML;
  // Constructor
  // targetXmlStr contiene la ruta a un archivo XML
  function Product(targetXmlStr:String) {
    /* Crear una referencia local a la clase actual.
      Aunque esté dentro del controlador de eventos onLoad de XML,
      puede hacer referencia a la clase actual y no sólo al paquete XML. */
    var thisObj:Product = this;
    // Crear una variable local, que se utiliza para cargar el archivo XML.
    var prodXml:XML = new XML();
    prodXml.ignoreWhite = true;
    prodXml.onLoad = function(success:Boolean) {
      if (success) {
        /* Si el XML se carga y analiza correctamente,
          establezca la variable m_products_xml de la clase en el
          documento XML analizado y llame a la función init. */
        thisObj.m_products_xml = this;
        thisObj.init();
      } else {
        /* Error al cargar el archivo XML. */
        trace("error loading XML");
      }
    };
    // Iniciar carga del documento XML
    prodXml.load(targetXmlStr);
  }
  public function init():Void {
    // Mostrar el paquete XML
    trace(this.m_products_xml);
  }
}
```

Dado que se intenta hacer referencia a la variable de miembro privada de un controlador `onLoad`, la palabra clave `this` realmente hace referencia a la instancia `prodXml` y no a la clase `Product`, que es lo que podría esperarse. Por este motivo, debe crear un puntero al archivo de clase local para que pueda hacer referencia a la clase directamente desde el controlador `onLoad`.

Para más información sobre clases, consulte el "Clases y ámbito" en la página 256. Para más información sobre ámbitos, consulte "Gestión del ámbito" en la página 793.

# Estructuración de un archivo de clase

Puede crear clases en archivos de ActionScript 2.0 independientes que se importan en un archivo SWF cuando se compila.

Puede crear clases en archivos de ActionScript 2.0 independientes que se importen a un archivo SWF cuando se compile una aplicación. Para crear un archivo de clase, deberá escribir código que incluya una determinada metodología y ordenación. Esta metodología se describe en las siguientes secciones.

Las siguientes convenciones de estructuración de un archivo de clase muestran cómo ordenar partes de una clase para aumentar la eficacia y mejorar la legibilidad del código.

### Para estructurar un archivo de clase, utilice los siguientes elementos:

1. Añada comentarios de documentación que incluyan una descripción general del código, además de información del autor y de la versión.
2. Añada las sentencias de importación (si resulta aplicable).
3. Escriba una declaración de clase o declaración de interfaz como la siguiente:

   ```
   UserClass{...}
   ```
4. Incluya los comentarios de implementación de clase o interfaz que sean necesarios.

   En este comentario, añada información que sea relevante para toda la clase o interfaz.
5. Añada todas las variables estáticas.

   Escriba primero las variables de clases públicas y, a continuación, las variables de clases privadas.
6. Añada variables de instancia.

   Escriba primero las variables de miembros públicas y, a continuación, las variables de miembros privadas.
7. Añada la sentencia constructora, como la del siguiente ejemplo:

   ```
   public function UserClass(username:String, password:String) {...}
   ```
8. Escriba los métodos.

   Agrupe los métodos por la funcionalidad y no por la accesibilidad o el ámbito. Esta forma de organizar los métodos ayuda a mejorar la legibilidad y claridad del código.
9. Escriba los métodos getter/setter (captador/definidor) en el archivo de clase.

## Directrices para la creación de una clase

Tenga presentes las siguientes directrices cuando cree un archivo de clase:

- Coloque una sola declaración en cada línea.
- No coloque varias declaraciones en una sola línea.

  Por ejemplo, aplique formato a las declaraciones como se muestra en el siguiente ejemplo:

  ```
  var prodSkuNum:Number;    // Número de identificación de SKU del producto
  var prodQuantityNum:Number; // Cantidad de producto
  ```

  Este ejemplo muestra una forma mejor que incluir ambas declaraciones en una sola línea. Coloque estas declaraciones al principio de un bloque de código.
- Inicialice las variables locales al declararlas.

  Las propiedades de una clase sólo se deben inicializar en la declaración si el inicializador es una constante en tiempo de compilación.
- Declare las variables antes de utilizarlas por primera vez.

  Aquí se incluyen también los bucles.
- Evite utilizar declaraciones locales que oculten declaraciones de nivel superior.

  Por ejemplo, no declare una variable dos veces, como se muestra en el siguiente ejemplo:

  ```
  var counterNum:Number = 0;
  function myMethod() {
    for (var counterNum:Number = 0; counterNum<=4; counterNum++) {
      // sentencias;
    }
  }
  ```

  Este código declara la misma variable dentro de un bloque interno, algo que debe evitarse.
- No asigne muchas variables a un único valor de una sentencia.

  Siga esta convención, ya que, de lo contrario, el código resulta difícil de leer, como muestra el siguiente código ActionScript:

  ```
  playBtn.onRelease = playBtn.onRollOut = playsound;
  ```

  o

  ```
  class User {
    private var m_username:String, m_password:String;
  }
  ```
- Haga el método o la propiedad públicos sólo si existe un motivo para ello. De lo contrario, haga los métodos y propiedades privados.

- No haga un *uso excesivo* de las funciones de captores/definidores en el archivo de clase.

  Las funciones de captores/definidores resultan excelentes para diversos fines (consulte "Métodos getter (captador) y setter (definidor)" en la página 226), sin embargo es posible que un uso excesivo indique que podría mejorar la arquitectura o la organización de la aplicación.
- Defina como privadas la mayoría de las variables de miembros a menos que tenga un buen motivo para hacerlas públicas.

  Desde el punto de vista del diseño, es mucho mejor definir variables de miembros privadas y permitir el acceso a las variables únicamente a través de un grupo de funciones de captores/definidores.

## Utilización del prefijo this en archivos de clases

Utilice la palabra clave `this` como prefijo dentro de las clases para métodos y variables de miembros. Aunque no es necesario, es fácil saber que una propiedad o método pertenece a una clase cuando tiene un prefijo; sin él, no es posible determinar si la propiedad o el método pertenece a una superclase.

También es posible utilizar un prefijo de nombre de clase en variables estáticas y métodos, e incluso en una clase. Esto contribuye a calificar las referencias que establezca. La calificación de las referencias mejora la legibilidad del código. En función del entorno de codificación que se utilice, la utilización de prefijos puede activar además la capacidad de completar código y las sugerencias. El código siguiente muestra la utilización de un prefijo en una propiedad estática con un nombre de clase:

```
class Widget {
  public static var widgetCount:Number = 0;
  public function Widget() {
    Widget.widgetCount++;
  }
}
```

**NOTA** En opinión de algunos desarrolladores, no es necesario añadir estos prefijos. Macromedia recomienda añadir la palabra clave `this` como prefijo, ya que puede facilitar la legibilidad y ayudarle a escribir un código limpio al proporcionar contexto.

## Inicialización

En los valores iniciales de las variables, asigne un valor predeterminado o permita el valor `undefined`, como se muestra en el siguiente ejemplo de clase. Al inicializar propiedades en línea, la expresión situada en el lado derecho de una asignación debe ser una constante en tiempo de compilación. Es decir, la expresión no puede hacer referencia a nada que se haya establecido o definido durante la ejecución. Las constantes en tiempo de compilación incluyen literales de cadena, números, valores booleanos, null y undefined, así como funciones constructoras para las siguientes clases del más alto nivel: Array, Boolean, Number, Object y String. Esta clase establece los valores iniciales de `m_username` y `m_password` en cadenas vacías:

```
class User {
  private var m_username:String = "";
  private var m_password:String = "";
  function User(username:String, password:String) {
    this.m_username = username;
    this.m_password = password;
  }
}
```

Elimine variables o convierta las variables en `null` cuando ya no las necesite. Si se establecen las variables en `null` puede mejorarse el rendimiento. Este proceso suele denominarse en inglés *garbage collection* (eliminación de datos innecesarios). Eliminar las variables ayuda a optimizar el uso de la memoria en tiempo de ejecución, porque los activos no necesarios se eliminan del archivo SWF. Es mejor eliminar variables que establecerlas en `null`. Para más información sobre el rendimiento, consulte "Optimización del código" en la página 807.

| NOTA | Flash Player 8 ha introducido mejoras en la eliminación de datos innecesarios en Flash Player. |
|---|---|

Para más información sobre la asignación de nombres para las variables, consulte "Asignación de nombre a variables" en la página 780. Para más información sobre la eliminación de objetos, consulte %{delete statement}% en *Referencia del lenguaje ActionScript 2.0*.

Una de las formas más sencillas de inicializar código con ActionScript 2.0 es mediante el uso de clases. Puede encapsular todas las instrucciones de inicialización de una instancia en la función constructora de la clase o extraerlas en un método independiente, al que llamaría de forma explícita después de crear la variable, tal y como muestra el siguiente código:

```
class Product {
  function Product() {
    var prodXml:XML = new XML();
    prodXml.ignoreWhite = true;
    prodXml.onLoad = function(success:Boolean) {
      if (success) {
        trace("loaded");
      } else {
        trace("error loading XML");
      }
    };
    prodXml.load("products.xml");
  }
}
```

El código siguiente podría ser la primera llamada de función en la aplicación y la única que realizaría para llevar a cabo la inicialización. El fotograma 1 de un archivo FLA que carga XML puede utilizar código similar al siguiente código ActionScript:

```
if (init == undefined) {
  var prodXml:XML = new XML();
  prodXml.ignoreWhite = true;
  prodXml.onLoad = function(success:Boolean) {
    if (success) {
      trace("loaded");
    } else {
      trace("error loading XML");
    }
  };
  prodXml.load("products.xml");
  init = true;
}
```

## Utilización de sentencias trace

Utilice sentencias `trace` en los documentos para facilitar la depuración del código durante la edición del archivo FLA. Por ejemplo, si utiliza una sentencia `trace` y un bucle `for`, puede ver los valores de las variables en el panel Salida, por ejemplo, cadenas, matrices y objetos, como se muestra en el ejemplo siguiente:

```
var dayArr:Array = ["sun", "mon", "tue", "wed", "thu", "fri", "sat"];
var numOfDays:Number = dayArr.length;
for (var i = 0; i<numOfDays; i++) {
  trace(i+": "+dayArr[i]);
}
```

Este código muestra la siguiente información en el panel Salida:

```
0: sun
1: mon
2: tue
3: wed
4: thu
5: fri
6: sat
```

El uso de una sentencia `trace` es una forma eficaz de depurar el código ActionScript 2.0.

Puede eliminar las sentencias `trace` al publicar un archivo SWF para mejorar levemente el rendimiento de la reproducción. Antes de publicar un archivo SWF, abra Configuración de publicación y seleccione Omitir acciones de trazado en la ficha Flash. Para más información sobre la utilización de una función trace, consulte %{trace function}% en *Referencia del lenguaje ActionScript 2.0*.

La herramienta de depurador también resulta útil para depurar código ActionScript. Para más información, consulte el Capítulo 18, “Depuración de aplicaciones”.

## Utilización del prefijo super

Si hace referencia a un método en la clase principal, añada el prefijo `super` al método para que otros desarrolladores sepan desde dónde se invoca el método. El siguiente fragmento de código ActionScript 2.0 muestra el uso del ámbito correcto mediante el prefijo `super`:

En el ejemplo siguiente se crean dos clases. Se utiliza la palabra clave `super` en la clase Socks para llamar a funciones en la clase principal (Clothes). Aunque las clases Socks y Clothes contienen un método denominado `getColor()`, el uso del prefijo `super` permite hacer referencia específicamente a los métodos y propiedades de la clase base. Cree un archivo AS nuevo llamado Clothes.as e introduzca el código siguiente:

```
class Clothes {
   private var color:String;
   function Clothes(paramColor) {
      this.color = paramColor;
      trace("[Clothes] I am the constructor");
   }
   function getColor():String {
      trace("[Clothes] I am getColor");
      return this.color;
   }
   function setColor(paramColor:String):Void {
      this.color = paramColor;
      trace("[Clothes] I am setColor");
   }
}
```

Cree una clase nueva denominada Socks que amplía la clase Clothes, tal como se muestra en el siguiente ejemplo:

```
class Socks extends Clothes {
   private var color:String;
   function Socks(paramColor:String) {
      this.color = paramColor;
      trace("[Socks] I am the constructor");
   }
   function getColor():String {
      trace("[Socks] I am getColor");
      return super.getColor();
   }
   function setColor(paramColor:String):Void {
      this.color = paramColor;
      trace("[Socks] I am setColor");
   }
}
```

A continuación, cree un nuevo archivo AS o FLA e introduzca el siguiente código ActionScript en el documento:

```
import Socks;
var mySock:Socks = new Socks("maroon");
trace(" -> "+mySock.getColor());
mySock.setColor("Orange");
trace(" -> "+mySock.getColor());
```

El resultado siguiente se muestra en el panel Salida:

```
[Clothes] I am the constructor
[Socks] I am the constructor
[Socks] I am getColor
[Clothes] I am getColor
-> maroon
[Socks] I am setColor
[Socks] I am getColor
[Clothes] I am getColor
-> Orange
```

Si olvidó poner la palabra clave `super` en el método `getColor()` de la clase Socks, el método `getColor()` podría llamarse a sí mismo repetidamente, lo que provocaría que el script fallase a causa de los problemas de recursión infinita. El panel Salida mostraría el siguiente error si no ha utilizado la palabra clave super:

```
[Socks] I am getColor
[Socks] I am getColor
...
[Socks] I am getColor
256 levels of recursion were exceeded in one action list.
This is probably an infinite loop.
Further execution of actions has been disabled in this SWF file.
```

## Cómo evitar la utilización de la sentencia with

Uno de los conceptos más difíciles de entender para quienes están aprendiendo a utilizar código ActionScript 2.0 es el uso de la sentencia `with`. Tome como ejemplo el siguiente código que utiliza la sentencia `with`:

```
this.attachMovie("circleClip", "circle1Clip", 1);
with (circle1Clip) {
  _x = 20;
  _y = Math.round(Math.random()*20);
  _alpha = 15;
  createTextField("labelTxt", 100, 0, 20, 100, 22);
  labelTxt.text = "Circle 1";
  someVariable = true;
}
```

En este código, se añade una instancia de clip de película de la biblioteca y se utiliza la sentencia `with` para modificar sus propiedades. Si no se especifica el ámbito de una variable, no siempre se sabe dónde se establecen las propiedades, por lo que el código puede resultar confuso. En el código anterior, podría esperar que se estableciera `someVariable` en el clip de película `circle1Clip`, pero se establece realmente en una línea de tiempo del archivo SWF.

Es más sencillo comprender el código si se especifica de forma explícita el ámbito de las variables, en lugar de utilizar la sentencia `with`. En el ejemplo siguiente se muestra un código ActionScript un poco más largo, pero mejor, donde se especifica el ámbito de las variables:

```
this.attachMovie("circleClip", "circle1Clip", 1);
circle1Clip._x = 20;
circle1Clip._y = Math.round(Math.random()*20);
circle1Clip._alpha = 15;
circle1Clip.createTextField("labelTxt", 100, 0, 20, 100, 22);
circle1Clip.labelTxt.text = "Circle 1";
circle1Clip.someVariable = true;
```

Una excepción a esta regla: cuando trabaja con la interfaz API de dibujo para dibujar formas, podría obtener varias llamadas similares a los mismos métodos (por ejemplo, `lineTo` o `curveTo`) como consecuencia de la funcionalidad de la interfaz API de dibujo. Por ejemplo, si dibuja un rectángulo sencillo, es necesario realizar cuatro llamadas independientes al método `lineTo`, como se muestra en el código siguiente:

```
this.createEmptyMovieClip("rectangleClip", 1);
with (rectangleClip) {
  lineStyle(2, 0x000000, 100);
  beginFill(0xFF0000, 100);
  moveTo(0, 0);
  lineTo(300, 0);
  lineTo(300, 200);
  lineTo(0, 200);
  lineTo(0, 0);
  endFill();
}
```

Si se ha utilizado un nombre de instancia completo en cada método `lineTo` o `curveTo`, el código se saturará rápidamente y será difícil leerlo y depurarlo.

## Utilización de funciones

Siempre que sea posible, reutilice los bloques de código. Una forma de reutilizar código es llamar a una función varias veces, en lugar de crear código distinto cada vez. Las funciones pueden ser fragmentos genéricos de código, por consiguiente, puede utilizar los mismos bloques de código para fines ligeramente distintos en un archivo SWF. La reutilización del código permite crear aplicaciones eficaces y reduce al mínimo la cantidad de código ActionScript 2.0 que debe escribirse, por lo que el tiempo de desarrollo es menor. Puede crear funciones en una línea de tiempo, en un archivo de clase o escribir código ActionScript que resida en un componente basado en código y reutilizarlos de diferentes formas.

Si utiliza ActionScript 2.0, evite escribir funciones en una línea de tiempo. Al utilizar código ActionScript 2.0, coloque las funciones en los archivos de clase siempre que sea posible, tal y como se muestra en el siguiente ejemplo:

```
class Circle {
public function area(radius:Number):Number {
  return (Math.PI*Math.pow(radius, 2));
}
public function perimeter(radius:Number):Number {
  return (2 * Math.PI * radius);
}
public function diameter(radius:Number):Number {
  return (radius * 2);
}
}
```

Utilice la siguiente sintaxis para crear funciones:

```
function myCircle(radius:Number):Number {
  //...
}
```

Evite utilizar la siguiente sintaxis, pues es difícil de leer:

```
myCircle = function(radius:Number):Number {
  //...
}
```

En el ejemplo siguiente se incluyen funciones en un archivo de clase. Es una de las prácticas recomendadas si se opta por utilizar ActionScript 2.0 porque maximiza la reutilización del código. Para reutilizar las funciones en otras aplicaciones, importe la clase existente en lugar de volver a escribir el código desde cero o duplicar las funciones en la nueva aplicación.

```
class mx.site.Utils {
  static function randomRange(min:Number, max:Number):Number {
    if (min>max) {
      var temp:Number = min;
      min = max;
      max = temp;
    }
    return (Math.floor(Math.random()*(max-min+1))+min);
  }
  static function arrayMin(numArr:Array):Number {
    if (numArr.length == 0) {
      return Number.NaN;
    }
    numArr.sort(Array.NUMERIC | Array.DESCENDING);
    var min:Number = Number(numArr.pop());
    return min;
  }
  static function arrayMax(numArr:Array):Number {
    if (numArr.length == 0) {
      return undefined;
    }
    numArr.sort(Array.NUMERIC);
    var max:Number = Number(numArr.pop());
    return max;
  }
}
```

Para utilizar estas funciones, puede añadir el siguiente código ActionScript al archivo FLA:

```
import mx.site.Utils;
var randomMonth:Number = Utils.randomRange(0, 11);
var min:Number = Utils.arrayMin([3, 3, 5, 34, 2, 1, 1, -3]);
var max:Number = Utils.arrayMax([3, 3, 5, 34, 2, 1, 1, -3]);
trace("month: "+randomMonth);
trace("min: "+min);
trace("max: "+max);
```

### Cómo detener la repetición de código

El controlador de eventos `onEnterFrame` es útil porque Flash puede utilizarlo para repetir código a la velocidad de fotogramas de un archivo SWF. Sin embargo, limite tanto como sea posible la cantidad de repetición utilizada en un archivo Flash para que el rendimiento no se vea afectado. Se consumen muchos recursos del procesador cuando, por ejemplo, se tiene un segmento de código que se repite cada vez que la cabeza lectora entra en un fotograma. Este comportamiento puede provocar problemas de rendimiento en equipos que reproducen el archivo SWF. Si utiliza el controlador de eventos `onEnterFrame` para cualquier tipo de animación o repetición en los archivos SWF, elimine el controlador `onEnterFrame` cuando termine de utilizarlo. En el siguiente código ActionScript 2.0, la repetición se detiene al eliminar el controlador de eventos `onEnterFrame`:

```
circleClip.onEnterFrame = function() {
  circleClip._alpha -= 5;
  if (circleClip._alpha<=0) {
    circleClip.unloadMovie();
    delete this.onEnterFrame;
    trace("deleted onEnterFrame");
  }
};
```

De forma similar, limite el uso de `setInterval` y recuerde borrar el intervalo cuando termine de utilizarlo para reducir los requisitos del procesador para el archivo SWF.

# Optimización de ActionScript y Flash Player

Si compila un archivo SWF que contiene ActionScript 2.0 con una configuración de publicación establecida en Flash Player 6 y ActionScript 1.0, el código funcionará mientras no se utilicen clases de ActionScript 2.0. El código no distingue entre mayúsculas y minúsculas, sólo se utiliza código Flash Player. Por lo tanto, si se compila el archivo SWF con una Configuración de publicación establecida en Flash Player 7 u 8 y ActionScript 1.0, Flash aplica la distinción entre mayúsculas y minúsculas.

Las anotaciones de tipo de datos (strict data types) se aplican durante la compilación para Flash Player 7 y 8 cuando se ha establecido la configuración de publicación en ActionScript 2.0.

ActionScript 2.0 compila al código de bytes de ActionScript 1.0 cuando se publican las aplicaciones, por lo que se puede hacer referencia a Flash Player 6, 7 u 8 cuando trabaje con ActionScript 2.0.

Para más información sobre la optimización de aplicaciones, consulte “Optimización del código”.

## Optimización del código

Tenga presentes las siguientes directrices al optimizar el código:

- Evite llamar a una función varias veces desde un bucle.

  Es mejor incluir en el bucle el contenido de una función pequeña.
- Utilice funciones nativas siempre que sea posible.

  Las funciones nativas son más rápidas que las funciones definidas por el usuario.
- No haga un uso excesivo del tipo Object.

  Las anotaciones de tipos de datos deben ser precisas, ya que de este modo se mejora el rendimiento. Sólo use el tipo Object si no hay ninguna alternativa razonable.
- Evite utilizar la función `eval()` o el operador de acceso a una matriz.

  A menudo es preferible y más eficaz establecer la referencia local una sola vez.
- Asigne `Array.length` a una variable antes de un bucle.

  Asigne `Array.length` a una variable antes de un bucle para utilizarla como condición, en lugar de usar la propia `myArr.length`. Por ejemplo,

  ```
  var fontArr:Array = TextField.getFontList();
  var arrayLen:Number = fontArr.length;
  for (var i:Number = 0; i < arrayLen; i++) {
    trace(fontArr[i]);
  }
  ```

  en lugar de:

  ```
  var fontArr:Array = TextField.getFontList();
  for (var i:Number = 0; i < fontArr.length; i++) {
    trace(fontArr[i]);
  }
  ```
- Céntrese en optimizar los bucles y en cualquier acción que se repita.

  Flash Player consume una gran cantidad de tiempo en procesar bucles (como los que utilizan la función `setInterval()`).
- Añada la palabra clave `var` al declarar una variable.
- No utilice variables de clase ni variables globales si basta con utilizar variables locales.

# Aplicación de formato a la sintaxis de ActionScript

La aplicación de formato estandarizado al código ActionScript 2.0 resulta esencial para lograr un código que pueda mantenerse, al tiempo que facilita la comprensión y modificación del código a otros desarrolladores. Por ejemplo, sería muy difícil seguir la lógica de un archivo FLA que no tuviera sangrado ni comentarios y que tuviera un formato y unas convenciones de asignación de nombres incoherentes. El sangrado de bloques de código (por ejemplo, bucles y sentencias `if`) facilita la legibilidad y la depuración del código.

Para más información sobre la aplicación de formato al código, consulte los siguientes temas:



## Directrices generales para la asignación de formatos

Cuando utiliza espacios y saltos de línea y aplica sangrías para añadir espacios en blanco al código, se incrementa la legibilidad del mismo. Los espacios en blanco incrementan la legibilidad ya que ayudan a mostrar la jerarquía del código. Mejorar la legibilidad del código ActionScript 2.0 para hacer que resulte más fácil de entender es importante para los estudiantes y para los usuarios experimentados que trabajan en proyectos complejos. La legibilidad también es importante para depurar el código ActionScript, porque es mucho más sencillo localizar los errores si el código tiene el formato correcto y los espacios adecuados.

Hay varias formas de escribir un fragmento de código ActionScript 2.0 o aplicarle formato. Observará diferencias en la manera en que los desarrolladores forman la sintaxis en varias líneas en el editor de ActionScript (el panel Acciones o la ventana Script), como, por ejemplo, el lugar en que se sitúan las llaves (`{}`) o los paréntesis [`()`]).

Macromedia recomienda las siguientes prácticas para lograr la máxima legibilidad posible en el código ActionScript.

- Inserte una línea en blanco entre los párrafos (módulos) de código ActionScript.

  Los párrafos de código ActionScript son grupos de código relacionado lógicamente. La adición de una línea en blanco entre ellos permite a los usuarios leer el código ActionScript con mayor facilidad y comprender su lógica.

- Utilice un sangrado coherente en el código para ayudar a mostrar la jerarquía de la estructura del código.

  Utilice el mismo estilo de sangrado en todo el código ActionScript y asegúrese de que alinea correctamente las llaves ({}). La alineación de las llaves mejora la legibilidad del código. Si la sintaxis de ActionScript es correcta, Flash aplica automáticamente la sangría correcta al código cuando se presiona Intro (Windows) o Retorno (Macintosh). También puede hacer clic en el botón Formato automático del editor de ActionScript (el panel Acciones o la ventana Script) para aplicar sangría al código ActionScript si la sintaxis es correcta.

- Utilice saltos de línea para facilitar la lectura de sentencias complejas.

  Puede aplicar formato a algunas sentencias, como las sentencias condicionales, de diversas maneras. A veces, la aplicación de formato a sentencias en varias líneas en lugar de una sola línea mejora la legibilidad del código.

- Incluya un espacio después de una palabra clave que vaya seguida de paréntesis [()].

  Esto se muestra en el siguiente ejemplo de código ActionScript:

```
do {
  //alguna acción
} while (condition);
```

- No inserte un espacio entre el nombre de un método y los paréntesis.

  Esto se muestra en el siguiente ejemplo de código ActionScript:

```
function checkLogin():Boolean {
  // sentencias;
}
checkLogin();
```

  o

```
printSize("size is " + foo + "\n");
```

- Inserte un espacio después de las comas en una lista de argumentos.

  El uso de espacios detrás de las comas facilita la distinción entre llamadas a métodos y palabras clave, tal como se muestra en el siguiente ejemplo:

  ```
  function addItems(item1:Number, item2:Number):Number {
    return (item1 + item2);
  }
  var sum:Number = addItems(1, 3);
  ```

- Utilice espacios para separar todos los operadores y sus operandos.

  El uso de espacios facilita la distinción entre llamadas a métodos y palabras clave, tal como se muestra en el siguiente ejemplo:

  ```
  //bien
  var sum:Number = 7 + 3;
  //mal
  var sum:Number=7+3;
  ```

  Una excepción a esta directriz es el operador de punto (`.`).

- No inserte un espacio entre operadores unarios y sus operandos.

  Por ejemplo, incremento (++) y decremento (--), como se muestra en el siguiente ejemplo:

  ```
  while (d++ = s++)
  -2, -1, 0
  ```

- No inserte espacios después de un paréntesis inicial y antes de un paréntesis final.

  Esto se muestra en el siguiente ejemplo de código ActionScript:

  ```
  //mal
  ( "size is " + foo + "\n" );
  //bien
  ("size is " + foo + "\n");
  ```

- Coloque cada sentencia en una línea independiente para aumentar la legibilidad del código ActionScript.

  Esto se muestra en el siguiente ejemplo de código ActionScript:

  ```
  theNum++;       // Correcto
  theOtherNum++;  // Correcto
  aNum++; anOtherNum++;  // Incorrecto
  ```

- No incorpore asignaciones.

  Las sentencias incorporadas se utilizan en ocasiones para mejorar el rendimiento de un archivo SWF durante la ejecución, pero el código resulta más difícil de leer y depurar. Esto se muestra en el siguiente ejemplo de código ActionScript (pero recuerde que debe evitar los nombres de un solo carácter en el código real):

  ```
  var myNum:Number = (a = b + c) + d;
  ```

- Asigne las variables como sentencias independientes.

  Esto se muestra en el siguiente ejemplo de código ActionScript (pero recuerde que debe evitar los nombres de un solo carácter en el código real):

```
var a:Number = b + c;
var myNum:Number = a + d;
```

- Inserte un salto de línea antes de un operador.
- Rompa una línea después de una coma.
- Alinee la segunda línea con el principio de la expresión de la anterior línea de código.

| NOTA | La aplicación de sangrado y sangrado automático puede controlarse mediante la selección de Edición › Preferencias (Windows) o Flash › Preferencias (Macintosh) y, a continuación, la ficha ActionScript. |
|---|---|

## Escritura de sentencias condicionales

Siga estas directrices al escribir sentencias condicionales:

- Coloque las condiciones en líneas independientes en las sentencias `if`, `else..if` e `if..else`.
- Utilice llaves (`{}`) para las sentencias `if`.
- Aplique formato a las llaves como se muestra en los siguientes ejemplos:

```
//sentencia if
if (condition) {
  // sentencias
}

// sentencia if..else
if (condition) {
  // sentencias
} else {
  // sentencias
}

// sentencia else..if
if (condition) {
  // sentencias
} else if (condition) {
  // sentencias
} else {
  // sentencias
}
```

Si se escriben condiciones complejas, es aconsejable agrupar las condiciones con el uso de paréntesis [()]. Si no se utilizan paréntesis, el desarrollador u otras personas que trabajen con el código ActionScript 2.0 pueden encontrar errores de precedencia del operador.

Por ejemplo, en el código siguiente las condiciones no se escriben entre paréntesis:

```
if (fruit == apple && veggie == leek) {}
```

En el código siguiente se utiliza el formato correcto, pues las condiciones se escriben entre paréntesis:

```
if ((fruit == apple) && (veggie == leek)) {}
```

Puede escribir una sentencia condicional que devuelve un valor booleano de dos formas. El segundo ejemplo es preferible:

```
if (cartArr.length>0) {
  return true;
} else {
  return false;
}
```

Compare este ejemplo con el anterior:

```
// mejor
return (cartArr.length > 0);
```

El segundo fragmento es más corto y tiene menos expresiones para evaluar. Es más fácil leerlo y entenderlo.

En el siguiente ejemplo se comprueba si la variable *y* es mayor que cero (0) y devuelve el resultado de `x/y` o un valor cero (0).

```
return ((y > 0) ? x/y : 0);
```

En el siguiente ejemplo se muestra otra forma de escribir este código. Este ejemplo es preferible:

```
if (y>0) {
  return x/y;
} else {
  return 0;
}
```

La sintaxis abreviada de la sentencia `if` del primer ejemplo se conoce como operador condicional (`?:`). Permite convertir sentencias `if..else` sencillas en una sola línea de código. En este caso, la sintaxis abreviada reduce la legibilidad.

Si tiene que utilizar operadores condicionales, coloque entre paréntesis la condición inicial (antes del signo de interrogación [`?`]) para mejorar la legibilidad del código. En el fragmento de código anterior se incluye un ejemplo.

## Escritura de sentencias compuestas

Las sentencias compuestas contienen una lista de sentencias entre llaves ({}). Las sentencias entre llaves aparecen con sangría con respecto a la sentencia compuesta. Esto se muestra en el siguiente ejemplo de código ActionScript:

```
if (a == b) {
  // Este código aparece sangrado.
  trace("a == b");
}
```

Coloque llaves alrededor de las sentencias cuando formen parte de una estructura de control (`if..else` o `for`), aunque contengan una sola sentencia. En el siguiente ejemplo se muestra código escrito de forma incorrecta:

```
//mal
if (numUsers == 0)
  trace("no users found.");
```

Aunque este código es válido, se considera que está incorrectamente escrito porque las sentencias no aparecen entre llaves. En este caso, si añade otra sentencia después de la sentencia `trace`, el código se ejecutará independientemente de que la variable `numUsers` sea igual a 0.

```
//mal
var numUsers:Number = 5;
if (numUsers == 0)
  trace("no users found.");
  trace("I will execute");
```

La ejecución del código sin tener en cuenta la variable `numUsers` puede producir resultados inesperados. Por este motivo, añada llaves como se muestra en el siguiente ejemplo:

```
var numUsers:Number = 0;
if (numUsers == 0) {
  trace("no users found");
}
```

Al escribir una condición, no añada al código la parte `==true` redundante, como se muestra a continuación:

```
if (something == true) {
  // sentencias
}
```

Si compara con `false`, podría utilizar `if (something==false)` o `if(!something)`.

## Escritura de una sentencia for

Puede escribir la sentencia `for` utilizando el siguiente formato:

```
for (init; condition; update) {
  // sentencias
}
```

La siguiente estructura muestra la sentencia `for`:

```
var i:Number;
for (var i = 0; i<4; i++) {
  myClip.duplicateMovieClip("newClip" + i + "Clip", i + 10, {_x:i*100,
  _y:0});
}
```

Recuerde incluir un espacio después de cada expresión de una sentencia `for`.

## Escritura de sentencias while y do..while

Puede escribir sentencias `while` con el siguiente formato:

```
while (condition) {
  // sentencias
}
```

Puede escribir sentencias `do-while` con el siguiente formato:

```
do {
  // sentencias
} while (condition);
```

## Escritura de sentencias return

No utilice paréntesis [()] con ninguna sentencia return que contenga valores. Sólo debe utilizar paréntesis con sentencias `return` si contribuyen a que el valor sea más obvio, tal y como se muestra en la tercera línea del siguiente fragmento de código ActionScript:

```
return;
return myCar.paintColor;
// uso de paréntesis para que el valor de return sea obvio
return ((paintColor)? paintColor: defaultColor);
```

## Escritura de sentencias switch

- Todas las sentencias `switch` incluyen un caso predeterminado.

  El caso predeterminado es el último en una sentencia `switch`. El caso `default` incluye una sentencia `break` que evita un error de paso al siguiente caso si se añade posteriormente otro caso.
- Si un caso no contiene una sentencia break, se producirá una condición de *paso al siguiente caso* (consulte `case A` en el siguiente ejemplo de código).

  La sentencia debe incluir un comentario en lugar de la sentencia `break`, como puede ver en el siguiente ejemplo después de `case A`. En este ejemplo, si la condición coincide con el caso A, se ejecutan ambos casos, A y B.

Puede escribir sentencias `switch` con el siguiente formato:

```
switch (condition) {
case A :
  // sentencias
  // pasa al siguiente caso
case B :
  // sentencias
  break;
case Z :
  // sentencias
  break;
default :
  // sentencias
  break;
}
```

## Escritura de sentencias try..catch y try..catch..finally

Escriba las sentencias `try..catch` y `try..catch..finally` con los siguientes formatos:

```
var myErr:Error;
// try..catch
try {
  // sentencias
} catch (myErr) {
  // sentencias
}

// try..catch..finally
try {
  // sentencias
} catch (myErr) {
  // sentencias
} finally {
  // sentencias
}
```

## Utilización de sintaxis de detector

Puede escribir detectores de eventos de diversas formas en Flash 8. En los siguientes ejemplos de código se muestran algunas de las técnicas habituales. El primer ejemplo muestra una sintaxis de detector con el formato correcto, que utiliza un componente Loader para cargar contenido en un archivo SWF. El evento `progress` se inicia cuando se carga contenido y el evento `complete` indica la finalización de la carga.

```
var boxLdr:mx.controls.Loader;
var ldrListener:Object = new Object();
ldrListener.progress = function(evt:Object) {
  trace("loader loading:" + Math.round(evt.target.percentLoaded) + "%");
};
ldrListener.complete = function(evt:Object) {
  trace("loader complete:" + evt.target._name);
};
boxLdr.addEventListener("progress", ldrListener);
boxLdr.addEventListener("complete", ldrListener);
boxLdr.load("http://www.helpexamples.com/flash/images/image1.jpg");
```

Se puede introducir una pequeña variación en el primer ejemplo de esta sección si se utiliza el método `handleEvent`, pero esta técnica es algo más incómoda. Macromedia no recomienda esta técnica, pues es necesario utilizar una serie de sentencias `if..else` o una sentencia `switch` para detectar el evento que se captura.

```
var boxLdr:mx.controls.Loader;
var ldrListener:Object = new Object();

ldrListener.handleEvent = function(evt:Object) {
  switch (evt.type) {
  case "progress" :
    trace("loader loading:" + Math.round(evt.target.percentLoaded) + "%");
    break;
  case "complete" :
    trace("loader complete:" + evt.target._name);
    break;
  }
};
boxLdr.addEventListener("progress", ldrListener);
boxLdr.addEventListener("complete", ldrListener);
boxLdr.load("http://www.helpexamples.com/flash/images/image1.jpg");
```

APÉNDICE A

# Mensajes de error

A

Macromedia Flash Basic 8 y Macromedia Flash Professional 8 ofrecen una función para crear informes de error durante la compilación cuando se publica para ActionScript 2.0 (valor predeterminado). La tabla siguiente contiene una lista de mensajes de error que el compilador de Flash puede generar:

| Número de error | Texto del mensaje |
|---|---|
| 1093 | Se esperaba un nombre de clase. |
| 1094 | Se espera un nombre de clase base después de la palabra clave 'extends'. |
| 1095 | Se ha utilizado un atributo de miembro incorrectamente. |
| 1096 | No se puede repetir el mismo nombre de miembro más de una vez. |
| 1097 | Todas las funciones de miembro de clase requieren un nombre. |
| 1099 | Las definiciones de clase no admiten este tipo de declaración. |
| 1100 | Ya se ha definido una clase o interfaz con este nombre. |
| 1101 | El tipo no coincide. |
| 1102 | No hay ninguna clase que lleve por nombre '‹NombreClase›'. |
| 1103 | No hay ninguna propiedad que lleve por nombre '‹Nombrepropiedad›'. |
| 1104 | Se ha intentado llamar a una función cuando no se trataba de una función. |
| 1105 | El tipo de la declaración de asignación no coincide. Se ha encontrado [lhs-type] donde es necesario [rhs-type]. |
| 1106 | El miembro de clase es privado y no permite el acceso. |
| 1107 | Las interfaces no admiten declaraciones de variables. |
| 1108 | Las interfaces no admiten declaraciones de eventos. |
| 1109 | Las interfaces no admiten declaraciones de captador/definidor. |
| 1110 | Las interfaces no admiten miembros de clase privados. |
| 1111 | Las interfaces no admiten cuerpos de función. |

| Número de error | Texto del mensaje |
|---|---|
| 1112 | Una clase no se puede ampliar. |
| 1113 | Una interfaz no se puede ampliar. |
| 1114 | No hay ninguna interfaz definida con este nombre. |
| 1115 | Una clase no puede ampliar una interfaz. |
| 1116 | Una interfaz no puede ampliar una clase. |
| 1117 | Se espera un nombre de interfaz después de la palabra clave ‘implements’. |
| 1118 | Una clase no puede implementar otra clase, sólo interfaces. |
| 1119 | La clase debe implementar el método ‘Nombremétodo’ de la interfaz ‘Nombreinterfaz’. |
| 1120 | La implementación de un método de interfaz tiene que ser un método, no una propiedad. |
| 1121 | Una clase no puede ampliar la misma interfaz más de una vez. |
| 1122 | La implementación del método de interfaz no coincide con su definición. |
| 1123 | Esta construcción sólo está disponible en ActionScript 1.0. |
| 1124 | Esta construcción sólo está disponible en ActionScript 2.0. |
| 1125 | Las interfaces no admiten miembros de clase estáticos. |
| 1126 | La expresión devuelta tiene que coincidir con el tipo de devolución de la función. |
| 1127 | Esta función requiere una acción 'return'. |
| 1128 | Atributo utilizado fuera de una clase. |
| 1129 | Las funciones cuyo tipo de devolución sea Void no podrán devolver un valor. |
| 1130 | La cláusula 'extends' debe aparecer antes de la cláusula 'implements'. |
| 1131 | Después de ':' se espera un identificador de tipo. |
| 1132 | Las interfaces deben utilizar la palabra clave 'extends' y no 'implements'. |
| 1133 | Una clase no podrá ampliar más de una clase. |
| 1134 | Una interfaz no podrá ampliar más de una interfaz. |
| 1135 | No hay ningún método que lleve por nombre ‘<Nombremétodo>’. |
| 1136 | Las definiciones de interfaz no admiten este tipo de declaración. |
| 1137 | Las funciones set requieren exactamente un parámetro. |
| 1138 | Las funciones get no requieren ningún parámetro. |
| 1139 | Sólo se pueden definir clases en scripts de clase ActionScript 2.0 externos. |

| Número de error | Texto del mensaje |
|---|---|
| 1140 | Los scripts de clase ActionScript 2.0 sólo pueden definir construcciones de clase o interfaz. |
| 1141 | Existe un conflicto entre el nombre de esta clase, '‹A.B.C›', y el nombre de otra clase que se ha cargado, '‹A.B›'.<br>(Este error tiene lugar cuando el compilador de ActionScript 2.0 no puede compilar una clase porque el nombre completo de una clase existente forma parte del nombre de la clase con la que entra en conflicto. Por ejemplo, la compilación de la clase `mx.com.util` genera el error 1141 si la clase `mx.com` es una clase compilada.) |
| 1142 | No se pudo cargar la clase o interfaz '‹nombre de clase o interfaz›'. |
| 1143 | Las interfaces sólo pueden definirse en scripts de clase ActionScript 2.0 externos. |
| 1144 | No se puede acceder a variables de instancia en funciones estáticas. |
| 1145 | No se pueden anidar las definiciones de clases e interfaces. |
| 1146 | La propiedad a la que se hace referencia no cuenta con el atributo estático. |
| 1147 | Esta llamada a un operador de superclase no coincide con el superconstructor. |
| 1148 | En los métodos de interfaz sólo se admite el atributo público. |
| 1149 | No se puede utilizar la palabra clave 'import' como directiva. |
| 1150 | Para poder utilizar esta acción, deberá exportar su película de Flash como Flash 7. |
| 1151 | Para poder utilizar esta expresión, deberá exportar su película de Flash como Flash 7. |
| 1152 | Esta cláusula de excepción no está colocada correctamente. |
| 1153 | Una clase sólo puede tener un constructor. |
| 1154 | Los constructores no pueden devolver un valor. |
| 1155 | Los constructores no pueden especificar un tipo de devolución. |
| 1156 | Las variables no pueden ser del tipo Void. |
| 1157 | Los parámetros de función no pueden ser del tipo Void. |
| 1158 | Sólo puede accederse directamente a los miembros estáticos a través de las clases. |
| 1159 | Varias de las interfaces implementadas contienen el mismo método con tipos diferentes. |
| 1160 | Ya existe una clase o interfaz definida con ese nombre. |

| Número de error | Texto del mensaje |
|---|---|
| 1161 | No es posible borrar clases, interfaces o tipos integrados. |
| 1162 | No hay ninguna clase que lleve este nombre. |
| 1163 | La palabra clave ‹palabraclave› está reservada para ActionScript 2.0 y, por lo tanto, no se podrá utilizar aquí. |
| 1164 | No se ha concluido la definición del atributo personalizado. |
| 1165 | Sólo se podrá definir una clase o interfaz por cada archivo as de ActionScript 2.0. |
| 1166 | La clase que se está compilando, '‹A.b›', no coincide con la clase que se importó, '‹A.B›'.<br>(Este error tiene lugar cuando el nombre de una clase se escribe con mayúsculas/minúsculas diferentes de las de una clase importada. Por ejemplo, la compilación de la clase `mx.com.util` genera el error 1166 si la sentencia `import mx.Com` aparece en el archivo util.as.) |
| 1167 | Debe especificar un nombre de clase. |
| 1168 | El nombre de clase que ha introducido contiene un error de sintaxis. |
| 1169 | El nombre de interfaz que ha introducido contiene un error de sintaxis. |
| 1170 | El nombre de clase base que ha introducido contiene un error de sintaxis. |
| 1171 | El nombre de interfaz base que ha introducido contiene un error de sintaxis. |
| 1172 | Tiene que escribir un nombre de interfaz. |
| 1173 | Tiene que escribir un nombre de clase o de interfaz. |
| 1174 | El nombre de clase o de interfaz que ha introducido incluye un error de sintaxis. |
| 1175 | No se puede acceder a 'variable' desde este ámbito. |
| 1176 | El atributo 'get/set/private/public/static' aparece en varias ocasiones. |
| 1177 | Se ha utilizado un atributo de clase incorrectamente. |
| 1178 | No se pueden utilizar variables de instancia y funciones para inicializar variables estáticas. |
| 1179 | Se han descubierto circularidades de tiempo de ejecución entre las siguientes clases: ‹lista de clases definidas por el usuario›.<br>Este error de tiempo de ejecución indica que las clases personalizadas hacen referencia unas a otras de forma incorrecta. |
| 1180 | El Flash Player que se desea utilizar como destino no admite depuración. |
| 1181 | El Flash Player que se desea utilizar como destino no admite el evento releaseOutside. |

| Número de error | Texto del mensaje |
|---|---|
| 1182 | El Flash Player que se desea utilizar como destino no admite el evento dragOver. |
| 1183 | El Flash Player que se desea utilizar como destino no admite el evento dragOver. |
| 1184 | El Flash Player que se desea utilizar como destino no admite acciones de arrastre. |
| 1185 | El Flash Player que se desea utilizar como destino no admite la acción loadMovie. |
| 1186 | El Flash Player que se desea utilizar como destino no admite la acción getURL. |
| 1187 | El Flash Player que se desea utilizar como destino no admite la acción FSCommand. |
| 1188 | No se admiten declaraciones de importación en definiciones de clase o interfaz. |
| 1189 | No se puede importar la clase '‹A.B›' porque su nombre de hoja ya está dirigido a la clase que se está definiendo, '‹C.B›'.<br>Por ejemplo, la compilación de la clase `util` genera el error 1189 si la sentencia `import mx.util` aparece en el archivo util.as.) |
| 1190 | No se puede importar la clase '‹A.B›' porque su nombre de hoja ya está dirigido a la clase importada '‹C.B›'.<br>(Por ejemplo, la compilación de `import jv.util` genera el error 1190 si la sentencia `import mx.util` aparece también en el archivo AS.) |
| 1191 | Sólo se pueden inicializar las variables de instancia de una clase para compilar/cronometrar expresiones de constantes. |
| 1192 | Las funciones de miembro de clase no pueden llamarse igual que una función de constructor de una superclase. |
| 1193 | Existe un conflicto entre el nombre de esta clase, '‹NombreClase›', y el nombre de otra clase que se ha cargado. |
| 1194 | Hay que llamar primero al superconstructor en el cuerpo del constructor. |
| 1195 | El identificador '‹Nombreclase›' no se dirigirá al objeto integrado '‹NombreClase›' en tiempo de ejecución. |
| 1196 | Hay que definir la clase '‹A.B.NombreClase›' en un archivo cuya ruta relativa sea ‹'A.B›'. |
| 1197 | En el nombre de clase '‹NombreClase›' no se utiliza el carácter comodín '*' correctamente. |

| Número de error | Texto del mensaje |
|---|---|
| 1198 | Las mayúsculas/minúsculas de la función de miembro ‘<nombreclase>’ no coinciden con las del nombre de clase que se está definiendo, ‘<NombreClase>’, y no se interpretarán como constructor de clase en tiempo de ejecución. |
| 1199 | El único tipo permitido para repetidores de reproducción indefinida for-in es Cadena. |
| 1200 | Las funciones de definidor no pueden devolver un valor. |
| 1201 | Los únicos atributos permitidos para las funciones constructoras son public y private. |
| 1202 | No se encontró el archivo 'toplevel.as', que es necesario para la verificación de tipos de ActionScript 2.0. Asegúrese de que el directorio '$(LocalData)/Classes' aparece en la ruta de clases global de Preferencias de ActionScript. |
| 1203 | Branch entre ‹spanStart› y ‹spanEnd›» supera el intervalo de 32 KB. |
| 1204 | No se encontró ninguna clase o paquete que lleve por nombre '<packageName>' en el paquete '<PackageName>'. |
| 1205 | El Flash Player que se desea utilizar como destino no admite la acción FSCommand2. |
| 1206 | La función Member '<nombreFunción>' es mayor que 32 KB. |
| 1207 | La función Anonymous alrededor de la línea ‹númeroLínea› supera el intervalo de 32 KB. |
| 1208 | La función Code alrededor de la línea ‹númeroLínea› supera el intervalo de 32 KB. |
| 1210 | El nombre del paquete '<NombrePaquete>' no puede utilizarse además como nombre de método. |
| 1211 | El nombre del paquete '<NombrePaquete>' no puede utilizarse además como nombre de propiedad. |
| 1212 | No se pudo crear el archivo ASO de la clase '<NombreClase>'. Asegúrese de que el nombre de clase completo es lo suficientemente corto para que el nombre de archivo ASO, '<NombreClase.aso>', no supere los 255 caracteres. |
| 1213 | ActionScript no permite este tipo de comillas Cámbielas por comillas dobles estándar (rectas). |

APÉNDICE B

# Operadores de Flash 4 no admitidos

B

En la siguiente tabla se enumeran los operadores exclusivos de Flash 4 no admitidos por ActionScript 2.0. No utilice estos operadores a no ser que esté publicando en Flash Player 4 o versiones anteriores.

| Operador | Descripción | Asociatividad |
|---|---|---|
| `not` | NOT lógico | De derecha a izquierda |
| `and` | AND lógico | De izquierda a derecha |
| `or` | OR lógico (estilo de Flash 4) | De izquierda a derecha |
| `add` | Concatenación de cadenas (anteriormente &) | De izquierda a derecha |
| `instanceof` | Instancia de | De izquierda a derecha |
| `lt` | Menor que (versión para cadenas) | De izquierda a derecha |
| `le` | Menor o igual que (versión para cadenas) | De izquierda a derecha |
| `gt` | Mayor que (versión para cadenas) | De izquierda a derecha |
| `ge` | Mayor o igual que (versión para cadenas) | De izquierda a derecha |
| `eq` | Igual (versión para cadenas) | De izquierda a derecha |
| `ne` | Distinto (versión para cadenas) | De izquierda a derecha |

APÉNDICE C

# Teclas del teclado y valores de códigos de tecla

C

En las tablas siguientes se muestran todas las teclas de un teclado estándar y los valores correspondientes del código de tecla así como los valores de código de tecla ASCII utilizados para identificar las teclas en ActionScript:



Puede utilizar constantes de teclas para interceptar el comportamiento incorporado de las teclas presionadas. Para más información sobre el controlador on(), consulte %{on handler}% en *Referencia del lenguaje ActionScript 2.0*. Para capturar los valores de código de teclas y los de tecla ASCII mediante un archivo SWF y pulsaciones de teclas, puede utilizar el código ActionScript siguiente:

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  trace("DOWN -> Code: " + Key.getCode() + "\tACSII: " + Key.getAscii() +
  "\tKey: " + chr(Key.getAscii()));
};
Key.addListener(keyListener);
```

Para más información sobre la clase Key, consulte %{Key}% en *Referencia del lenguaje ActionScript 2.0*. Para capturar teclas al probar un archivo SWF en el entorno de edición (Control > Probar película), asegúrese de seleccionar Control > Deshabilitar métodos abreviados de teclado.

## Letras de la A a la Z y números estándar del 0 al 9

En la tabla siguiente se enumeran las teclas de un teclado estándar para las letras de la A a la Z y los números del 0 al 9, con los valores correspondientes del código de tecla usados para identificar las teclas en ActionScript.

| Tecla de letra o número | Código de tecla | Código de tecla ASCII |
|---|---|---|
| A | 65 | 65 |
| B | 66 | 66 |
| C | 67 | 67 |
| D | 68 | 68 |
| E | 69 | 69 |
| F | 70 | 70 |
| G | 71 | 71 |
| H | 72 | 72 |
| I | 73 | 73 |
| J | 74 | 74 |
| K | 75 | 75 |
| L | 76 | 76 |
| M | 77 | 77 |
| N | 78 | 78 |
| O | 79 | 79 |
| P | 80 | 80 |
| Q | 81 | 81 |
| R | 82 | 82 |
| S | 83 | 83 |
| T | 84 | 84 |
| U | 85 | 85 |
| V | 86 | 86 |
| W | 87 | 87 |
| X | 88 | 88 |
| Y | 89 | 89 |

| Tecla de letra o número | Código de tecla | Código de tecla ASCII |
|---|---|---|
| Z | 90 | 90 |
| 0 | 48 | 48 |
| 1 | 49 | 49 |
| 2 | 50 | 50 |
| 3 | 51 | 51 |
| 4 | 52 | 52 |
| 5 | 53 | 53 |
| 6 | 54 | 54 |
| 7 | 55 | 55 |
| 8 | 56 | 56 |
| 9 | 57 | 57 |
| a | 65 | 97 |
| b | 66 | 98 |
| c | 67 | 99 |
| d | 68 | 100 |
| e | 69 | 101 |
| f | 70 | 102 |
| g | 71 | 103 |
| h | 72 | 104 |
| i | 73 | 105 |
| j | 74 | 106 |
| k | 75 | 107 |
| l | 76 | 108 |
| m | 77 | 109 |
| n | 78 | 110 |
| o | 79 | 111 |
| p | 80 | 112 |
| q | 81 | 113 |
| r | 82 | 114 |

| Tecla de letra o número | Código de tecla | Código de tecla ASCII |
|---|---|---|
| s | 83 | 115 |
| t | 84 | 116 |
| u | 85 | 117 |
| v | 86 | 118 |
| w | 87 | 119 |
| x | 88 | 120 |
| y | 89 | 121 |
| z | 90 | 122 |

## Teclas del teclado numérico

En la tabla siguiente se enumeran las teclas de un teclado numérico con los valores correspondientes del código de tecla usados para identificar las teclas en ActionScript:

| Tecla del teclado numérico | Código de tecla | Código de tecla ASCII |
|---|---|---|
| Teclado numérico 0 | 96 | 48 |
| Teclado numérico 1 | 97 | 49 |
| Teclado numérico 2 | 98 | 50 |
| Teclado numérico 3 | 99 | 51 |
| Teclado numérico 4 | 100 | 52 |
| Teclado numérico 5 | 101 | 53 |
| Teclado numérico 6 | 102 | 54 |
| Teclado numérico 7 | 103 | 55 |
| Teclado numérico 8 | 104 | 56 |
| Teclado numérico 9 | 105 | 57 |
| Multiplicar | 106 | 42 |
| Sumar | 107 | 43 |
| Intro | 13 | 13 |
| Restar | 109 | 45 |

| Tecla del teclado numérico | Código de tecla | Código de tecla ASCII |
|---|---|---|
| Decimal | 110 | 46 |
| Dividir | 111 | 47 |

## Teclas de función

En la tabla siguiente se enumeran las teclas de función de un teclado estándar con los valores correspondientes del código de tecla usados para identificar las teclas en ActionScript:

| Tecla de función | Código de tecla | Código de tecla ASCII |
|---|---|---|
| F1 | 112 | 0 |
| F2 | 113 | 0 |
| F3 | 114 | 0 |
| F4 | 115 | 0 |
| F5 | 116 | 0 |
| F6 | 117 | 0 |
| F7 | 118 | 0 |
| F8 | 119 | 0 |
| F9 | 120 | 0 |
| F10 | Esta tecla está reservada por el sistema y no puede utilizarse en ActionScript. | Esta tecla está reservada por el sistema y no puede utilizarse en ActionScript. |
| F11 | 122 | 0 |
| F12 | 123 | 0 |
| F13 | 124 | 0 |
| F14 | 125 | 0 |
| F15 | 126 | 0 |

## Otras teclas

En la tabla siguiente se enumeran las teclas de un teclado estándar que no son letras, números, teclas del teclado numérico ni teclas de función, con los valores correspondientes del código de tecla usados para identificar las teclas en ActionScript:

| Key | Código de tecla | Código de tecla ASCII |
|---|---|---|
| Retroceso | 8 | 8 |
| Tabulador | 9 | 9 |
| Intro | 13 | 13 |
| Mayús | 16 | 0 |
| Control | 17 | 0 |
| Bloq Mayús | 20 | 0 |
| Esc | 27 | 27 |
| Barra espaciadora | 32 | 32 |
| Re Pág | 33 | 0 |
| Av Pág | 34 | 0 |
| Fin | 35 | 0 |
| Inicio | 36 | 0 |
| Flecha izquierda | 37 | 0 |
| Flecha arriba | 38 | 0 |
| Flecha derecha | 39 | 0 |
| Flecha abajo | 40 | 0 |
| Insert | 45 | 0 |
| Supr | 46 | 127 |
| Bloq Num | 144 | 0 |
| ScrLk | 145 | 0 |
| Pausa/Interr | 19 | 0 |
| ; : | 186 | 59 |
| = + | 187 | 61 |
| - _ | 189 | 45 |
| / ? | 191 | 47 |
| ` ~ | 192 | 96 |

| Key | Código de tecla | Código de tecla ASCII |
|---|---|---|
| [ { | 219 | 91 |
| \ \| | 220 | 92 |
| ] } | 221 | 93 |
| " ' | 222 | 39 |
| , | 188 | 44 |
| . | 190 | 46 |
| / | 191 | 47 |

Para valores de código de tecla y ASCII, utilice el código ActionScript al principio de este apéndice y presione la tecla que desee para conocer el código de la misma.

APÉNDICE D

# Escritura de scripts para versiones anteriores de Flash Player

D

ActionScript ha cambiado considerablemente con cada nueva versión de las herramientas de edición de Macromedia Flash y Flash Player. Al crear contenido para Macromedia Flash Player 8, podrá utilizar todas las prestaciones de ActionScript. Puede utilizar Flash 8 para crear contenido para versiones anteriores de Flash Player, pero no puede utilizar todos los elementos de ActionScript.

En este capítulo se proporcionan las directrices que le ayudarán a escribir scripts con una sintaxis correcta para la versión de reproductor que desea utilizar.

NOTA

Puede consultar estudios sobre penetración de versiones de Flash Player en el sitio Web de Macromedia; consulte www.macromedia.com/software/player_census/flashplayer/.

## Utilización de versiones anteriores de Flash Player

Cuando escriba los scripts, utilice la información de disponibilidad para cada elemento en *Referencia del lenguaje ActionScript 2.0* y determinar si el elemento que desea usar es compatible con la versión de Flash Player que va a emplear. También hallará información sobre los elementos que puede utilizar en la caja de herramientas Acciones; los elementos no compatibles con la versión que desea utilizar aparecerán en amarillo.

Si crea contenido para Flash Player 6, 7 u 8, debe utilizar ActionScript 2.0, que ofrece una serie de funciones importantes que no se encuentran disponibles en ActionScript 1.0, como por ejemplo una gestión de errores de compilador mejorada y capacidades de programación orientada a objetos más fiables.

Para especificar el reproductor y la versión de ActionScript que desea utilizar al publicar un documento, seleccione Archivo > Configuración de publicación y elija las opciones correspondientes en la ficha Flash. Si la versión para la que necesita crear es Flash Player 4, consulte la siguiente sección.

# Utilización de Flash 8 para crear contenido para Flash Player 4

Si desea utilizar Flash 8 para crear contenido para Flash Player 4, especifique Flash Player 4 en la ficha Flash del cuadro de diálogo Configuración de publicación (Archivo > Configuración de publicación).

ActionScript de Flash Player 4 solamente tiene un tipo de datos primitivo básico que se utiliza tanto para la manipulación numérica como para la manipulación de cadenas. Al crear una aplicación para Flash Player 4, debe utilizar los operadores de cadena desfasados que se encuentran en la categoría Eliminado en nuevas versiones > Operadores de la caja de herramientas de ActionScript.

Puede utilizar las funciones siguientes de Flash 8 al publicar para Flash Player 4:

- El operador de acceso a matriz y objeto (`[]`)
- El operador de punto (`.`)
- Los operadores lógicos, los operadores de asignación y los operadores de incremento previo e incremento/decremento posterior
- El operador de módulo (`%`) y todos los métodos y propiedades de la clase Math

Originariamente, Flash Player 4 no es compatible con los siguientes elementos de lenguaje. Flash 8 los exporta como aproximaciones de series, lo que da lugar a resultados con menor precisión numérica. Además, debido a la inclusión de aproximaciones de series en el archivo SWF, estos elementos ocupan más espacio en los archivos SWF de Flash Player 4 que en los archivos SWF de Flash Player 5 o posterior.

- Las acciones `for`, `while`, `do..while`, `break` y `continue`
- Las acciones `print()` y `printAsBitmap()`
- La acción `switch`

Para más información, consulte "Utilización de versiones anteriores de Flash Player" en la página 833.

# Utilización de Flash 8 para abrir archivos de Flash 4

ActionScript de Flash 4 sólo tenía un tipo de datos verdadero: string. Utilizaba diferentes tipos de operadores en expresiones para indicar si el valor debería ser tratado como una cadena o como un número. En las versiones posteriores de Flash, puede utilizar un conjunto de operadores para todos los tipos de datos.

Cuando se utiliza Flash 5 o una versión posterior para abrir un archivo creado en Flash 4, Flash convierte automáticamente las expresiones de ActionScript para hacerlas compatibles con la nueva sintaxis. Flash realiza las siguientes conversiones de tipos de datos y operadores:

- El operador = en Flash 4 se utilizó para la igualdad numérica. En Flash 5 y versiones posteriores, == es el operador de igualdad y = es el operador de asignación. Cualquiera de los operadores = en los archivos de Flash 4 se convierte automáticamente en ==.
- Flash realiza automáticamente conversiones de tipo para garantizar que los operadores se comportan del modo esperado. Debido a la introducción de múltiples tipos de datos, los siguientes operadores tienen nuevos significados:

  `+`, `==`, `!=`, `<>`, `<`, `>`, `>=`, `<=`

  En ActionScript de Flash 4, estos operadores siempre eran operadores numéricos. En Flash 5 y versiones posteriores, se comportan de manera diferente según el tipo de datos de los operandos. Para evitar cualquier diferencia semántica en los archivos importados, la función `Number()` se inserta alrededor de todos los operandos de estos operadores. Los números constantes son números obvios, de modo que no se incluyen en `Number()`. Para más información sobre estos operadores, consulte la tabla de operadores en "Precedencia y asociatividad de operadores" en la página 145 y "Operadores de Flash 4 no admitidos" en la página 823.
- En Flash 4, la secuencia de escape `\n` generaba un carácter de retorno de carro (ASCII 13). En Flash 5 y versiones posteriores, para cumplir el estándar ECMA-262, `\n` genera un carácter de avance de línea (ASCII 10). Una secuencia `\n` en los archivos FLA de Flash 4 se convierte automáticamente en `\r`.
- El operador `&` en Flash 4 se utilizaba para la suma de cadenas. En Flash 5 y versiones posteriores, `&` es el operador AND en modo bit. El operador de suma de cadenas ahora se denomina `add`. Cualquiera de los operadores `&` en los archivos de Flash 4 se convierten automáticamente en operadores `add`.
- Muchas de las funciones de Flash 4 no necesitan paréntesis de cierre, por ejemplo, `Get Timer`, `Set Variable`, `Stop` y `Play`. Para crear una sintaxis coherente, la función `getTimer` y todas las acciones requieren ahora paréntesis [`()`]. Estos paréntesis se añaden automáticamente durante la conversión.

- En Flash 5 y versiones posteriores, cuando la función `getProperty` se ejecuta en un clip de película que no existe, devuelve el valor `undefined`, no `0`. La sentencia `undefined == 0` es `false` en ActionScript en las versiones posteriores a Flash 4 (en Flash 4, `undefined == 1`). En Flash 5 y versiones posteriores, este problema se soluciona al convertir los archivos de Flash 4 introduciendo las funciones `Number()` en las comparaciones de igualdad. En el siguiente ejemplo, `Number()` hace que `undefined` se convierta en `0` para que la comparación sea efectiva:

```
getProperty("clip", _width) == 0
Number(getProperty("clip", _width)) == Number(0)
```

| NOTA | Si ha utilizado palabras clave de Flash 5 o de versiones anteriores como nombres de variable en ActionScript de Flash 4, la sintaxis devolverá un error al realizar la compilación en Flash 8. Para solucionar este problema, debe cambiar el nombre de sus variables en todas las ubicaciones. Para más información, consulte "Palabras reservadas" en la página 104 y "Asignación de nombre a variables" en la página 348. |
|---|---|

## Utilización de sintaxis con barras

La sintaxis con barras (`/`) se utilizó en Flash 3 y 4 para indicar la ruta de destino de un clip de película o de una variable. En la sintaxis con barras, éstas se utilizaban en lugar de los puntos y las variables iban precedidas de dos puntos, como se muestra en el siguiente ejemplo:

```
myMovieClip/childMovieClip:myVariable
```

Para escribir la misma ruta de destino en la sintaxis con puntos, que admite Flash Player 5 y versiones posteriores, utilice la sintaxis siguiente:

```
myMovieClip.childMovieClip.myVariable
```

La sintaxis con barra inversa se utilizaba comúnmente con la acción `tellTarget`, pero ya no se recomienda su uso. Ahora es recomendable emplear la acción `with` debido a que es más compatible con la sintaxis con punto. Para más información, consulte `%{tellTarget function}%` y `%{with statement}%` en *Referencia del lenguaje ActionScript 2.0*.

APÉNDICE E

# Programación orientada a objetos con ActionScript 1.0

E

La información de este apéndice procede de la documentación de Macromedia Flash MX y ofrece información sobre la utilización del modelo de objetos ActionScript 1.0 para crear scripts. Se incluye aquí por las siguientes razones:

- Si desea escribir scripts orientados a objetos compatibles con Flash Player 5, debe utilizar ActionScript 1.0.
- Si ya utiliza ActionScript 1.0 para escribir scripts orientados a objetos y no está listo para cambiar a ActionScript 2.0, puede consultar este apéndice para encontrar o revisar la información necesaria para escribir sus scripts.

Si nunca ha utilizado ActionScript para escribir scripts orientados a objetos y no necesita utilizar Flash Player 5, no haga uso de la información de este apéndice, ya que la escritura de scripts orientados a objetos con ActionScript 1.0 está desfasada. Para obtener información sobre cómo utilizar ActionScript 2.0, consulte el Capítulo 6, “Clases”, en la página 195.

Este capítulo contiene las siguientes secciones:



NOTA

En algunos ejemplos de este apéndice se utiliza el método Object.registerClass(). Este método sólo funciona con Flash Player 6 y versiones posteriores; no emplee este método si va a utilizar Flash Player 5.

# ActionScript 1.0

| NOTA | Numerosos usuarios de Flash pueden beneficiarse en gran medida de la utilización de ActionScript 2.0, particularmente en el caso de aplicaciones complejas. Para obtener información sobre cómo utilizar ActionScript 2.0, consulte el Capítulo 6, “Clases”, en la página 195. |
|---|---|

ActionScript es un lenguaje de programación orientado a objetos. Este tipo de programación utiliza *objetos*, o estructuras de datos, para agrupar propiedades y métodos que controlan el comportamiento o el aspecto del objeto. Los objetos permiten organizar y reutilizar el código. Después de definir un objeto, puede referirse a él por el nombre sin tener que redefinirlo cada vez que lo utiliza.

Una *clase* es una categoría genérica de objetos. Una clase define una serie de objetos que tienen propiedades comunes y pueden controlarse de la misma forma. Las propiedades son atributos que definen un objeto, como su tamaño, posición, color, transparencia, etc. Las propiedades se definen para una clase y los valores de las propiedades se definen para objetos individuales de la clase. Los métodos son funciones que pueden establecer o recuperar propiedades de un objeto. Por ejemplo, puede definir un método para calcular el tamaño de un objeto. Al igual que las propiedades, los métodos se definen para una clase de objeto y se invocan para objetos individuales de la clase.

ActionScript incluye varias clases incorporadas, como la clase MovieClip o la clase Sound, entre otras. También puede crear clases personalizadas para definir categorías de objetos para las aplicaciones.

Los objetos en ActionScript pueden ser meros contenedores de datos, o estar representados gráficamente en el escenario como clips de película, botones o campos de texto. Todos los clips de película son instancias de la clase incorporada MovieClip, y todos los botones son instancias de la clase incorporada Button. Cada instancia de clip de película contiene todas las propiedades (por ejemplo, `_height`, `_rotation`, `_totalframes`) y todos los métodos (por ejemplo, `gotoAndPlay()`, `loadMovie()` y `startDrag()`) de la clase MovieClip.

Para definir una clase, debe crear una función especial denominada *función constructora*. Las clases incorporadas tienen funciones constructoras incorporadas. Por ejemplo, si desea información sobre un ciclista que aparece en la aplicación, puede crear una función constructora, `Biker()`, con las propiedades `time` y `distance` y el método `getSpeed()`, que indique la velocidad a la que se desplaza el ciclista:

```
function Biker(t, d) {
  this.time = t;
  this.distance = d;
  this.getSpeed = function() {return this.time / this.distance;};
}
```

En este ejemplo, se crea una función que necesita dos elementos de información, o *parámetros*, para realizar su tarea: `t` y `d`. Al llamar a la función para crear nuevas instancias del objeto, debe pasar los parámetros a dicha función. El código siguiente crea instancias del objeto Biker denominadas `emma` y `hamish` y realiza un seguimiento de la velocidad de la instancia `emma` mediante el método `getSpeed()` del código ActionScript anterior:

```
emma = new Biker(30, 5);
hamish = new Biker(40, 5);
trace(emma.getSpeed()); // realiza un seguimiento de 6
```

En la creación de scripts orientados a objetos, las clases pueden recibir propiedades y métodos de ellas mismas, de acuerdo a un orden determinado; esta función se denomina *herencia*. La herencia puede utilizarse para ampliar o redefinir las propiedades y métodos de una clase. Una clase que hereda propiedades y métodos de otra clase recibe el nombre de *subclase*. La clase que pasa propiedades y métodos a otra clase se denomina *superclase*. Una clase puede ser a la vez subclase y superclase.

Un objeto es un tipo de datos complejo que contiene cero o más propiedades y métodos. Cada propiedad tiene un nombre y un valor, como los tiene una variable. Las propiedades están asociadas al objeto y contienen los valores que pueden cambiarse y recuperarse. Estos valores pueden ser de cualquier tipo de datos: cadena, número, booleano, objeto, clip de película o no definido. Las siguientes propiedades tienen varios tipos de datos:

```
customer.name = "Jane Doe";
customer.age = 30;
customer.member = true;
customer.account.currentRecord = 609;
customer.mcInstanceName._visible = true;
```

La propiedad de un objeto también puede ser un objeto. En la línea 4 del ejemplo anterior, `account` es una propiedad del objeto `customer` y `currentRecord` es una propiedad de `account`. El tipo de datos de la propiedad `currentRecord` es numérico.

# Creación de un objeto personalizado en ActionScript 1.0

| NOTA | Numerosos usuarios de Flash pueden beneficiarse en gran medida de la utilización de ActionScript 2.0, particularmente en el caso de aplicaciones complejas. Para obtener información sobre cómo utilizar ActionScript 2.0, consulte el Capítulo 6, “Clases”, en la página 195. |
|---|---|

Para crear un objeto personalizado, debe definir una función constructora. A una función constructora siempre se le asigna el mismo nombre que al tipo de objeto que crea. Puede utilizar la palabra clave `this` en el cuerpo de la función constructora para hacer referencia al objeto que crea el constructor; al llamar a una función constructora, Flash pasa la palabra clave `this` a la función como parámetro oculto. La función constructora que se muestra en el siguiente ejemplo crea un círculo con la propiedad `radius`:

```
function Circle(radius) {
  this.radius = radius;
}
```

Tras definir la función constructora, debe crear una instancia del objeto. Anteponga el operador `new` al nombre de la función constructora y asigne un nombre de variable a la nueva instancia. En el código siguiente, por ejemplo, se utiliza el operador `new` para crear un objeto Circle con un radio de 5 y se asigna a la variable `myCircle`:

```
myCircle = new Circle(5);
```

| NOTA | Un objeto tiene el mismo ámbito que la variable a la que se ha asignado. |
|---|---|

# Asignación de métodos a un objeto personalizado en ActionScript 1.0

NOTA

Numerosos usuarios de Flash pueden beneficiarse en gran medida de la utilización de ActionScript 2.0, particularmente en el caso de aplicaciones complejas. Para obtener información sobre cómo utilizar ActionScript 2.0, consulte el Capítulo 6, “Clases”, en la página 195.

Los métodos de un objeto pueden definirse en la función constructora del objeto. No obstante, esta técnica no se recomienda porque define el método cada vez que se utiliza la función constructora. El siguiente ejemplo crea los métodos `getArea()` y `getDiameter()`: y realiza un seguimiento del área y el diámetro de la instancia `myCircle` construida con un radio definido en 55:

```
function Circle(radius) {
  this.radius = radius;
  this.getArea = function(){
    return Math.PI * this.radius * this.radius;
  };
  this.getDiameter = function() {
    return 2 * this.radius;
  };
}
var myCircle = new Circle(55);
trace(myCircle.getArea());
trace(myCircle.getDiameter());
```

Cada función constructora tiene una propiedad `prototype` que se crea automáticamente al definir la función. La propiedad `prototype` indica los valores predeterminados de la propiedad para los objetos creados con esa función. Cada nueva instancia de un objeto tiene una propiedad `__proto__` que hace referencia a la propiedad `prototype` de la función constructora que la ha creado. Por consiguiente, si se asignan métodos a la propiedad `prototype` de un objeto, dichos métodos sólo estarán disponibles para las instancias nuevas creadas del objeto. Lo mejor es asignar un método a la propiedad `prototype` de la función constructora, ya que de este modo sólo existe en un lugar y las nuevas instancias del objeto (o clase) hacen referencia al mismo. Pueden utilizarse las propiedades `prototype` y `__proto__` para ampliar los objetos de modo que pueda reutilizarse el código de forma orientada a los objetos. Para más información, consulte “Creación de herencia en ActionScript 1.0” en la página 845.

En el procedimiento siguiente se muestra cómo asignar un método `getArea()` a un objeto Circle personalizado.

**Para asignar un método a un objeto personalizado:**

1. Defina la función constructora `Circle()`:

```
function Circle(radius) {
  this.radius = radius;
}
```

2. Defina el método `getArea()` del objeto Circle. El método `getArea()` calcula el área del círculo. En el siguiente ejemplo, puede utilizar un literal de función para definir el método `getArea()` y asignar la propiedad `getArea` al objeto prototipo del círculo:

```
Circle.prototype.getArea = function () {
     return Math.PI * this.radius * this.radius;
};
```

3. En el siguiente ejemplo se crea una instancia del objeto Circle:

```
var myCircle = new Circle(4);
```

4. Llame al método `getArea()` del nuevo objeto `myCircle` mediante el siguiente código:

```
var myCircleArea = myCircle.getArea();
trace(myCircleArea); // realiza un seguimiento de 50.265...
```

   ActionScript busca el objeto `myCircle` para el método `getArea()`. Dado que el objeto no tiene un método `getArea()`, se busca el objeto prototipo `Circle.prototype` para `getArea()`. ActionScript lo encuentra, lo llama y traza `myCircleArea`.

# Definición de métodos de controlador de eventos en ActionScript 1.0

| NOTA | Numerosos usuarios de Flash pueden beneficiarse en gran medida de la utilización de ActionScript 2.0, particularmente en el caso de aplicaciones complejas. Para obtener información sobre cómo utilizar ActionScript 2.0, consulte el Capítulo 6, “Clases”, en la página 195. |
|---|---|

Puede crear una clase de ActionScript para clips de película y definir los métodos de controlador de eventos del objeto prototipo de esta nueva clase. La definición de los métodos del objeto prototipo hace que todas las instancias de este símbolo respondan a estos eventos del mismo modo.

También puede añadir métodos de controlador de eventos `onClipEvent()` u `on()` a una sola instancia para proporcionar instrucciones exclusivas que sólo se ejecutarán cuando se produzca el evento de dicha instancia. Los métodos `onClipEvent()` y `on()` no prevalecen sobre el método de controlador de eventos; ambos eventos hacen que se ejecuten sus scripts. Sin embargo, si define los métodos de controlador de eventos del objeto prototipo y también define un método de controlador de eventos para una instancia determinada, la definición de la instancia prevalece sobre la definición del objeto prototipo.

### Para definir un método de controlador de eventos en un objeto prototipo de un objeto:

1. Cree un símbolo de clip de película y establezca el identificador de vinculación `theID` mediante la selección del símbolo en el panel Biblioteca y eligiendo Vinculación en el menú emergente de Biblioteca.
2. En el panel Acciones (Ventana > Acciones), utilice la sentencia `function` para definir una clase nueva, tal como se muestra en el siguiente ejemplo:

   ```
   // defina una clase
   function myClipClass() {}
   ```

   La nueva clase se asigna a todas las instancias de clip de película añadidas a la aplicación mediante la línea de tiempo o con el método `attachMovie()` o `duplicateMovieClip()`. Si desea que estos clips de película puedan acceder a los métodos y propiedades del objeto incorporado MovieClip, deberá hacer que la nueva clase herede de la clase MovieClip.
3. Introduzca código como el del siguiente ejemplo:

   ```
   // heredar de la clase MovieClip
   myClipClass.prototype = new MovieClip();
   ```

   Ahora, la clase `myClipClass` hereda todas las propiedades y métodos de la clase MovieClip.
4. Introduzca código como el del siguiente ejemplo para definir los métodos de controlador de eventos para la nueva clase:

   ```
   // defina los métodos de controlador de eventos para la clase myClipClass
   myClipClass.prototype.onLoad = function() {trace("movie clip loaded");}
   myClipClass.prototype.onEnterFrame = function() {trace("movie clip
     entered frame");}
   ```
5. Seleccione Ventana > Biblioteca para abrir el panel Biblioteca si aún no está abierto.
6. Seleccione los símbolos que desea asociar con la nueva clase y elija Vinculación en el menú emergente del panel Biblioteca.
7. En el cuadro de diálogo Propiedades de vinculación, seleccione Exportar para ActionScript.

8. Introduzca un identificador de vinculación en el cuadro de texto Identificador.

   El identificador de vinculación debe ser el mismo para todos los símbolos que desea asociar con la nueva clase. En el ejemplo `myClipClass`, el identificador es `theID`.

9. Introduzca código como el del siguiente ejemplo en el panel Acciones:

```
// registre la clase
Object.registerClass("theID", myClipClass);
this.attachMovie("theID","myName",1);
```

   Este paso registra cualquier símbolo cuyo identificador de vinculación sea `theID` con la clase `myClipClass`. Todas las instancias de `myClipClass` tienen métodos de controlador de eventos que se comportan tal como se definió en el paso 4. También se comportan como todas las instancias de la clase MovieClip, puesto que en el paso 3 se indicó que la nueva clase debe heredar de la clase MovieClip.

   El código completo se muestra en el siguiente ejemplo:

```
function myClipClass(){}

myClipClass.prototype = new MovieClip();
myClipClass.prototype.onLoad = function(){
  trace("movie clip loaded");
}
myClipClass.prototype.onPress = function(){
  trace("pressed");
}

myClipClass.prototype.onEnterFrame = function(){
  trace("movie clip entered frame");
}

myClipClass.prototype.myfunction = function(){
  trace("myfunction called");
}

Object.registerClass("myclipID",myClipClass);
this.attachMovie("myclipID","clipName",3);
```

# Creación de herencia en ActionScript 1.0

| NOTA | Numerosos usuarios de Flash pueden beneficiarse en gran medida de la utilización de ActionScript 2.0, particularmente en el caso de aplicaciones complejas. Para obtener información sobre cómo utilizar ActionScript 2.0, consulte el Capítulo 6, "Clases", en la página 195. |
|---|---|

La herencia es una forma de organizar, ampliar y reutilizar funciones. Las subclases heredan las propiedades y los métodos de las superclases y, además, añaden sus propios métodos y propiedades especializados. Para poner un ejemplo real, podemos decir que Bicicleta sería una superclase y que Bicicleta de montaña (BTT) y Triciclo serían subclases de dicha superclase. Ambas subclases contienen, o *heredan,* los métodos y las propiedades de la superclase (por ejemplo, `wheels`). Cada subclase tiene, asimismo, sus propias propiedades y métodos que se amplían a la superclase (la subclase BTT, por ejemplo, tendría una propiedad `gears`). Para crear herencia en ActionScript pueden utilizarse los elementos `prototype` y `__proto__`.

Todas las funciones constructoras tienen una propiedad `prototype` que se crea automáticamente cuando se define la función. La propiedad `prototype` indica los valores predeterminados de la propiedad para los objetos creados con esa función. La propiedad `prototype` puede utilizarse para asignar propiedades y métodos a una clase. Para más información, consulte "Asignación de métodos a un objeto personalizado en ActionScript 1.0" en la página 841.

Todas las instancias de una clase tienen una propiedad `__proto__` que indica de qué objeto heredan métodos y propiedades. Al utilizar una función constructora para crear un objeto, se define la propiedad `__proto__` para hacer referencia a la propiedad `prototype` de su función constructora.

La herencia se comporta de acuerdo a una jerarquía determinada. Cuando se llama a la propiedad o al método de un objeto, ActionScript busca en el objeto para ver si existe tal elemento. Si no existe, ActionScript busca la información (`myObject.__proto__`) en la propiedad `__proto__` del objeto. Si la propiedad no pertenece al objeto `__proto__` del objeto, ActionScript busca en `myObject.__proto__.__proto__`, y así sucesivamente.

En el ejemplo siguiente se define la función constructora `Bike()`:

```
function Bike(length, color) {
  this.length = length;
  this.color = color;
  this.pos = 0;
}
```

En el código siguiente se añade el método `roll()` a la clase Bike:

```
Bike.prototype.roll = function() {return this.pos += 20;};
```

A continuación puede realizar un seguimiento de la posición de la bicicleta con el siguiente código:

```
var myBike = new Bike(55, "blue");
trace(myBike.roll());   // realiza un seguimiento de 20.
trace(myBike.roll());   // realiza un seguimiento de 40.
```

En lugar de añadir `roll()` a la clase MountainBike y a la clase Tricycle, puede crear la clase MountainBike con la clase Bike como superclase, como se muestra en el siguiente ejemplo:

```
MountainBike.prototype = new Bike();
```

A continuación, puede llamar al método `roll()` de MountainBike, como se indica en el siguiente ejemplo:

```
var myKona = new MountainBike(20, "teal");
trace(myKona.roll());   // realiza un seguimiento de 20
```

Los clips de película no heredan entre sí. Para crear herencia en los clips de película, puede utilizar el método `Object.registerClass()` para asignar una clase distinta de MovieClip a los clips de película.

# Adición de propiedades de captador/definidor a objetos en ActionScript 1.0

| NOTA | Numerosos usuarios de Flash pueden beneficiarse en gran medida de la utilización de ActionScript 2.0, particularmente en el caso de aplicaciones complejas. Para obtener información sobre cómo utilizar ActionScript 2.0, consulte el Capítulo 6, “Clases”, en la página 195. |
|---|---|

Puede crear propiedades de captador/definidor para un objeto mediante el método `Object.addProperty()`.

La *función captador* carece de parámetros. El valor devuelto puede ser de cualquier tipo. El tipo de valor puede cambiar según la invocación. El valor devuelto se trata como el valor actual de la propiedad.

La *función definidor* acepta un parámetro: el nuevo valor de la propiedad. Por ejemplo, si la propiedad x se asigna mediante la sentencia x = 1, la función definidor recibirá el parámetro 1 del tipo número. El valor devuelto de la función definidor se ignora.

Cuando Flash lee una propiedad de captador/definidor, invoca la función captador y el valor devuelto por la función se convierte en un valor de `prop`. Cuando Flash escribe una propiedad de captador/definidor, invoca la función definidor y le pasa el nuevo valor como parámetro. Si existe una propiedad con ese nombre concreto, la nueva propiedad lo sobrescribe.

Puede añadir propiedades de captador/definidor (getter/setter) a los objetos prototipo. Si añade una propiedad de captador/definidor a un objeto prototipo, todas las instancias del objeto que heredan el objeto prototipo heredarán la propiedad de captador/definidor. Puede añadir una propiedad de captador/definidor a una ubicación, el objeto prototipo, y aplicarla a todas las instancias de una clase (como si se añadieran métodos a objetos prototipo). Si se invoca una función captador/definidor para una propiedad de captador/definidor de un objeto prototipo heredado, la referencia que se pasa a la función captador/definidor será el objeto referenciado originalmente, no el objeto prototipo.

El comando Depurar > Mostrar variables en el modo de prueba es compatible con las propiedades de captador/definidor que se añaden a los objetos mediante el método `Object.addProperty()`. Las propiedades que se añaden así a un objeto se muestran junto con otras propiedades del mismo en el panel Salida. Las propiedades de captador/definidor se identifican en el panel Salida con el prefijo `[getter/setter]`. Para más información sobre el comando Mostrar variables, consulte "Utilización del panel Salida" en la página 768.

# Utilización de las propiedades del objeto Function en ActionScript 1.0

NOTA

Numerosos usuarios de Flash pueden beneficiarse en gran medida de la utilización de ActionScript 2.0, particularmente en el caso de aplicaciones complejas. Para obtener información sobre cómo utilizar ActionScript 2.0, consulte el Capítulo 6, "Clases", en la página 195.

Puede especificar el objeto al que se aplica una función y los valores de los parámetros que se pasan a la función mediante los métodos `call()` y `apply()` del objeto Function. Cada función de ActionScript se representa mediante un objeto Function, de modo que todas las funciones admiten los métodos `call()` y `apply()`. Al crear una clase personalizada mediante una función constructora, o al definir métodos para una clase personalizada utilizando una función, puede invocar los métodos `call()` y `apply()` para la función.

# Invocación de una función mediante el método Function.call() en ActionScript 1.0

| NOTA | Numerosos usuarios de Flash pueden beneficiarse en gran medida de la utilización de ActionScript 2.0, particularmente en el caso de aplicaciones complejas. Para obtener información sobre cómo utilizar ActionScript 2.0, consulte el Capítulo 6, "Clases", en la página 195. |
|---|---|

El método `Function.call()` invoca la función representada por un objeto Function.

En casi todos los casos puede utilizarse el operador de llamada de función (`()`) en lugar del método `call()`. El operador de llamada de función hace que el código sea conciso y legible. El método `call()` es de gran utilidad cuando debe controlarse explícitamente el parámetro `this` de la llamada de función. Normalmente, si se invoca una función como método de un objeto, el parámetro `this` se establece en `myObject` dentro del cuerpo de la función, como se muestra en el siguiente ejemplo:

```
myObject.myMethod(1, 2, 3);
```

En algunos casos, es posible que desee que `this` haga referencia a otro elemento; por ejemplo, si debe invocarse una función como un método de un objeto, pero en realidad no se almacena como método de dicho objeto, como se muestra en el siguiente ejemplo:

```
myObject.myMethod.call(myOtherObject, 1, 2, 3);
```

Puede pasar el valor `null` para el parámetro *`thisObject`* para invocar una función como función regular y no como un método de un objeto. Por ejemplo, las llamadas de función siguientes son equivalentes:

```
Math.sin(Math.PI / 4)
Math.sin.call(null, Math.PI / 4)
```

### Para invocar una función mediante el método Function.call():

- Utilice la siguiente sintaxis:

  *`myFunction`*`.call(`*`thisObject`*`, `*`parameter1`*`, ..., `*`parameterN`*`)`

  Este método utiliza los siguientes parámetros:

  - El parámetro *`thisObject`* especifica el valor de `this` en el cuerpo de la función.
  - Los parámetros *`parameter1`*`..., `*`parameterN`* especifican parámetros que se pasan a *`myFunction`*. Puede especificar cero o más parámetros.

# Especificación de un objeto al que se aplica una función mediante Function.apply() en ActionScript 1.0

| NOTA | Numerosos usuarios de Flash pueden beneficiarse en gran medida de la utilización de ActionScript 2.0, particularmente en el caso de aplicaciones complejas. Para obtener información sobre cómo utilizar ActionScript 2.0, consulte el Capítulo 6, “Clases”, en la página 195. |
|---|---|

El método `Function.apply()` especifica el valor de `this` que debe utilizarse en una función que llama ActionScript. Este método también especifica parámetros que deben pasarse a la función llamada.

Los parámetros se especifican como un objeto Array. Suele ser útil cuando no se conoce el número de parámetros hasta que se ejecuta el script.

Para más información, consulte `%{apply (método Function.apply)}%` en *Referencia del lenguaje ActionScript 2.0*.

### Para especificar el objeto al que se aplica una función mediante el método Function.apply():

- Utilice la siguiente sintaxis:

  *myFunction*`.apply(`*thisObject*`, `*argumentsObject*`)`

  Este método utiliza los siguientes parámetros:

  - El parámetro *thisObject* especifica el objeto al que se aplica *myFunction*.
  - El parámetro *argumentsObject* define una matriz cuyos elementos se pasan a *myFunction* como parámetros.

APÉNDICE F

# Terminología

F

Como con cualquier otro lenguaje de creación de scripts, ActionScript utiliza su propia terminología. Macromedia Flash también usa su terminología específica. En la siguiente lista se facilita una introducción a importantes términos de ActionScript y Flash relativos a la programación con ActionScript y específicos al uso de un entorno de edición de Flash.

**Editor de ActionScript**: editor de código que se muestra en el panel Acciones y la ventana Script. Está formado por distintas funciones, como el formato automático, la visualización de caracteres ocultos y partes de código de color de los scripts. (Consulte también: ventana Script, panel Acciones).

**Panel Acciones**: panel del entorno de edición de Flash en el que se escribe código ActionScript.

**Función anónima**: función sin nombre a la que se hace referencia cuando se crea. Para obtener información y ver un ejemplo, consulte "Escritura de funciones anónimas y callback" en la página 176.

**Mapa de bits**: se refiere a texto de mapa de bits que no utiliza variaciones de color para que sus bordes dentados se muestren más suaves, a diferencia del texto suavizado (consulte la definición siguiente).

**Suavizado**: se refiere a afinar texto de modo que los bordes de los caracteres no aparezcan excesivamente dentados en la pantalla. La opción de suavizado en Flash permite obtener un texto más legible alineando los contornos del texto a los límites de los píxeles y resulta efectivo para representar con mayor claridad los tamaños de fuente más pequeños.

**Matrices**: objetos cuyas propiedades se identifican mediante números que representan sus posiciones en la estructura. Básicamente, una matriz es una lista de elementos.

**Entorno de edición**: espacio de trabajo de Flash que incluye todos los elementos de la interfaz de usuario. Los archivos de script o FLA (en la ventana Script) se crean a través del entorno de edición.

**Imágenes de mapa de bits** (o gráficos *raster*): son normalmente imágenes realistas fotográficas o gráficos con una gran cantidad de detalles. Cada píxel (o *bit*) de la imagen contiene un dato y la combinación de estos bits forma la imagen. Los mapas de bits se pueden guardar en formatos de archivo JPEG, BMP o GIF. Otro tipo de gráfico, diferente al mapa de bits, es el *vector*.

**Booleano**: valor verdadero (`true`) o falso (`false`).

**Caché**: se refiere a la información que se vuelve a utilizar en la aplicación o la que se almacena en el equipo de forma que se pueda volver a utilizar. Por ejemplo, si se descarga una imagen de Internet, a menudo se almacena en caché para que se pueda volver a ver sin descargar los datos de la misma.

**Funciones callback**: funciones anónimas que se asocian a un determinado evento. Una función llama a una función callback tras producirse un determinado evento, por ejemplo, después de que algo termine de cargarse (`onLoad()`) o finalice su animación (`onMotionFinished()`). Para más información y ver ejemplos, consulte "Escritura de funciones anónimas y callback" en la página 176.

**Caracteres**: letras, numerales y puntuación que se combinan para formar cadenas. Los parámetros se denominan a veces *glifos*.

**Clase:** tipo de datos que puede emplearse para definir un nuevo tipo de objeto. Para definir una clase, se utiliza la palabra clave `class` en un archivo de script externo (no en un script que se esté escribiendo en el panel Acciones).

**Ruta de clases**: se refiere a la lista de carpetas en las que Flash busca las definiciones de interfaces y clases. Al crear un archivo de clase, es necesario guardarlo en uno de los directorios especificados en la ruta de clases o en un subdirectorio de ésta. La ruta de clases existe a nivel global (aplicación) y de documento.

**Constante:** elemento cuyo valor no cambia. Por ejemplo, la constante `Key.TAB` siempre tiene el mismo significado: indica la tecla Tabulador de un teclado. Las constantes son útiles para comparar valores.

**Funciones constructoras** (o *constructores*): funciones que se utilizan para definir (inicializar) las propiedades y métodos de una clase. Por definición, los constructores son funciones incluidas en definiciones de clases que tienen el mismo nombre que la clase. Por ejemplo, el código siguiente define una clase Circle e implementa una función constructora:

```
// archivo Circle.as
class Circle {
  private var circumference:Number;
// constructor
  function Circle(radius:Number){
    this.circumference = 2 * Math.PI * radius;
  }
}
```

El término *constructor* también se emplea para crear un objeto (crear instancias del mismo) basado en una clase determinada. Las sentencias siguientes son llamadas a las funciones constructoras de la clase Array incorporada y de la clase Circle personalizada:

```
var my_array:Array = new Array();
var my_circle:Circle = new Circle(9);
```

**Tipos de datos**: describe el tipo de información que puede contener una variable o un elemento de ActionScript. Los tipos de datos incorporados de ActionScript son: cadena, número, valor booleano, objeto, clip de película, función, nulo y no definido. Para más información, consulte "Tipos de datos" en la página 328.

**Fuentes de dispositivo**: fuentes especiales en Flash que no están incorporadas en un archivo SWF de Flash. En lugar de ello, Flash Player utiliza la fuente disponible en el equipo local más parecida a la fuente de dispositivo. Dado que los contornos de fuente no están incorporados, el tamaño del archivo SWF es menor que cuando se utilizan contornos de fuente incorporados. No obstante, como las fuentes de dispositivo no están incorporadas, el aspecto del texto que se cree puede ser diferente al esperado en los sistemas que no tengan instalada una fuente que corresponda a la fuente de dispositivo. Flash incluye tres fuentes de dispositivo: _sans (similar a la Helvetica y Arial), _serif (similar a la Times Roman) y _typewriter (similar a la Courier).

**Sintaxis con puntos**: se refiere a cuando se utiliza un operador de punto (.) (sintaxis con puntos) para acceder a propiedades o métodos que pertenecen a un objeto o instancia del escenario mediante ActionScript. También puede utilizar el operador de punto para identificar la ruta de destino de una instancia (como, por ejemplo, un clip de película), una variable, una función o un objeto. Una expresión que utiliza la sintaxis con puntos empieza por el nombre del objeto o clip de película seguido de un punto y termina con el elemento que desee especificar.

**Eventos**: tienen lugar durante la reproducción de un archivo SWF. Por ejemplo, cuando se carga un clip de película se generan diferentes eventos: la cabeza lectora accede a un fotograma, el usuario hace clic en un botón o clip de película o el usuario introduce información mediante el teclado.

**Controladores de eventos**: eventos especiales que se gestionan cuando se hace clic con el ratón o cuando finaliza la carga de datos. Se distinguen dos tipos de controladores de eventos de ActionScript: métodos de controlador de eventos y detectores de eventos. (También existen dos controladores de eventos, %{on handler}% y %{onClipEvent handler}%, que pueden asignarse directamente a botones y clips de película.) En la caja de herramientas Acciones, cada objeto de ActionScript que tiene métodos de controlador de eventos o detectores de eventos cuenta con una subcategoría denominada Eventos o Detectores. Algunos comandos pueden utilizarse como controladores de eventos y como detectores de eventos y se incluyen en ambas subcategorías. Para más información sobre gestión de eventos, consulte "Gestión de eventos" en la página 305.

**Expresión:** cualquier combinación válida de símbolos de ActionScript que representan un valor. Una expresión está formada por operadores y operandos. Por ejemplo, en la expresión `x + 2`, `x` y `2` son operandos y `+` es un operador.

**Contenedor de Flash Player**: se refiere al sistema que contiene la aplicación Flash, como un navegador o la aplicación de escritorio. Se puede añadir ActionScript y JavaScript para facilitar la comunicación entre el contenedor Flash Player y un archivo SWF.

**FlashType**: se refiere a la tecnología de representación avanzada de fuentes que utiliza Flash 8. Por ejemplo, el texto de mapa de bits para legibilidad utiliza la tecnología de representación de FlashType mientras que el texto de mapa de bits para animación no. Para obtener información, consulte "Representación de fuentes y texto suavizado" en la página 434.

**Scripts de fotograma**: bloques de código que se añaden a un fotograma en una línea de tiempo.

**Funciones**: bloques de código reutilizable que aceptan parámetros y pueden devolver un valor. Para más información, consulte "Funciones y métodos" en la página 169.

**Literales de función**: funciones sin nombre que se declaran en una expresión en lugar de en una sentencia. Son de gran utilidad si se desea usar una función temporalmente o utilizar una función en el código en el que podría emplearse una expresión.

**IDE**: siglas que corresponden a "entorno de desarrollo integrado, Integrated Development Environment". Se trata de una aplicación en la que un desarrollador puede codificar, probar y depurar aplicaciones en un entorno interactivo. En ocasiones, la herramienta de edición de Flash se denomina un IDE.

**Identificador:** nombre que se utiliza para identificar una variable, una propiedad, un objeto, una función o un método. El primer carácter debe ser una letra, un carácter de subrayado (_) o un símbolo de dólar ($). Los caracteres siguientes deben ser una letra, un número, un carácter de subrayado o un símbolo de dólar. Por ejemplo, `firstName` es el nombre de una variable.

**Instancias** son objetos que contienen todas las propiedades y métodos de una clase concreta. Por ejemplo, todas las matrices son instancias de la clase Array, de modo que puede utilizar cualquiera de los métodos o propiedades de la clase Array con cualquier instancia de matriz.

**Nombres de instancia** son nombres exclusivos que permiten convertir en destino instancias creadas por usted, o instancias de clip de película y botón en el escenario. Por ejemplo, en el siguiente código, "names" y "studentName" son nombres de instancia para dos objetos, una matriz y una cadena:

```
var names:Array = new Array();
var studentName:String = new String();
```

Utilice el inspector de propiedades para asignar nombres de instancia a las instancias del escenario. Por ejemplo, un símbolo maestro de la biblioteca podría denominarse `counter` y las dos instancias de dicho símbolo en el archivo SWF podrían denominarse `scorePlayer1_mc` y `scorePlayer2_mc`. En el siguiente código se utilizan nombres de instancia para establecer una variable denominada `score` en cada instancia del clip de película:

```
this.scorePlayer1_mc.score = 0;
this.scorePlayer2_mc.score = 0;
```

Se puede emplear strict data typing al crear instancias para que aparezcan sugerencias para el código a medida que se escribe éste.

**Palabra clave:** palabra reservada que tiene un significado especial. Por ejemplo, `var` es una palabra clave que se utiliza para declarar variables locales. Una palabra clave no puede utilizarse como identificador. Por ejemplo, `var` no es un nombre de variable válido. Para obtener una lista de palabras clave, consulte "Palabras clave" en la página 103 y "Palabras reservadas" en la página 104.

**Literales**: representan valores que tienen un determinado tipo, como literales numéricos o literales de cadena. Los literales no se almacenan en una variable. Un literal es un valor que aparece directamente en el código y es un valor constante (sin cambiar) de los documentos de Flash. Consulte también *literal de función* y *literal de cadena*.

**Método:** función asociada a una clase. Por ejemplo, `sortOn()` es un método incorporado asociado a la clase Array. También puede crear funciones que actúen como métodos, ya sea para objetos basados en clases incorporadas o para objetos basados en clases que haya creado. Por ejemplo, en el código siguiente, `clear()` pasa a ser un método de un objeto `controller` definido anteriormente:

```
function reset(){
  this.x_pos = 0;
  this.y_pos = 0;
}
controller.clear = reset;
controller.clear();
```

Los siguientes ejemplos muestran cómo crear métodos de una clase:

```
//Ejemplo de ActionScript 1.0
A = new Object();
A.prototype.myMethod = function() {
  trace("myMethod");
}

//Ejemplo de ActionScript 2.0
class B {
  function myMethod() {
    trace("myMethod");
  }
}
```

**Función con nombre**: tipo de función que se crea con frecuencia en el código ActionScript para realizar todo tipo de acciones. Para obtener información y ver un ejemplo, consulte "Escritura de funciones con nombre" en la página 175.

**Código de objeto**: código ActionScript que se asigna a las instancias. Para añadir código de objeto, seleccione una instancia en el escenario y, a continuación, escriba código en el panel Acciones. No se recomienda asignar código a objetos en el escenario. Para obtener información sobre prácticas recomendadas, consulte "Recomendaciones y convenciones de codificación para ActionScript 2.0" en la página 775.

**Objeto**: conjunto de propiedades y métodos; cada objeto tiene su propio nombre y es una instancia de una clase determinada. Los objetos incorporados están predefinidos en el lenguaje ActionScript. Por ejemplo, la clase incorporada Date ofrece información procedente del reloj del sistema.

**Operador:** término que calcula un nuevo valor a partir de uno o más valores. Por ejemplo, el operador de suma (+) suma dos o más valores para generar un nuevo valor. Los valores manipulados por los operadores se denominan *operandos*.

**Parámetro** (denominado también *argumento*): marcador de posición que permite pasar valores a las funciones. La siguiente función `welcome()`, por ejemplo, utiliza dos valores que recibe de los parámetros `firstName` y `hobby`:

```
function welcome(firstName:String, hobby:String):String {
var welcomeText:String = "Hello, " + firstName + ". I see you enjoy " +
  hobby +".";
  return welcomeText;
}
```

**Paquetes**: directorios que contienen uno o más archivos de clase y que residen en un directorio de ruta de clases determinado (consulte "Paquetes" en la página 198).

**Fijación de scripts**: permite fijar varios scripts de diversos objetos y trabajar con ellos simultáneamente en el panel Acciones. Funciona mejor con el navegador de scripts.

**JPEG progresivas**: imágenes que se construyen de forma gradual y se muestran a medida que se descargan de un servidor. Una imagen JPEG normal se mostrará línea a línea cuando se descargue de un servidor.

**Propiedad:** atributo que define un objeto. Por ejemplo, `length` es una propiedad de todas las matrices que especifica el número de elementos de la matriz.

**Signos**: caracteres especiales que ayudan a formar código ActionScript. Existen diversos signos de lenguaje en Flash. El tipo más habitual de signos son el punto y coma (;), los dos puntos (:), los paréntesis [()] y las llaves ({}). Cada uno de estos signos tiene un significado específico en el lenguaje de Flash y contribuye a definir tipos de datos, terminar sentencias o estructurar el código ActionScript.

**Asistente de script**: nuevo modo asistido del panel Acciones. El asistente de script facilita la creación de scripts sin que sea necesario tener un profundo conocimiento de ActionScript. Ayuda a crear sus scripts seleccionando elementos de la caja de herramientas Acciones del panel Acciones y proporciona una interfaz de campos de texto, botones de opción y casillas de verificación que proponen las variables correctas y otros componentes del lenguaje de creación de scripts. Esta función es similar al *modo normal* de las ediciones anteriores de la herramienta de edición de Flash.

**Panel Script**: panel del panel Acciones o la ventana Script donde se escribe el código ActionScript.

**Ventana Script**: entorno de edición de código en el que puede crear y modificar scripts externos, como archivos ActionScript y JavaScript de Flash. Por ejemplo, seleccione Archivo > Nuevo y, a continuación, elija Archivo ActionScript para utilizar la ventana Script con objeto de escribir un archivo de clase.

**Sentencias** son elementos del lenguaje que realizan o especifican una acción. Por ejemplo, la sentencia `return` devuelve un resultado como valor de la función en la que se ejecuta. La sentencia `if` evalúa una condición para determinar la siguiente acción que debe realizarse. La sentencia `switch` crea una estructura ramificada para sentencias de ActionScript.

**Cadena**: secuencia de caracteres y tipo de datos. Consulte "Cadenas y la clase String" en la página 480 para más información.

**Literal de cadena**: secuencia de caracteres encerrada completamente por caracteres de comillas rectos. Los caracteres son en sí mismos un valor de datos, no una referencia a estos. Un literal de cadena no es un objeto String. Para más información, consulte "Cadenas y la clase String" en la página 480.

**Superficie**: clip de película que tiene activado el indicador de caché de mapa de bits. Para obtener información sobre la caché de mapa de bits, consulte "Asignación de caché para un clip de película" en la página 398.

**Sintáxis**: se refiere a la gramática y la ortografía de un lenguaje que le permite programar. El compilador no comprende la sintaxis incorrecta, por lo que observará errores o advertencias en el panel Salida cuando intente comprobar el documento en el entorno de prueba. Por consiguiente, la sintaxis es un conjunto de reglas y directrices que le ayudan a formar código ActionScript correcto.

**Rutas de destino:** direcciones jerárquicas de nombres de instancias de clips de película, variables y objetos de un archivo SWF. El nombre de una instancia de clip de película se asigna en el inspector de propiedades del clip de película. (La línea de tiempo principal siempre tiene el nombre `_root`.) Se puede utilizar una ruta de destino para dirigir una acción a un clip de película u obtener o definir el valor de una variable o propiedad. Por ejemplo, la sentencia siguiente es la ruta de destino a la propiedad `volume` del objeto denominado `stereoControl`:

```
stereoControl.volume
```

**Texto**: serie de una o más cadenas que se puede mostrar en un campo de texto o dentro de un componente de la interfaz de usuario.

**Campos de texto**: elementos visuales del escenario que permiten mostrar texto a un usuario y que se pueden crear mediante la herramienta Texto o código ActionScript. Flash permite establecer campos de texto como editables (de sólo lectura), posibilita la aplicación de formato HTML, activa la compatibilidad con varias líneas, la creación de máscaras de contraseñas o la aplicación de una hoja de estilos CSS al texto con formato HTML.

**Formato de texto**: se puede aplicar a un campo de texto o a ciertos caracteres dentro de un campo de texto. A continuación se muestran algunos ejemplos de opciones de formato de texto que se pueden aplicar: alineación, sangrado, negrita, color, tamaño de fuente, anchura de márgenes, cursiva y espaciado entre caracteres.

**Funciones de nivel superior**: funciones que no pertenecen a una clase (a veces se denominan *funciones predefinidas* o *incorporadas*), lo que significa que pueden llamarse sin un constructor. Ejemplos de funciones que están incorporadas en el nivel superior del lenguaje ActionScript son `trace()` y `setInterval()`.

**Funciones definidas por el usuario**: son aquellas que crea usted mismo para usarlas en aplicaciones, a diferencia de las funciones en las clases incorporadas, que realizan funciones predefinidas. Deberá asignar nombre a las funciones usted mismo y añadir sentencias en el bloque de función.

**Variable**: identificador que almacena valores de cualquier tipo de datos. Las variables pueden crearse, modificarse y actualizarse. Los valores almacenados en una variable pueden recuperarse para ser utilizados en scripts. En el siguiente ejemplo, los identificadores situados a la izquierda de los signos igual son variables:

```
var x:Number = 5;
var name:String = "Lolo";
var c_color:Color = new Color(mcinstanceName);
```

Para más información sobre el uso de variables, consulte "Variables" en la página 343.

**Gráficos vectoriales**: representan imágenes mediante líneas y curvas, denominadas vectores, que también incluyen propiedades de color y posición. Cada vector utiliza cálculos matemáticos, en lugar de bits, para describir la forma, lo que les permite ajustar la escala sin degradar la calidad de la imagen. Otro tipo de gráfico es el *mapa de bits*, que se representa mediante puntos o píxeles.

# Índice alfabético

## Símbolos



## Números



## A

## B



## C

## D



## E

## F

## G



## H



## I

## J



## L

## M

## N



## O

## P

## R



## S

## T

## U



## V

## W



## X