

MX



macromedia®

FLASH™MX
2004

Utilización de componentes

Marcas comerciales

Add Life to the Web, Afterburner, Aftershock, Andromedia, Allaire, Animation PowerPack, Aria, Attain, Authorware, Authorware Star, Backstage, Bright Tiger, Clustercats, ColdFusion, Contribute, Design In Motion, Director, Dream Templates, Dreamweaver, Drumbeat 2000, EDJE, EJIPT, Extreme 3D, Fireworks, Flash, Fontographer, FreeHand, Generator, HomeSite, JFusion, JRun, Kawa, Know Your Site, Knowledge Objects, Knowledge Stream, Knowledge Track, LikeMinds, Lingo, Live Effects, el logotipo y el diseño de MacRecorder, Macromedia, Macromedia Action!, Macromedia Flash, el logotipo y el diseño de Macromedia M, Macromedia Spectra, el logotipo y el diseño de Macromedia xRes, MacroModel, Made with Macromedia, el logotipo y el diseño de Made with Macromedia, el logotipo y el diseño de MAGIC, Mediamaker, Movie Critic, Open Sesame!, Roundtrip, Roundtrip HTML, Shockwave, Sitespring, SoundEdit, Titlemaker, UltraDev, Web Design 101, what the web can be y Xtra son marcas registradas o marcas comerciales de Macromedia, Inc. y pueden estar registradas en Estados Unidos o en otras jurisdicciones, incluidas las internacionales. Otros nombres de productos, logotipos, diseños, títulos, palabras o frases mencionados en esta publicación pueden ser marcas comerciales, marcas de servicio o nombres registrados de Macromedia, Inc. o de otras entidades y pueden estar registrados en ciertas jurisdicciones, incluidas las internacionales.

Información de terceros

Esta guía contiene vínculos a sitios Web de terceros que no están bajo el control de Macromedia y, por consiguiente, Macromedia no se hace responsable del contenido de dichos sitios Web. El acceso a uno de los sitios Web de terceros mencionados en esta guía será a cuenta y riesgo del usuario. Macromedia proporciona estos vínculos únicamente como ayuda y su inclusión no implica que Macromedia se haga responsable del contenido de dichos sitios Web.

La tecnología de compresión y descompresión de voz tiene licencia de Nellymoser, Inc. (www.nellymoser.com).



La tecnología de compresión y descompresión de vídeo Sorenson™ Spark™ tiene licencia de Sorenson Media, Inc.

Navegador Opera® Copyright © 1995-2002 Opera Software ASA y sus proveedores. Todos los derechos reservados.

Limitación de garantías de Apple

APPLE COMPUTER, INC. NO GARANTIZA, DE FORMA EXPRESA NI IMPLÍCITA, LA COMERCIABILIDAD O IDONEIDAD PARA UN FIN DETERMINADO DEL PAQUETE DE SOFTWARE INFORMÁTICO INCLUIDO. LA EXCLUSIÓN DE GARANTÍAS IMPLÍCITAS NO ESTÁ PERMITIDA EN ALGUNOS ESTADOS. LA RESTRICCIÓN ANTERIOR PUEDE NO AFECTARLE. ESTA GARANTÍA LE PROPORCIONA DERECHOS LEGALES ESPECÍFICOS. PUEDE TENER OTROS DERECHOS QUE VARÍAN SEGÚN LA LEGISLACIÓN LOCAL.

Copyright © 2003 Macromedia, Inc. Todos los derechos reservados. No se permite la copia, fotocopia, reproducción, traducción ni la conversión en formato electrónico o legible por equipos, ya sea de forma total o parcial de este manual, sin la autorización previa por escrito de Macromedia, Inc. Número de referencia ZFL70M500SP

Agradecimientos

Director: Erick Vera

Dirección del proyecto: Stephanie Gowin, Barbara Nelson

Redacción: Jody Bleye, Mary Burger, Kim Diezel, Stephanie Gowin, Dan Harris, Barbara Herbert, Barbara Nelson, Shirley Ong, Tim Statler

Directora de edición: Rosana Francescato

Edición: Mary Ferguson, Mary Kraemer, Noreen Maher, Antonio Padial, Lisa Stanziano, Anne Szabla

Administración de la producción: Patrice O'Neill

Producción y diseño multimedia: Adam Barnett, Christopher Basmajian, Aaron Begley, John Francis, Jeff Harmon

Localización: Tim Hussey, Seungmin Lee, Masayo Noda, Simone Pux, Yuko Yagi, Jorge G. Villanueva

Primera edición: agosto de 2003

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

CONTENIDO

INTRODUCCIÓN: Primeros pasos con los componentes.	7
Audiencia	7
Requisitos del sistema	8
Instalación de componentes	8
Documentación.	9
Convenciones tipográficas	10
Términos utilizados en este manual	10
Recursos adicionales	10
 CAPÍTULO 1: Los componentes	11
Ventajas de los componentes de la v2.	11
Categorías de componentes	12
Arquitectura de componentes	12
Novedades en los componentes de la v2	13
Clips compilados y archivos SWC	14
Accesibilidad y componentes	14
 CAPÍTULO 2: Trabajo con componentes	15
Panel Componentes	15
Componentes del panel Biblioteca	16
Componentes del panel Inspector de componentes y del inspector de propiedades	16
Componentes en vista previa dinámica	17
Trabajo con archivos SWC y clips compilados	18
Adición de componentes a documentos de Flash	18
Definición de parámetros de componentes	21
Eliminación de componentes de documentos de Flash	21
Utilización de las sugerencias para el código	22
Eventos de componentes	22
Creación de un desplazamiento personalizado de la selección	24
Administración de la profundidad del componente en un documento	25
Utilización de preloader con componentes	25
Actualización de los componentes de la v1 a la arquitectura de la v2	26

CAPÍTULO 3: Personalización de componentes	27
Utilización de estilos para personalizar el texto y el color de un componente	27
Temas	35
Aplicación de aspectos a los componentes	37
 CAPÍTULO 4: Diccionario de componentes	 45
Componentes de la interfaz de usuario (IU)	45
Componentes de medios	46
Componentes de datos	47
Administradores	47
Pantallas	47
Componente Accordion	47
Componente Alert.	48
Componente Button	48
Interfaz CellRenderer	59
Componente CheckBox	59
Componente ComboBox	67
Paquete DataBinding	95
Componente DataGrid	95
Componente DataHolder	95
Componente DataProvider	95
Componente DataSet	95
Componente DateChooser	95
Componente DateField	95
Clase DepthManager.	95
Clase FocusManager	101
Clase Form	107
Componente Label	107
Componente List	112
Componente Loader	141
Componente MediaController	151
Componente MediaDisplay	151
Componente MediaPlayer	151
Componente Menu	151
Componente MenuBar	151
Componente NumericStepper	151
Clase PopUpManager	160
Componente ProgressBar	162
Componente RadioButton	176
Componente RDBMSResolver	186
Interfaz de RemoteProcedureCall	186
Clase Screen	186
Componente ScrollPane	186
Clase StyleManager	202
Clase Slide.	204
Componente TextArea.	204
Componente TextInput	216
Componente Tree	227
Clase UIComponent.	227

Clase UIEventDispatcher	234
Clase UIObject	236
Paquete WebServices	253
Componente WebServiceConnector	254
Componente Window	254
Componente XMLConnector	264
Componente XUpdateResolver	264
CAPÍTULO 5: Creación de componentes	265
Novedades	265
Trabajo con el entorno de Flash	265
Creación de componentes	268
Escritura de ActionScript del componente	270
Importación de clases	272
Selección de una clase principal	272
Escritura del constructor	274
Control de versiones	274
Nombres de clase, símbolo y propietario	275
Definición de captadores y definidores	275
Metadatos de componente	276
Definición de parámetros de componentes	283
Implementación de métodos principales	283
Gestión de eventos	284
Aplicación de aspectos	288
Adición de estilos	289
Cómo facilitar el acceso a los componentes	289
Exportación del componente	290
Cómo facilitar la utilización del componente	292
Recomendaciones para el diseño de un componente	292
ÍNDICE ALFABÉTICO	295

INTRODUCCIÓN

Primeros pasos con los componentes

Macromedia Flash MX 2004 y Macromedia Flash MX Professional 2004 son las herramientas estándar de edición profesional para la creación de publicaciones Web de gran impacto. Los componentes son bloques de creación para aplicaciones de Internet ricas en contenido que proporcionan dicho impacto. Un componente es un clip de película con parámetros que se definen durante la edición en Macromedia Flash. El código ActionScript es una interfaz API que permite a los desarrolladores volver a utilizar y compartir código, así como encapsular complejas funciones que los diseñadores pueden utilizar y personalizar sin necesidad de utilizar ActionScript.

Los componentes están integrados en la versión 2 (v2) de la arquitectura de componentes de Macromedia, lo que permite crear con facilidad y rapidez aplicaciones sólidas con apariencia y comportamiento uniformes. En este manual se explica la forma de crear aplicaciones con componentes de la v2 y se describen las interfaces API (interfaz de programación de aplicaciones) de los componentes. Incluye ejemplos de escenarios y procedimientos que utilizar con los componentes de la v2 de Flash MX 2004 o Flash MX Professional 2004, así como descripciones de las interfaces API de los componentes, en orden alfabético.

Puede utilizar los componentes creados por Macromedia, descargar componentes creados por otros desarrolladores o crear los suyos propios.

Audiencia

Este manual está dirigido a desarrolladores que crean aplicaciones de Flash MX 2004 o Flash MX Professional 2004 y desean utilizar componentes para agilizar el desarrollo. Deberá estar familiarizado con el desarrollo de aplicaciones en Macromedia Flash, con el código ActionScript y con Macromedia Flash Player.

En el manual se presupone que el usuario ya ha instalado Flash MX 2004 o Flash MX Professional 2004 y sabe cómo utilizarlo. Antes de utilizar los componentes, debe completar la lección “Crear una interfaz de usuario con componentes” (seleccione Ayuda > Cómo > Inicio rápido > Crear una interfaz de usuario con componentes).

Si desea escribir el menor código ActionScript posible, puede arrastrar los componentes a un documento, establecer sus parámetros en el inspector de propiedades o en el panel Inspector de componentes y adjuntar un controlador `on()` directamente a un componente en el panel Acciones para controlar los eventos de componentes.

Si es un programador que desea crear aplicaciones más sólidas, puede crear componentes de forma dinámica, utilizar sus interfaces API para establecer propiedades y llamar a métodos en tiempo de ejecución, así como utilizar el modelo de eventos detector para controlar los eventos.

Para más información, consulte el [Capítulo 2, “Trabajo con componentes”](#), en la [página 15](#).

Requisitos del sistema

Los componentes de Macromedia no tienen ningún requisito del sistema adicional a los de Flash MX 2004 o Flash MX Professional 2004.

Instalación de componentes

Al iniciar por primera vez Flash MX 2004 o Flash MX Professional 2004, ya hay un conjunto de componentes de Macromedia instalado. Pueden verse en el panel Componentes.

Flash MX 2004 incluye los componentes siguientes:

- [Componente Button](#)
- [Componente CheckBox](#)
- [Componente ComboBox](#)
- [Componente Label](#)
- [Componente List](#)
- [Componente Loader](#)
- [Componente NumericStepper](#)
- [Clase PopUpManager](#)
- [Componente ProgressBar](#)
- [Componente RadioButton](#)
- [Componente ScrollPane](#)
- [Componente TextArea](#)
- [Componente TextInput](#)
- [Componente Window](#)

Flash MX Professional 2004 incluye los componentes de Flash MX 2004 y los siguientes componentes y clases adicionales:

- [Componente Accordion](#)
- [Componente Alert](#)
- [Paquete DataBinding](#)
- [Componente DateField](#)
- [Componente DataGrid](#)
- [Componente DataHolder](#)
- [Componente DataSet](#)
- [Componente DateChooser](#)
- [Clase Form](#)
- [Componente MediaController](#)

- [Componente MediaPlayer](#)
- [Componente MediaPlayer](#)
- [Componente Menu](#)
- [Componente RDBMSResolver](#)
- [Clase Screen](#)
- [Componente Tree](#)
- [Componente WebServiceConnector](#)
- [Componente XMLConnector](#)
- [Componente XUpdateResolver](#)

Para verificar la instalación de los componentes de Flash MX 2004 o Flash MX Professional 2004:

- 1 Inicie Flash.
- 2 Seleccione Ventana > Paneles de desarrollo > Componentes para abrir el panel Componentes, si aún no está abierto.
- 3 Seleccione UI Components para expandir el árbol y ver los componentes instalados.

También puede descargar componentes de [Macromedia Exchange](#). Para instalar componentes descargados de Exchange, descargue e instale [Macromedia Extensions Manager](#).

Todos los componentes, ya se trate de un archivo SWC o de un archivo FLA (véase “[Clips compilados y archivos SWC](#)” en la página 14), pueden aparecer en el panel Componentes de Flash. Realice estos pasos para instalar componentes en un equipo con Windows o Macintosh.

Para instalar componentes en un equipo con Windows o Macintosh:

- 1 Salga de Flash.
- 2 Coloque el archivo SWC o FLA que contiene el componente en la siguiente carpeta del disco duro:
 - Disco duro/Applications/Macromedia Flash MX 2004/First Run/Components (Macintosh)
 - \Archivos de programa\Macromedia\Flash MX 2004\<idioma>\First Run\Components (Windows)
- 3 Abra Flash.
- 4 Seleccione Ventana > Paneles de desarrollo > Componentes para ver los componentes del panel Componentes, si aún no está abierto.

Documentación

En este documento se explican los detalles de la utilización de componentes para desarrollar aplicaciones de Flash. Se presupone que el lector tiene conocimientos generales de Macromedia Flash y ActionScript. Hay disponible por separado documentación específica sobre Flash y productos relacionados.

- Para obtener información sobre Macromedia Flash, consulte los apartados *Utilización de Flash*, *Guía de referencia de ActionScript* y *Diccionario de ActionScript* de la Ayuda.
- Para obtener información sobre el acceso a los servicios Web con aplicaciones de Flash, consulte *Utilización de convirtiendo a remoto de Flash*.

Convenciones tipográficas

En este manual se utilizan las siguientes convenciones tipográficas:

- *Fuente en cursiva* indica un valor que se debe sustituir (por ejemplo, en una ruta de carpeta).
- Fuente para código indica código ActionScript.
- *Fuente para código en cursiva* indica un parámetro de ActionScript.
- **Fuente en negrita** indica una entrada de caracteres.

Nota: la fuente negrita no es la misma que la fuente utilizada en los encabezados run-in, es decir, los encabezados que aparecen en la misma línea que el texto al que introducen. La fuente de los encabezados run-in se utiliza como una alternativa a las viñetas.

Términos utilizados en este manual

En este manual se utilizan los términos siguientes:

en tiempo de ejecución Cuando el código se ejecuta en Flash Player.

durante la edición Mientras se trabaja en el entorno de edición de Flash.

Recursos adicionales

Para obtener información reciente sobre Flash, además de las sugerencias de usuarios expertos, temas avanzados, ejemplos, consejos y otras actualizaciones, consulte el sitio Web [Macromedia DevNet](#), que se actualiza con regularidad. Visite el sitio Web regularmente para conocer las últimas noticias sobre Flash y cómo sacar el máximo partido del programa.

Para obtener notas técnicas, actualizaciones de la documentación y vínculos a recursos adicionales de la comunidad de Flash, consulte el [Centro de soporte de Macromedia Flash](#) en http://www.macromedia.com/go/flash_support_sp.

Para obtener información detallada sobre los términos, la sintaxis y la utilización de ActionScript, consulte los apartados *Guía de referencia de ActionScript* y *Diccionario de ActionScript* de la Ayuda.

CAPÍTULO 1

Los componentes

Los componentes son clips de película con parámetros que permiten modificar la apariencia y el comportamiento. Un componente puede proporcionar cualquier función que su creador pueda imaginar. Puede ser desde un sencillo control de interfaz de usuario como un botón de opción o una casilla de verificación, hasta un elemento con contenido, como un panel de desplazamiento; por otra parte, un componente también puede ser un elemento que no se ve, como FocusManager, que permite controlar qué objeto pasa a estar seleccionado en una aplicación.

Los componentes permiten a cualquier usuario crear complejas aplicaciones de Macromedia Flash MX 2004 y Macromedia Flash MX Professional 2004, aunque no tenga conocimientos avanzados de ActionScript. En lugar de crear botones personalizados, cuadros combinados y listas, basta con arrastrar dichos componentes desde el panel Componentes para añadir funciones a las aplicaciones. Asimismo, la apariencia de los componentes se puede personalizar fácilmente y conseguir, de esta manera, una mayor adaptación a las necesidades de diseño propias.

Los componentes están integrados en la versión 2 (v2) de la arquitectura de componentes de Macromedia, lo que permite crear con facilidad y rapidez aplicaciones sólidas con apariencia y comportamiento uniformes. La arquitectura de la v2 incluye clases en las que se basan todos los componentes, mecanismos de estilo y de aspecto para personalizar fácilmente la apariencia de los componentes, un modelo de eventos difusor/detector, gestión de profundidad y selección, implementación de accesibilidad, etc.

Cada componente dispone de parámetros predefinidos que se pueden establecer mientras se edita en Flash. Asimismo, cada uno dispone de un conjunto único de métodos, propiedades y eventos de ActionScript, llamado también *API* (interfaz de programación de aplicaciones), que permite definir parámetros y opciones adicionales en tiempo de ejecución.

Flash MX 2004 y Flash MX Professional 2004 incluyen muchos componentes de Flash nuevos y varias versiones nuevas de componentes que se incluyeron en Flash MX. Para obtener una lista completa de los componentes que se incluyen con Flash MX 2004 y Flash MX Professional 2004, consulte [“Instalación de componentes” en la página 8](#). También puede descargar componentes creados por miembros de la comunidad de Flash en [Macromedia Exchange](#).

Ventajas de los componentes de la v2

Los componentes permiten establecer la separación entre codificación y diseño. Asimismo, permiten volver a utilizar código, ya sea en componentes que se creen, o bien descargando e instalando componentes creados por otros desarrolladores.

Los componentes permiten a los programadores crear funciones que los diseñadores pueden utilizar en aplicaciones. Los desarrolladores pueden encapsular en componentes las funciones utilizadas con mayor frecuencia, y los diseñadores pueden personalizar el aspecto y el comportamiento de los componentes cambiando los parámetros en el inspector de propiedades o en el panel Inspector de componentes.

Los miembros de la comunidad de Flash pueden utilizar [Macromedia Exchange](#) para intercambiar componentes. Si utiliza componentes, ya no tendrá que crear cada uno de los elementos de una aplicación Web compleja desde cero. Simplemente, busque los componentes que necesite y póngalos en un documento de Flash para crear una aplicación nueva.

Los componentes que se basan en la arquitectura de componentes de la v2 comparten una funcionalidad básica, como estilos, gestión de eventos, aplicación de aspectos, gestión de selección y de profundidad. Cuando se añade el primer componente de la v2 a una aplicación, se añaden alrededor de 25 K al documento, que incluyen dicha funcionalidad básica. Si añade más componentes, esos mismos 25 K se volverán a utilizar también para ellos, por lo que el aumento de tamaño del documento será inferior al esperado. Para más información sobre cómo actualizar los componentes de la v1 a la v2, consulte [“Actualización de los componentes de la v1 a la arquitectura de la v2” en la página 26](#).

Categorías de componentes

Los componentes incluidos en Flash MX 2004 y Flash MX Professional 2004 se dividen en cuatro categorías: componentes de interfaz de usuario (IU), componentes multimedia, componentes de datos y administradores. Los controles de la interfaz de usuario permiten al usuario interactuar con la aplicación; por ejemplo, los componentes RadioButton, CheckBox y TextInput son controles de la IU. Los componentes multimedia le permiten reproducir imágenes y sonido directamente en una aplicación; el componente MediaPlayer es un componente multimedia. Los componentes de datos permiten cargar y manipular la información de las fuentes de datos; los componentes WebServiceConnector y XMLConnector son componentes de datos. Los administradores son componentes que no se ven y que permiten gestionar una función (por ejemplo, selección o profundidad) en una aplicación; FocusManager, DepthManager, PopUpManager y StyleManager son los componentes de administrador que se incluyen con Flash MX 2004 y Flash MX Professional 2004. Para obtener una lista completa de cada categoría, consulte el [Capítulo 4, “Diccionario de componentes”, en la página 45](#).

Arquitectura de componentes

Puede utilizar el inspector de propiedades o el panel Inspector de componentes para cambiar los parámetros del componente y utilizar las funciones básicas de los componentes. No obstante, si desea tener un mayor control sobre los componentes, tendrá que utilizar las interfaces API de los mismos y poseer conocimientos sobre cómo se crearon.

Los componentes de Flash MX 2004 y Flash MX Professional 2004 se crean utilizando la versión 2 (v2) de la arquitectura de componentes de Macromedia. Flash Player 6 y Flash Player 7 admiten los componentes de la versión 2. Dichos componentes no siempre son compatibles con componentes creados mediante la arquitectura de la versión 1 (v1) (todos los componentes anteriores a Flash MX 2004). Asimismo, Flash Player 7 no admite los componentes de la versión 1. Para más información, consulte [“Actualización de los componentes de la v1 a la arquitectura de la v2” en la página 26](#).

Los componentes de la v2 están incluidos en el panel Componentes como símbolos de clip compilado (SWC). Un clip compilado es un clip de película de componente cuyo código se ha compilado. Los clips compilados tienen vistas previas dinámicas incorporadas y no se pueden editar, aunque se pueden cambiar los parámetros en el inspector de propiedades o en el panel Inspector de componentes, al igual que con cualquier otro componente. Para más información, consulte [“Clips compilados y archivos SWC” en la página 14](#).

Los componentes de la v2 se escriben en ActionScript 2. Cada componente es una clase y cada clase es un paquete de ActionScript. Por ejemplo, un componente de botón de opción es una instancia de la clase `RadioButton` cuyo nombre de paquete es `mx.controls`. Para más información sobre paquetes, consulte [“Utilización de paquetes” en el apartado Guía de referencia de ActionScript de la Ayuda](#).

Todos los componentes creados con la versión 2 de la arquitectura de componentes de Macromedia son subclases de las clases `UIObject` y `UIComponent`, y heredan todos los métodos, eventos y propiedades de dichas clases. Asimismo, muchos componentes son subclases de otros componentes. La ruta de herencia de cada componente se muestra en el [Capítulo 4, “Diccionario de componentes”, en la página 45](#).

Todos los componentes utilizan también el mismo modelo de evento, los mismos estilos basados en CSS y el mismo mecanismo de aplicación de aspectos incorporados. Para más información sobre los estilos y la aplicación de aspectos, consulte el [Capítulo 3, “Personalización de componentes”, en la página 27](#). Para más información sobre la gestión de eventos, consulte el [Capítulo 2, “Trabajo con componentes”, en la página 15](#).

Novedades en los componentes de la v2

El **panel Inspector de componentes** le permite cambiar los parámetros de los componentes al mismo tiempo que edita en Macromedia Flash y en Macromedia Dreamweaver. (Véase [“Componentes del panel Inspector de componentes y del inspector de propiedades” en la página 16](#).)

Modelo de eventos del detector permite a los objetos detectores de funciones gestionar eventos. (Véase [“Eventos de componentes” en la página 22](#).)

Las **propiedades de aspecto** permiten cargar estados sólo cuando es necesario. (Véase [“Aplicación de aspectos a los componentes” en la página 37](#).)

Estilos basados en CSS permite crear una apariencia uniforme en todas las aplicaciones. (Véase [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 27](#).)

Temas permite arrastrar una apariencia nueva a un conjunto de componentes. (Véase [“Temas” en la página 35](#).)

Tema Halo proporciona a las aplicaciones una interfaz de usuario lista, útil y flexible.

Clases de administrador proporciona una manera fácil de gestionar la selección y profundidad de una aplicación. (Véase [“Creación de un desplazamiento personalizado de la selección” en la página 24](#) y [“Administración de la profundidad del componente en un documento” en la página 25](#).)

Las **clases básicas `UIObject` y `UIComponent`** proporcionan funciones básicas a todos los componentes. (Véase [“Clase `UIComponent`” en la página 227](#) y [“Clase `UIObject`” en la página 236](#).)

Empaquetado como archivo SWC permite una fácil distribución y ocultación de código. Véase [“Creación de componentes”](#). Es posible que tenga que descargar la versión más actualizada del PDF del Centro de soporte de Flash para consultar esta información.

Vinculación de datos incorporados está disponible mediante el panel Inspector de componentes. Para más información sobre esta función, presione el botón de actualización de la Ayuda.

Jerarquía de clases ampliable fácilmente mediante ActionScript 2 le permite crear namespaces exclusivos, importar clases cuando sea necesario y poner fácilmente en subclases para ampliar componentes. Véase [“Creación de componentes”](#) y la *guía de referencia de ActionScript*. Es posible que tenga que descargar la versión más actualizada del PDF del Centro de soporte de Flash para consultar esta información.

Clips compilados y archivos SWC

Los clips compilados se utilizan para compilar previamente símbolos complejos de un documento de Flash. Por ejemplo, un clip de película con gran cantidad de código ActionScript que no cambie a menudo puede convertirse en un clip compilado. Como resultado, tanto el comando Probar película como Publicar necesitarían menos tiempo para ejecutarse.

Un archivo SWC es el tipo de archivo utilizado para guardar y distribuir componentes. Si coloca un archivo SWC en la carpeta First Run\Components, el componente aparecerá en el panel Componentes. Si añade un componente al escenario desde el panel Componentes, se añadirá un símbolo de clip compilado a la biblioteca.

Para más información sobre archivos SWC, consulte [“Creación de componentes”](#). Es posible que tenga que descargar la versión más actualizada del PDF del Centro de soporte de Flash para consultar esta información.

Accesibilidad y componentes

Cada vez resulta más importante que el contenido Web sea accesible, es decir, que lo puedan utilizar usuarios con diferentes discapacidades. Los invidentes pueden acceder al contenido visual de las aplicaciones de Flash mediante la utilización de un software de lectura de pantalla que proporciona una descripción hablada del contenido de la pantalla.

Cuando se crea un componente, el autor puede escribir código ActionScript que permite la comunicación entre el componente y el lector de pantalla. A continuación, cuando el desarrollador utiliza componentes para crear una aplicación en Flash, recurre al panel Accesibilidad para configurar la instancia de cada componente.

La mayoría de los componentes creados por Macromedia están diseñados teniendo en cuenta la accesibilidad. Para saber si un componente es accesible, consulte su entrada en el [Capítulo 4, “Diccionario de componentes”](#), en la [página 45](#). Cuando cree una aplicación en Flash, tendrá que añadir una línea de código por cada componente

`(mx.accessibility.ComponentNameAccImpl.enableAccessibility());` y definir los parámetros de accesibilidad en el panel Accesibilidad. La accesibilidad de los componentes funciona igual que para todos los clips de película de Flash. Para más información, consulte [“Creación de contenido accesible”](#) en el apartado Utilización de Flash de la Ayuda. Es posible que tenga que actualizar el sistema de Ayuda para consultar esta información.

Existe la posibilidad de navegar con el teclado por la mayoría de los componentes creados por Macromedia. Cada entrada del componente del [Capítulo 4, “Diccionario de componentes”](#), en la [página 45](#) indica si puede controlar o no el componente con el teclado.

CAPÍTULO 2

Trabajo con componentes

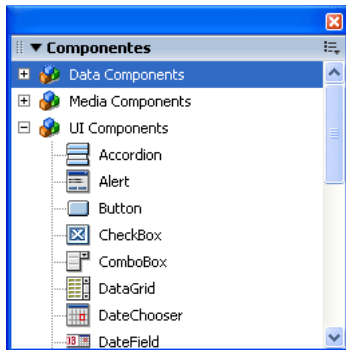
Existen varias formas de trabajar con componentes en Macromedia Flash MX 2004 y Macromedia Flash MX Professional 2004. El panel Componentes sirve para ver componentes y añadirlos a un documento durante su edición. Una vez añadido el componente al documento, las propiedades de éste pueden verse en el inspector de propiedades o en el panel Inspector de componentes. Los componentes pueden comunicarse entre sí escuchando sus eventos y gestionándolos mediante ActionScript. También puede gestionar la profundidad del componente en un documento y controlar cuándo se selecciona el componente.

Panel Componentes

En el panel Componentes se almacenan todos los componentes. Cuando se instala Flash MX 2004 o Flash MX Professional 2004 y se inicia por primera vez, los componentes de la carpeta Macromedia Flash 2004/First Run/Components (Macintosh) o Macromedia\Flash 2004\<idioma>\First Run\Components (Windows) aparecen en el panel Componentes.

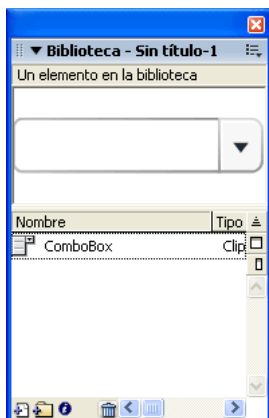
Para mostrar el panel Componentes:

- Seleccione Ventana > Paneles de desarrollo > Componentes.



Componentes del panel Biblioteca

Cuando se añade un componente a un documento, el componente aparece como un símbolo de clip compilado (SWC) en el panel Biblioteca.



Componente ComboBox del panel Biblioteca.

Puede agregar más instancias de un componente arrastrando el icono del componente desde la biblioteca al escenario.

Para más información sobre clips compilados, consulte [“Trabajo con archivos SWC y clips compilados” en la página 18](#).

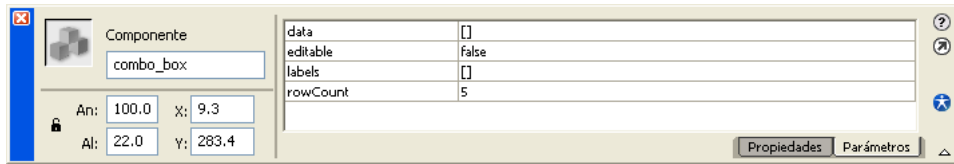
Componentes del panel Inspector de componentes y del inspector de propiedades

Una vez que haya añadido una instancia de un componente a un documento de Flash, utilice el inspector de propiedades para definir y ver información de la instancia. Para crear una instancia de un componente, arrástrelo del panel Componentes al escenario; asigne un nombre a la instancia en el inspector de propiedades y, a continuación, especifique los parámetros de la instancia mediante los campos de la ficha Parámetros. También puede definir parámetros para una instancia de componente mediante el panel Inspector de componentes. No importa qué panel utiliza para definir los parámetros; es sólo cuestión de preferencias personales. Para más información sobre cómo definir parámetros, consulte [“Definición de parámetros de componentes” en la página 21](#).

Para ver información sobre una instancia de componente en el inspector de propiedades:

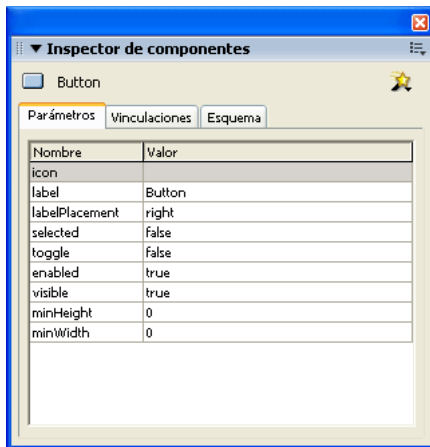
- 1 Seleccione Ventana > Propiedades.
- 2 Seleccione una instancia de componente en el escenario.

- 3 Para ver los parámetros, haga clic en la ficha Parámetros.



Para ver los parámetros de una instancia de componente en el panel Inspector de componentes:

- 1 Seleccione Ventana > Inspector de componentes.
- 2 Seleccione una instancia de componente en el escenario.
- 3 Para ver los parámetros, haga clic en la ficha Parámetros.

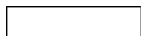


Componentes en vista previa dinámica

La función Vista previa dinámica, que se activa de forma predeterminada, permite ver los componentes del escenario tal como aparecerán en el contenido de Flash publicado, incluido el tamaño aproximado. La vista previa dinámica refleja parámetros diferentes para componentes diferentes. Para obtener información sobre qué parámetros de componente se reflejan en la vista previa dinámica, consulte la entrada de cada componente en el [Capítulo 4, “Diccionario de componentes”, en la página 45](#). Los componentes en vista previa dinámica no son funcionales. Para probar la funcionalidad de los componentes, utilice el comando Control > Probar película.



Componente Button con la vista previa dinámica activada.



Componente Button con la vista previa dinámica desactivada.

Para activar o desactivar la función Vista previa dinámica:

- Seleccione Control > Activar vista previa dinámica. Si aparece una marca de verificación junto a la opción, ésta está activada.

Para más información, consulte “[Creación de componentes](#)”. Es posible que tenga que descargar la versión más actualizada del PDF del Centro de soporte de Flash para consultar esta información.

Trabajo con archivos SWC y clips compilados

Los componentes que se incluyen con Flash MX 2004 o Flash MX Professional 2004 no son archivos FLA, sino archivos SWC. SWC es el formato de archivo de Macromedia para componentes. Si añade un componente al escenario desde el panel Componentes, se añadirá un símbolo de clip compilado a la biblioteca. Un SWC es un clip compilado que se ha exportado para su distribución.

Un clip de película también se puede “compilar” en Flash y convertir en un símbolo de clip compilado. El símbolo de clip compilado se comporta como el símbolo de clip de película desde el que se compiló, aunque los clips compilados se muestran y se publican más rápido que los símbolos de clip de película normales. Los clips compilados no se pueden editar, aunque tienen propiedades que aparecen en el inspector de propiedades y en el panel Inspector de componentes e incluyen una vista previa dinámica.

Los componentes que se incluyen con Flash MX 2004 o Flash MX Professional 2004 ya están convertidos en clips compilados. Si crea un componente, puede exportarlo como SWC para su distribución. Para más información, consulte “[Creación de componentes](#)”. Es posible que tenga que descargar la versión más actualizada del PDF del Centro de soporte de Flash para consultar esta información.

Para compilar un símbolo de clip de película:

- Seleccione el clip de película en la biblioteca, haga clic con el botón derecho del ratón (Windows) o con la tecla Control presionada (Macintosh) y seleccione Convertir en clip compilado.

Para exportar un archivo SWC:

- Seleccione el clip de película en la biblioteca, haga clic con el botón derecho del ratón (Windows) o con la tecla Control presionada (Macintosh) y seleccione Exportar archivo SWC.

Nota: Flash MX 2004 y Flash MX Professional 2004 siguen admitiendo los componentes FLA.

Adición de componentes a documentos de Flash

Si arrastra un componente desde el panel Componentes al escenario, se añadirá un símbolo de clip compilado en el panel Biblioteca. Una vez que el símbolo de clip compilado está en la biblioteca, también puede añadir el componente a un documento en tiempo de ejecución mediante el método `UIObject.createClassObject()` de ActionScript.

- Los usuarios que empiezan a utilizar Flash pueden usar el panel Componentes para añadir componentes a documentos de Flash, especificar parámetros básicos mediante el inspector de propiedades o el panel Parámetros de componentes y utilizar el controlador de eventos `on()` para controlar los componentes.

- Los usuarios de nivel intermedio pueden utilizar el panel Componentes para añadir componentes a documentos de Flash y, a continuación, utilizar el inspector de propiedades, los métodos ActionScript o ambos para especificar parámetros. También pueden utilizar el controlador de eventos `on()` o los detectores de eventos para gestionar los eventos del componente.
- Los programadores de Flash de nivel avanzado pueden utilizar una combinación del panel Componentes y de ActionScript para añadir componentes y especificar propiedades, o bien pueden implementar instancias de componente en tiempo de ejecución usando sólo ActionScript. También pueden utilizar detectores de eventos para controlar los componentes.

Si edita los aspectos de un componente y, a continuación, añade otra versión del componente o un componente que comparte los mismos aspectos, puede utilizar los aspectos editados o reemplazarlos por un nuevo conjunto de aspectos predeterminados. Si reemplaza los aspectos editados, todos los componentes que utilizan estos aspectos se actualizan con versiones predeterminadas de los aspectos. Para más información sobre cómo editar los aspectos, consulte el [Capítulo 3, “Personalización de componentes”, en la página 27](#).

Adición de componentes mediante el panel Componentes

Una vez que haya añadido un componente a un documento mediante el panel Componentes, puede añadir instancias adicionales del componente al documento arrastrando el componente desde el panel Biblioteca al escenario. Puede definir propiedades para instancias adicionales en la ficha Parámetros del inspector de propiedades o en el panel Parámetros de componentes.

Para añadir un componente a un documento de Flash mediante el panel Componentes:

- 1 Seleccione Ventana > Componentes.
- 2 Siga uno de estos procedimientos:
 - Arrastre un componente del panel Componentes al escenario.
 - Haga doble clic en un componente en el panel Componentes.
- 3 Si el componente es un FLA (todos los componentes instalados de la v2 son SWC) y si ha editado aspectos para otra instancia del mismo componente, o para un componente que comparte aspectos con el componente que se está añadiendo, siga uno de estos procedimientos:
 - Seleccione No reemplazar elementos ya existentes para mantener los aspectos editados y aplicarlos al nuevo componente.
 - Seleccione Reemplazar elementos ya existentes para reemplazar todos los aspectos por aspectos predeterminados. El nuevo componente y todas sus versiones anteriores, o las de los componentes que comparten sus aspectos, utilizarán los aspectos predeterminados.
- 4 Seleccione el componente en el escenario.
- 5 Seleccione Ventana > Propiedades.
- 6 En el inspector de propiedades, introduzca un nombre para la instancia de componente.
- 7 Haga clic en la ficha Parámetros y especifique parámetros para la instancia.
Para más información, consulte [“Definición de parámetros de componentes” en la página 21](#).
- 8 Cambie el tamaño del componente como desee.

Para más información sobre cómo cambiar el tamaño de tipos de componente específicos, consulte las entradas referentes al componente en cuestión en el [Capítulo 4, “Diccionario de componentes”, en la página 45](#).

- 9 Cambie el color y el formato de texto de un componente como desee mediante uno o varios de estos procedimientos:
 - Defina o cambie un valor de propiedad de estilo específico para una instancia de componente mediante el método `setStyle()` disponible para todos los componentes. Para más información, consulte [UIObject.setStyle\(\)](#).
 - Edite varias propiedades en la declaración de estilo `_global` asignada a todos los componentes de la v2.
 - Si lo desea, cree una declaración de estilo personalizada para instancias de componente específicas.
Para más información, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 27](#).
- 10 Personalice la apariencia del componente como desee mediante uno de estos procedimientos:
 - Aplique un tema (véase [“Temas” en la página 35](#)).
 - Edite los aspectos de un componente (véase [“Aplicación de aspectos a los componentes” en la página 37](#)).

Adición de componentes mediante ActionScript

Para añadir un componente a un documento mediante ActionScript, primero debe añadirlo a la biblioteca.

Puede utilizar métodos ActionScript para definir otros parámetros para componentes añadidos dinámicamente. Para más información, consulte el [Capítulo 4, “Diccionario de componentes”](#), en la [página 45](#).

Nota: en las instrucciones de esta sección se presupone que el usuario posee un conocimiento intermedio o avanzado de ActionScript.

Para añadir un componente a un documento de Flash mediante ActionScript:

- 1 Arrastre un componente del panel Componentes al escenario y elimínelo.
De esta manera, añadirá el componente a la biblioteca.
- 2 Seleccione el fotograma en la línea de tiempo donde desea colocar el componente.
- 3 Abra el panel Acciones si aún no está abierto.
- 4 Llame al método `createClassObject()` para crear la instancia de componente en tiempo de ejecución.

Este método puede llamarse por sí solo, o bien a partir de una instancia de componente. Como parámetros tiene un nombre de clase de componente, un nombre de instancia para la instancia nueva, una profundidad y un objeto de inicialización opcional. En el parámetro `className`, puede especificar el paquete de clase tal como se indica a continuación:

```
createClassObject(mx.controls.CheckBox, "cb", 5, {label:"Check Me"});
```

O puede importar el paquete de clase tal como se indica a continuación:

```
import mx.controls.CheckBox;  
createClassObject(CheckBox, "cb", 5, {label:"Check Me"});
```

Para más información, consulte [UIObject.createClassObject\(\)](#).

- 5 Utilice las propiedades y métodos ActionScript del componente para especificar opciones adicionales o sustituir parámetros establecidos durante la edición.

Para obtener información detallada sobre las propiedades y métodos ActionScript disponibles para cada componente, consulte sus respectivas entradas en el [Capítulo 4, “Diccionario de componentes”, en la página 45](#).

Tamaño de la etiqueta del componente y anchura y altura del componente

Si una instancia de componente que se ha añadido a un documento no es lo suficientemente grande para mostrar la etiqueta, el texto de la etiqueta aparece recortado. Si una instancia de componente es más grande que el texto, la zona activa sobrepasa la etiqueta.

Utilice la herramienta Transformación libre o el método `setSize()` para cambiar el tamaño de las instancias de componente. Puede llamar al método `setSize()` desde cualquiera de las instancias de componente (véase `UIObject.setSize()`). Si utiliza las propiedades de ActionScript `_width` y `_height` para ajustar la anchura y altura de un componente, se cambia el tamaño del componente pero el diseño del contenido no varía. Esto puede provocar la distorsión del componente al reproducir la película. Para más información sobre el tamaño de los componentes, consulte sus respectivas entradas en el [Capítulo 4, “Diccionario de componentes”, en la página 45](#).

Definición de parámetros de componentes

Todos los componentes tienen parámetros que se pueden definir para cambiar la apariencia y el comportamiento. Un parámetro es una propiedad o método que aparece en el inspector de propiedades o en el panel Inspector de componentes. Las propiedades y métodos que se utilizan con más frecuencia aparecen como parámetros de edición; los demás deben definirse mediante ActionScript. Todos los parámetros que se pueden definir editando, también se pueden definir mediante ActionScript. Los valores de los parámetros definidos mediante ActionScript sustituyen cualquier otro valor que se haya definido durante la edición.

Todos los componentes de la v2 heredan las propiedades y los métodos de la clase `UIObject` y la clase `UIComponent`; se trata de propiedades y métodos que utilizan todos los componentes, como por ejemplo, `UIObject.setSize()`, `UIObject.setStyle()`, `UIObject.x` y `UIObject.y`. Asimismo, cada componente tiene propiedades y métodos exclusivos, algunos de los cuales están disponibles como parámetros de edición. Por ejemplo, el componente `ProgressBar` tiene una propiedad `percentComplete` (`ProgressBar.percentComplete`), mientras que el componente `NumericStepper` dispone de las propiedades `nextValue` y `previousValue` (`NumericStepper.nextValue`, `NumericStepper.previousValue`).

Eliminación de componentes de documentos de Flash

Para eliminar instancias de un componente de un documento de Flash, debe borrar el componente de la biblioteca eliminando el icono del clip compilado.

Para eliminar un componente de un documento:

- 1 En el panel Biblioteca, seleccione el símbolo de clip compilado (SWC).
- 2 Haga clic en el botón Eliminar situado en la parte inferior del panel Biblioteca, o seleccione Eliminar en el menú de opciones del panel Biblioteca.
- 3 En el cuadro de diálogo Eliminar, haga clic en Eliminar para confirmar la operación.

Utilización de las sugerencias para el código

Cuando utiliza ActionScript2, puede escribir estrictamente una variable basada en una clase incorporada, incluidas las clases de componente. Si lo hace así, el editor ActionScript muestra sugerencias para el código de la variable. Por ejemplo, supongamos que escribe lo siguiente:

```
var myCheckBox:CheckBox  
myCheckBox.
```

En cuanto escriba el punto, Flash muestra una lista de métodos y propiedades disponibles para los componentes CheckBox, porque la variable se ha escrito como CheckBox. Para más información sobre la escritura de datos, consulte “Strict data typing” en el apartado Guía de referencia de ActionScript de la Ayuda. Para obtener información sobre cómo utilizar las sugerencias para el código cuando aparecen, consulte “Utilización de las sugerencias para el código” en el apartado Guía de referencia de ActionScript de la Ayuda.

Eventos de componentes

Todos los componentes tienen eventos que se difunden cuando el usuario interactúa con un componente o cuando ocurre algo significativo al componente. Para gestionar un evento, escriba código ActionScript que se ejecute cuando se activa el evento.

Los eventos de componentes pueden gestionarse de las siguientes maneras:

- Utilice el controlador de eventos de componentes `on()`.
- Utilice detectores de eventos.

Utilización de controladores de eventos de componentes

La forma más fácil de gestionar un evento de un componente es mediante el controlador de eventos de componentes `on()`. El controlador `on()` puede asignarse a una instancia de componente igual que se asigna un controlador a un botón o a un clip de película.

La palabra clave `this`, utilizada dentro de un controlador `on()` asociado a un componente, se refiere a la instancia de componente. Por ejemplo, el código siguiente, asociado a la instancia del componente Button `myButtonComponent`, envía “_level0.myButtonComponent” al panel Salida:

```
on(click){  
    trace(this);  
}
```

Para utilizar el controlador `on()`:

- 1 Arrastre un componente CheckBox desde el panel Componentes al escenario.
- 2 Seleccione el componente y elija Ventana > Acciones.
- 3 En este panel, introduzca los códigos siguientes:

```
on(click){  
    trace("clic en CheckBox");  
}
```

Puede introducir cualquier código que desee entre las llaves `{ }`.

- 4 Seleccione Control > Probar película y haga clic en la casilla de verificación para ver el trazado en el panel Salida.

Para más información, consulte la entrada correspondiente a cada evento en el [Capítulo 4](#), “Diccionario de componentes”, en la página 45.

Utilización de detectores de eventos de componentes

La forma más eficaz de gestionar eventos de componentes consiste en utilizar detectores. Los eventos se difunden mediante componentes y cualquier objeto que esté registrado como detector en el difusor de eventos (instancia de componente) puede recibir notificación del evento. Al detector se le asigna una función que gestiona el evento. Se pueden registrar varios detectores en una instancia de componente. Asimismo, se puede registrar un detector en varias instancias de componente.

Para utilizar el modelo de detector de eventos, debe crear un objeto detector con una propiedad que sea el nombre del evento. La propiedad se asigna a una función callback. A continuación, llama al método `UIEventDispatcher.addEventListener()` en la instancia de componente que difunde el evento y le pasa el nombre del evento y el del objeto detector. Llamar al método `UIEventDispatcher.addEventListener()` también recibe el nombre de “registrarse” o “suscribirse” a un detector, como se muestra a continuación:

```
listenerObject.eventName = function(evtObj){  
    // aquí código propio  
};  
componentInstance.addEventListener("eventName", listenerObject);
```

En el código anterior, la palabra clave `this`, si se utiliza en la función callback, tiene su ámbito en `listenerObject`.

El parámetro `evtObj` es un objeto `Event` que se genera automáticamente cuando se activa un evento y se pasa a la función callback del objeto detector. El objeto `Event` tiene propiedades que contienen información sobre el evento. Para más información, consulte “Clase `UIEventDispatcher`” en la página 234.

Para información sobre los eventos como difusiones de componentes, consulte la entrada correspondiente de cada componente en el [Capítulo 4, “Diccionario de componentes”, en la página 45](#).

Para registrar un detector de eventos, utilice el procedimiento siguiente:

- 1 Arrastre un componente `Button` desde el panel Componentes al escenario.
- 2 En el inspector de propiedades, introduzca el nombre de instancia **button**.
- 3 Arrastre un componente `TextInput` desde el panel Componentes al escenario.
- 4 En el inspector de propiedades, introduzca el nombre de instancia **myText**.
- 5 Seleccione el fotograma 1 de la línea de tiempo.
- 6 Seleccione **Ventana > Acciones**.
- 7 En este panel, introduzca los códigos siguientes:

```
form = new Object();  
form.click = function(evt){  
    myText.text = evt.target;  
}  
button.addEventListener("click", form);
```

La propiedad de destino del objeto `Event` es una referencia a la instancia que difunde el evento. Este código muestra el valor de la propiedad de destino en el campo de entrada de texto.

Sintaxis de eventos adicionales

Además de utilizar un objeto detector, también puede utilizar una función como detector. Un detector es una función si no pertenece a ningún objeto. Por ejemplo, el código siguiente crea la función de detección `myHandler` y la registra en `buttonInstance`:

```
function myHandler(eventObj){
    if (eventObj.type == "click"){
        // el código se escribe aquí
    }
}
buttonInstance.addEventListener("click", myHandler);
```

Nota: en un detector de función, la palabra clave `this` es `buttonInstance`, no la línea de tiempo en la que se define la función.

También puede utilizar objetos detectores que admitan el método `handleEvent`. Con independencia del nombre del evento, se llama al método `handleEvent` del objeto detector. Para gestionar varios eventos, es preciso utilizar una sentencia `if else` o `switch`, por lo que la sintaxis es complicada. Por ejemplo, el código siguiente utiliza una sentencia `if else` para gestionar los eventos `click` y `enter`:

```
myObj.handleEvent = function(o){
    if (o.type == "click"){
        // el código se escribe aquí
    } else if (o.type == "enter"){
        // el código se escribe aquí
    }
}
target.addEventListener("click", myObj);
target2.addEventListener("enter", myObj);
```

Existe otro estilo de sintaxis de evento que sólo debe utilizarse cuando se está editando un componente y se sabe que un objeto determinado es el único detector de un evento. En dicho caso, puede aprovechar que el modelo de evento de la v2 siempre llama a un método de la instancia de componente que es el nombre del evento más “Handler”. Por ejemplo, si desea gestionar el evento `click`, escriba el código siguiente:

```
componentInstance.clickHandler = function(o){
    // introduzca aquí el código propio
}
```

En el código anterior, la palabra clave `this`, si se utiliza en la función callback, tiene el ámbito de `componentInstance`.

Para más información, consulte [“Creación de componentes”](#). Es posible que tenga que descargar la versión más actualizada del PDF del Centro de soporte de Flash para consultar esta información.

Creación de un desplazamiento personalizado de la selección

Cuando un usuario presiona la tecla Tabulador para navegar por una aplicación de Flash o hace clic en una aplicación, [Clase FocusManager](#) determina qué componente se selecciona. No es preciso añadir una instancia de `FocusManager` a una aplicación, ni escribir código para activarlo.

Si un objeto `RadioButton` se selecciona, `FocusManager` examina el objeto, así como todos los objetos con el mismo valor `groupName`, y define la selección del objeto estableciendo el valor `true` en la propiedad `selected`.

Cada componente Window modal contiene una instancia de FocusManager para que los controles de dicha ventana se conviertan en su propio conjunto de tabuladores, lo que evita que el usuario acceda sin darse cuenta a los componentes de otras ventanas al presionar la tecla Tabulador.

Para que la selección pueda desplazarse por la aplicación, defina la propiedad `tabIndex` en los componentes que deban seleccionarse (botones incluidos). Cuando un usuario presiona la tecla Tabulador, [Clase FocusManager](#) busca un objeto activado cuya propiedad `tabIndex` sea superior al valor actual de `tabIndex`. Una vez que [Clase FocusManager](#) llegue a la propiedad `tabIndex` más alta, vuelve a cero. Por ejemplo, en este caso, el objeto `comment` (probablemente un componente TextArea) es el primero en seleccionarse y después se selecciona el objeto `okButton`:

```
comment.tabIndex = 1;
okButton.tabIndex = 2;
```

Para crear un botón que se seleccione cuando el usuario presione la tecla Intro (Windows) o Retorno (Macintosh), defina la propiedad `FocusManager.defaultPushButton` en el nombre de instancia del botón deseado, como en el ejemplo siguiente:

```
FocusManager.defaultPushButton = okButton;
```

La [Clase FocusManager](#) sustituye el rectángulo de selección de Flash Player predeterminado y dibuja un rectángulo de selección personalizado con esquinas redondeadas.

Administración de la profundidad del componente en un documento

Si desea colocar un componente por encima o por debajo de otro objeto de la aplicación, deberá utilizar [Clase DepthManager](#). La interfaz API de DepthManager le permite colocar los componentes de la interfaz de usuario (IU) en un orden en z apropiado (por ejemplo un cuadro combinado se despliega delante de otros componentes, aparecen puntos de inserción delante de todos los elementos, aparecen ventanas de diálogo sobre el contenido, etc.).

DepthManager tiene dos objetivos principales: administrar las asignaciones de profundidad relativa en cualquier documento y administrar las profundidades reservadas en la línea de tiempo raíz para servicios del sistema como, por ejemplo, el cursor y las sugerencias.

Para utilizar DepthManager, llame a sus métodos (véase [“Clase DepthManager” en la página 95](#)).

El código siguiente coloca la instancia del componente `loader` bajo el componente `button`:

```
loader.setDepthBelow(button);
```

Utilización de preloader con componentes

Los componentes están definidos de forma predeterminada en Exportar en primer fotograma. Esto hace que los componentes se carguen antes de que aparezca el primer fotograma de una aplicación. Si desea crear un preloader para una aplicación, debe anular la selección de Exportar en primer fotograma para cualquier símbolo de clip compilado de la biblioteca.

Nota: si está utilizando el componente ProgressBar para mostrar el progreso de carga, deje seleccionado Exportar en primer fotograma para ProgressBar.

Actualización de los componentes de la v1 a la arquitectura de la v2

Los componentes de la v2 se han escrito para que cumplan diversos estándares de Internet (relativos a [eventos](#), estilos, políticas de captador/definidor, etc.) y son muy diferentes de sus equivalentes de la v1 incorporados en Macromedia Flash MX y en los DRK que se usaban antes de Macromedia Flash MX 2004. Los componentes de la v2 tienen interfaces API distintas y se han escrito en ActionScript 2. Por tanto, la utilización conjunta de componentes de la v1 y la v2 en la misma aplicación puede provocar un comportamiento imprevisible. Para obtener información sobre cómo actualizar los componentes de la v1 para utilizar la gestión de eventos, los estilos y el acceso al captador/definidor de la versión 2 en las propiedades en lugar de los métodos, consulte [“Creación de componentes”](#). Es posible que tenga que descargar la versión más actualizada del PDF del Centro de soporte de Flash para consultar esta información.

Las aplicaciones de Flash que contienen componentes de la v1 funcionan correctamente en Flash Player 6 y Flash Player 7 cuando se publican para Flash Player 6 o Flash Player 6 versión 65. Si desea actualizar las aplicaciones para que funcionen cuando se publiquen para Flash Player 7, debe convertir el código para que utilice strict data-typing. Para más información, consulte [“Creación de clases con ActionScript 2”](#) en el apartado Diccionario de ActionScript de la Ayuda.

CAPÍTULO 3

Personalización de componentes

Puede cambiar la apariencia de los componentes cuando los utiliza en diferentes aplicaciones. Para ello, Macromedia Flash MX 2004 y Macromedia Flash MX Professional 2004 le ofrecen tres formas distintas:

- Utilizar la interfaz API de estilos.
- Aplicar un tema.
- Modificar o sustituir los aspectos de un componente.

La interfaz API (interfaz de programación de aplicaciones) de estilos dispone de métodos y propiedades que permiten cambiar el color y el formato de texto de un componente.

Un tema es un conjunto de estilos y aspectos que forman la apariencia de un componente.

Los aspectos son símbolos que se utilizan para visualizar componentes. La *aplicación de aspectos* es el proceso mediante el cual se cambia la apariencia de un componente, modificando o sustituyendo el gráfico de origen. Un aspecto puede ser una parte pequeña, como un borde o esquina de un borde, o bien una parte compuesta como la imagen completa de un botón sin oprimir (estado en el que se encuentra cuando no se ha presionado). Asimismo, puede ser un símbolo sin gráfico que contiene código que dibuja una parte del componente.

Utilización de estilos para personalizar el texto y el color de un componente

Todas las instancias de componente tienen propiedades de estilo y un método `setStyle()` y `getStyle()` (véase `UIObject.setStyle()` y `UIObject.getStyle()`) que se pueden utilizar para modificar las propiedades de estilo y tener acceso a las mismas. Puede utilizar los estilos para personalizar un componente de varias formas:

- Defina los estilos en una instancia de componente.
Puede cambiar las propiedades de texto y de color de una única instancia de componente. Este recurso es efectivo en determinadas situaciones, aunque puede tardar mucho tiempo si necesita definir propiedades individuales en todos los componentes de un documento.
- Utilice la declaración de estilo `_global` que define los estilos de todos los componentes de un documento.

Si desea que el documento completo tenga un aspecto uniforme, puede crear estilos en la declaración de estilo `_global`.

- Cree declaraciones de estilo personalizadas y aplíquelas a instancias de componente específicas. También puede hacer que grupos de componentes de un documento compartan un mismo estilo. Para ello, puede crear declaraciones de estilo personalizadas para aplicarlas a componentes específicos.
- Cree declaraciones de estilo de clase predeterminadas. También puede definir una declaración de estilo de clase predeterminada para que todas las instancias de una misma clase compartan una apariencia predeterminada.

Los cambios efectuados en las propiedades de estilo no aparecen al ver los componentes en el escenario mediante la función Vista previa dinámica. Para más información, consulte [“Componentes en vista previa dinámica” en la página 17.](#)

Definición de los estilos en una instancia de componente

Puede escribir código ActionScript para definir y obtener las propiedades de estilo de cualquier instancia de componente. Los métodos `UIObject.setStyle()` y `UIObject.getStyle()` pueden llamarse directamente desde cualquier componente. Por ejemplo, el código siguiente define el color de texto de una instancia de botón denominada `myButton`:

```
myButton.setStyle("color", 0xFF00FF);
```

Aunque puede tener acceso directo a los estilos como propiedades (por ejemplo, `myButton.color = 0xFF00FF`), es preferible utilizar los métodos `setStyle()` y `getStyle()` para que los estilos funcionen correctamente. Para más información, consulte [“Definición de valores de la propiedad de estilo” en la página 32.](#)

Nota: para definir más de una propiedad, no llame varias veces al método `UIObject.setStyle()`. Si desea cambiar varias propiedades o cambiar propiedades de varias instancias de componente, debe crear un formato de estilo personalizado. Para más información, consulte [“Definición de estilos para componentes específicos” en la página 29.](#)

Para definir o cambiar una propiedad de una sola instancia de componente:

- 1 Seleccione la instancia de componente en el escenario.
- 2 En el inspector de propiedades, asígnele el nombre de instancia **myComp**.
- 3 Abra el panel Acciones y seleccione Escena 1 y, a continuación, Capa 1: Fotograma 1.
- 4 Introduzca el código siguiente para cambiar el color de la instancia a azul:

```
myComp.setStyle("themeColor", "haloBlue");
```

La sintaxis siguiente especifica una propiedad y un valor para una instancia de componente:

```
instanceName.setStyle("propiedad", valor);
```

- 5 Seleccione Control > Probar película para ver los cambios.

Para obtener una lista de los estilos admitidos, consulte [“Estilos admitidos” en la página 33.](#)

Definición de estilos globales

La declaración de estilo `_global` se asigna a todos los componentes de Flash incorporados en la versión 2 de la arquitectura de componentes de Macromedia (componentes de la v2). El objeto `_global` tiene una propiedad llamada `style (_global.style)` que es una instancia de `CSSStyleDeclaration`. La propiedad `style` actúa como declaración de estilo `_global`. Si cambia el valor de una propiedad en la declaración de estilo `_global`, el cambio se aplica a todos los componentes del documento de Flash.

Algunos estilos se definen en la `CSSStyleDeclaration` de la clase de componente (por ejemplo, el estilo `backgroundColor` de los componentes `TextArea` y `TextInput`). La declaración de estilo de clase tiene prioridad sobre la declaración de estilo `_global` a la hora de determinar los valores de estilo, por lo que la definición de `backgroundColor` en la declaración de estilo `_global` no tendría efecto alguno sobre `TextArea` o `TextInput`. Para más información, consulte [“Utilización de los estilos global, personalizado y de clase en el mismo documento” en la página 31](#).

Para cambiar una o más propiedades en la declaración de estilo global:

- 1 Asegúrese de que el documento contiene como mínimo una instancia de componente.
Para más información, consulte [“Adición de componentes a documentos de Flash” en la página 18](#).
- 2 Cree una nueva capa en la línea de tiempo y asígnele un nombre.
- 3 Seleccione un fotograma en la capa nueva donde aparece el componente (o antes de que aparezca).
- 4 Abra el panel Acciones.
- 5 Utilice la sintaxis siguiente para cambiar las propiedades de la declaración de estilo `_global`. Sólo es necesario enumerar las propiedades cuyos valores desea cambiar, como se muestra a continuación:

```
_global.style.setStyle("color", 0xCC6699);  
_global.style.setStyle("themeColor", "haloBlue")  
_global.style.setStyle("fontSize", 16);  
_global.style.setStyle("fontFamily" , "_serif");
```


Para obtener una lista de estilos, consulte [“Estilos admitidos” en la página 33](#).
- 6 Seleccione Control > Probar película para ver los cambios.

Definición de estilos para componentes específicos

Puede crear declaraciones de estilo personalizadas para especificar un conjunto exclusivo de propiedades para componentes específicos de un documento de Flash. Cree una instancia nueva del objeto `CSSStyleDeclaration`, cree un nombre de estilo personalizado y póngalo en la lista `_global.styles` (`_global.styles.newStyle`), especifique las propiedades y valores del estilo, y asígnele a una instancia. Se puede acceder al objeto `CSSStyleDeclaration` si se ha colocado como mínimo una instancia de componente en el escenario.

Para modificar un formato de estilo personalizado se sigue el mismo procedimiento que para editar las propiedades de la declaración de estilo `_global`. En lugar de utilizar el nombre de la declaración de estilo `_global`, utilice la instancia `CSSStyleDeclaration`. Para más información sobre la declaración de estilo `_global`, consulte [“Definición de estilos globales” en la página 28](#).

Para obtener información sobre las propiedades del objeto `CSSStyleDeclaration`, consulte [“Estilos admitidos” en la página 33](#). Para obtener una lista de los estilos que los componentes admiten, consulte sus entradas individuales en el [Capítulo 4, “Diccionario de componentes”, en la página 45](#).

Para crear una declaración de estilo personalizada para componentes específicos:

- 1 Asegúrese de que el documento contiene como mínimo una instancia de componente.
Para más información, consulte [“Adición de componentes a documentos de Flash” en la página 18](#).

En este ejemplo se utilizan tres componentes de botón con los nombres de instancia a, b y c. Si utiliza otros componentes, asígneles nombres de instancia en el inspector de propiedades y utilice dichos nombres en el paso 9.

- 2 Cree una nueva capa en la línea de tiempo y asígnele un nombre.
- 3 Seleccione un fotograma en la capa nueva donde aparece el componente (o antes de que aparezca).
- 4 Abra el panel Acciones en modo Experto.
- 5 Utilice la sintaxis siguiente para crear una instancia del objeto `CSSStyleDeclaration` con el fin de definir el nuevo estilo de formato personalizado:

```
var styleObj = new mx.styles.CSSStyleDeclaration;
```

- 6 Defina la propiedad `styleName` de la declaración de estilo para asignar un nombre al estilo:

```
styleObj.styleName = "newStyle";
```

- 7 Ponga el estilo en la lista global de estilos:

```
_global.styles.newStyle = styleObj;
```

Nota: también puede crear un objeto `CSSStyleDeclaration` y asignarlo a una declaración de estilo nueva mediante la sintaxis siguiente:

```
var styleObj = _global.styles.newStyle = new  
mx.styles.CSSStyleDeclaration();
```

- 8 Utilice la sintaxis siguiente para especificar las propiedades que desea definir para la declaración de estilo `myStyle`:

```
styleObj.fontFamily = "_sans";  
styleObj.fontSize = 14;  
styleObj.fontWeight = "bold";  
styleObj.textDecoration = "underline";  
styleObj.color = 0x336699;  
styleObj.setStyle("themeColor", "haloBlue");
```

- 9 En el mismo panel Script, utilice la sintaxis siguiente para definir la propiedad `styleName` de dos componentes específicos en la declaración de estilo personalizada:

```
a.setStyle("styleName", "newStyle");  
b.setStyle("styleName", "newStyle");
```

También puede tener acceso a los estilos de una declaración de estilo personalizada mediante los métodos `setStyle()` y `getStyle()`. El código siguiente define el estilo `backgroundColor` en la declaración de estilo `newStyle`:

```
_global.styles.newStyle.setStyle("backgroundColor", "0xFFCCFF");
```

Definición de estilos para una clase de componente

Puede definir una declaración de estilo de clase para cualquier clase de componente (`Button`, `CheckBox`, etc.) que defina los estilos predeterminados de cada instancia de dicha clase. Antes de crear las instancias, es preciso crear la declaración de estilo. Algunos componentes, como `TextArea` y `TextInput`, tienen declaraciones de estilo de clase predefinidas de forma predeterminada, ya que las propiedades `borderStyle` y `backgroundColor` deben personalizarse.

El código siguiente crea una declaración de estilo de clase para `CheckBox` y define el color azul para la casilla de verificación:

```
var o = _global.styles.CheckBox = new mx.styles.CSSStyleDeclaration();  
o.color = 0x0000FF;
```

También puede tener acceso a los estilos de una declaración de estilo de clase mediante los métodos `setStyle()` y `getStyle()`. El código siguiente define el estilo de color en la declaración de estilo `RadioButton`:

```
_global.styles.RadioButton.setStyle("color", "blue");
```

Para más información sobre los estilos admitidos, consulte [“Estilos admitidos” en la página 33](#).

Utilización de los estilos global, personalizado y de clase en el mismo documento

Si define un estilo sólo en un sitio del documento, Flash utilizará dicha definición cuando necesite saber el valor de una propiedad. No obstante, un único documento de Flash puede tener una declaración de estilo `_global`, declaraciones de estilo personalizadas, propiedades de estilo definidas directamente en instancias de componente y declaraciones de estilo de clase predeterminadas. En dicha situación, Flash determina el valor de una propiedad buscando su definición en todos los lugares mencionados, siguiendo un orden específico.

En primer lugar, Flash buscará una propiedad de estilo en la instancia de componente. Si el estilo no está directamente definido en la instancia, Flash buscará la propiedad `styleName` de la instancia para ver si tiene asignada una declaración de estilo.

Si no se ha asignado la propiedad `styleName` a ninguna declaración de estilo, Flash buscará la propiedad en la declaración de estilo de clase predeterminada. Si no hay declaración de estilo de clase y la propiedad no hereda su valor, se activará la declaración de estilo `_global`. Si la propiedad no está definida en la declaración de estilo `_global`, la propiedad es `undefined`.

Si no hay declaración de estilo de clase y la propiedad hereda su valor, Flash buscará la propiedad en la instancia principal. Si la propiedad no está definida en la instancia principal, Flash comprobará la propiedad `styleName` de la instancia principal; si no está definida, Flash seguirá buscando en instancias principales hasta que llegue al nivel `_global`. Si la propiedad no está definida en la declaración de estilo `_global`, la propiedad es `undefined`.

`StyleManager` indica a Flash si un estilo hereda su valor o no. Para más información, consulte [“Clase `StyleManager`” en la página 202](#).

Nota: no se admite el valor `"inherit"` de CSS.

Propiedades del estilo de color

Las propiedades del estilo de color tienen un comportamiento diferente de las propiedades que no son de color. Todas las propiedades tienen un nombre que termina por “Color”; por ejemplo, `backgroundColor`, `disabledColor` y `color`. Cuando se cambian las propiedades del estilo de color, inmediatamente se cambia también el color de la instancia y de todas las instancias secundarias correspondientes. Los cambios restantes en la propiedad de estilo sólo indican que el objeto se debe volver a dibujar y que los cambios se producirán a partir del fotograma siguiente.

El valor de una propiedad de estilo de color puede ser un número, una cadena o un objeto. Si se trata de un número, representa el valor RVA del color como número hexadecimal (0xRRGGBB). Si el valor es una cadena, debe ser un nombre de color.

Los nombres de color son cadenas que se asignan a los colores que se utilizan con mayor frecuencia. Se pueden añadir nombres de color nuevos mediante `StyleManager` (véase [“Clase `StyleManager`” en la página 202](#)). En la tabla siguiente se enumeran los nombres de color predeterminados:

Nombre del color	Valor
black	0x000000
white	0xFFFFFFFF
red	0xFF0000
green	0x00FF00
blue	0x0000FF
magenta	0xFF00FF
yellow	0xFFFF00
cyan	0x00FFFF

Nota: si el nombre del color no está definido, es posible que el componente no se dibuje correctamente.

Puede utilizar cualquier identificador de `ActionScript` válido para crear nombres de color propios (por ejemplo, `"WindowText"` o `"ButtonText"`). Utilice `StyleManager` para definir colores nuevos, como en el ejemplo siguiente:

```
mx.styles.StyleManager.registerColorName("special_blue", 0x0066ff);
```

La mayoría de los componentes no pueden gestionar un objeto como valor de la propiedad de estilo de color. No obstante, determinados componentes pueden gestionar objetos de color que representan degradados u otras combinaciones de color. Para más información, consulte la sección “Utilización de estilos” de cada componente en el [Capítulo 4, “Diccionario de componentes”, en la página 45](#).

Puede utilizar las declaraciones de estilo de clase y los nombres de color para controlar fácilmente los colores del texto y los símbolos que se muestran en la pantalla. Por ejemplo, si desea una pantalla de configuración de la visualización que tenga la apariencia de Microsoft Windows, defina nombres de color como `ButtonText` y `WindowText`, y declaraciones de estilo de clase como `Button`, `CheckBox` y `Window`. Si define las propiedades del estilo de color de las declaraciones de estilo en `ButtonText` y `WindowText`, y proporciona una interfaz de usuario para que el usuario pueda cambiar los valores de `ButtonText` y `WindowText`, podrá proporcionar los mismos esquemas de color que Microsoft Windows, Macintosh OS o cualquier otro sistema operativo.

Definición de valores de la propiedad de estilo

Utilice el método `UIObject.setStyle()` para definir una propiedad de estilo en una instancia de componente, en la declaración de estilo global, en una declaración de estilo personalizada o en una declaración de estilo de clase. El código siguiente define el estilo de `color` de la instancia de un botón de opción en rojo:

```
myRadioButton.setStyle("color", "red");
```

El código siguiente define el estilo de `color` de la declaración de estilo personalizada `CheckBox`:

```
_global.styles.CheckBox.setStyle("color", "white");
```


El método `UIObject.setStyle()` sabe si un estilo hereda y notifica a los secundarios de la instancia si su estilo cambia. Asimismo, notifica a la instancia del componente que debe volver a dibujarse para reflejar el estilo nuevo. Por consiguiente, debe utilizar `setStyle()` para definir o cambiar estilos. No obstante, como optimización a la hora de crear declaraciones de estilo, puede definir directamente las propiedades en un objeto. Para más información, consulte [“Definición de estilos globales” en la página 28](#), [“Definición de estilos para componentes específicos” en la página 29](#) y [“Definición de estilos para una clase de componente” en la página 30](#).

Utilice el método `UIObject.getStyle()` para recuperar un estilo de una instancia de componente, la declaración de estilo `_global`, una declaración de estilo personalizada o una declaración de estilo de clase. El código siguiente obtiene el valor de la propiedad de color y lo asigna a la variable `o`:

```
var o = myRadioButton.getStyle("color");
```

El código siguiente obtiene el valor de una propiedad de estilo definida en la declaración de estilo `_global`:

```
var r = _global.style.getValue("marginRight");
```

Si el estilo no está definido, `getStyle()` puede devolver el valor `undefined`. No obstante, `getStyle()` comprende cómo se heredan las propiedades de estilo. Por consiguiente, aunque los estilos son propiedades, debe utilizar `UIObject.getStyle()` para tener acceso a los mismos sin tener que saber si el estilo se hereda.

Para más información, consulte `UIObject.getStyle()` y `UIObject.setStyle()`.

Estilos admitidos

Flash MX 2004 y Flash MX Professional 2004 se entregan con dos temas: *Halo* (`HaloTheme fla`) y *Sample* (`SampleTheme fla`). Cada tema admite un conjunto de estilos diferente. El tema *Sample* utiliza todos los estilos del mecanismo de estilos v2 y se proporciona para que pueda ver una muestra de dichos estilos en un documento. El tema *Halo* admite un subconjunto de los estilos del tema *Sample*.

La mayoría de los componentes de la v2 del estilo *Sample* admiten las propiedades de estilo siguientes. Para obtener información sobre qué estilos de *Halo* admiten los componentes individuales, consulte el [Capítulo 4, “Diccionario de componentes”, en la página 45](#).

Si se introducen valores no permitidos, se utilizará el valor predeterminado. Esto resulta importante si se vuelven a utilizar declaraciones de estilo CSS que utilizan valores que no pertenecen al subconjunto de valores de Macromedia.

Los componentes pueden admitir los estilos siguientes:

Estilo	Descripción
<code>backgroundColor</code>	Fondo de un componente. Éste es el único estilo de color que no hereda su valor. El valor predeterminado es transparente.
<code>borderColor</code>	Sección negra de un borde tridimensional o sección de color de un borde bidimensional. El valor predeterminado es <code>0x000000</code> (negro).
<code>borderStyle</code>	Borde del componente: puede ser <code>“none”</code> , <code>“inset”</code> , <code>“outset”</code> o <code>“solid”</code> . Este estilo no hereda su valor. El valor predeterminado es <code>“solid”</code> .

Estilo	Descripción
buttonColor	La cara de un botón y una sección del borde tridimensional. El valor predeterminado es 0xEFEFEF (gris claro).
color	Texto de la etiqueta de un componente. El valor predeterminado es 0x000000 (negro).
disabledColor	Color desactivado del texto. El color predeterminado es 0x848384 (gris oscuro).
fontFamily	Nombre de fuente del texto. El valor predeterminado es _sans.
fontSize	Tamaño en puntos de la fuente. El valor predeterminado es 10.
fontStyle	Estilo de la fuente: puede ser "normal" o "italic". El valor predeterminado es "normal".
fontWeight	Grosor de la fuente: puede ser "normal" o "bold". El valor predeterminado es "normal".
highlightColor	Sección del borde tridimensional. El valor predeterminado es 0xFFFFFFFF (blanco).
marginLeft	Número que indica el margen izquierdo del texto. El valor predeterminado es 0.
marginRight	Número que indica el margen derecho del texto. El valor predeterminado es 0.
scrollTrackColor	Guía de desplazamiento de la barra de desplazamiento. El valor predeterminado es 0xEFEFEF (gris claro).
shadowColor	Sección del borde tridimensional. El valor predeterminado es 0x848384 (gris oscuro).
symbolBackgroundColor	Color de fondo de las casillas de verificación y de los botones de opción. El valor predeterminado es 0xFFFFFFFF (blanco).
symbolBackgroundDisabledColor	Color de fondo de las casillas de verificación y de los botones de opción cuando están desactivados. El valor predeterminado es 0xEFEFEF (gris claro).
symbolBackgroundPressedColor	Color de fondo de las casillas de verificación y de los botones de opción cuando se presionan. El valor predeterminado es 0xFFFFFFFF (blanco).
symbolColor	Marca de verificación de la casilla de verificación o punto del botón de opción. El valor predeterminado es 0x000000 (negro).
symbolDisabledColor	Color de la marca de verificación o del punto del botón de opción desactivado. El valor predeterminado es 0x848384 (gris oscuro).
textAlign	Alineación del texto: puede ser "left", "right" o "center". El valor predeterminado es "left".

Estilo	Descripción
textDecoration	Decoración del texto: puede ser "none" o "underline". El valor predeterminado es "none".
textIndent	Número que indica la sangría del texto. El valor predeterminado es 0.

Temas

Los temas son conjuntos de estilos y de aspectos. El tema predeterminado de Flash MX 2004 y Flash MX Professional 2004 se llama Halo (HaloTheme fla). El tema Halo se ha desarrollado para permitir que los usuarios saquen el máximo rendimiento posible. Flash MX 2004 y Flash MX Professional 2004 incluye otro tema llamado Sample (SampleTheme fla). El tema Sample le permite experimentar con el conjunto completo de estilos disponibles para los componentes de la v2. El tema Halo sólo utiliza un subconjunto de los estilos disponibles. Los archivos del tema se encuentran en las carpetas siguientes:

- Windows: First Run\ComponentFLA
- Macintosh: First Run\ComponentFLA

Puede crear temas nuevos y aplicarlos a una aplicación para cambiar la apariencia de todos los componentes. Por ejemplo, puede crear un tema bidimensional y otro tridimensional.

Los componentes de v2 utilizan aspectos (símbolos de clip de película o gráficos) para visualizar las apariencias visuales. El archivo .as que define cada componente contiene código que carga aspectos específicos del componente. Por ejemplo, ScrollBar está programado para cargar el símbolo con el identificador de vinculación `ScrollDownArrowDown` como aspecto para el estado Presionado (abajo) de la flecha abajo. Puede crear fácilmente un tema nuevo haciendo una copia del tema Halo o el tema Sample y modificando los gráficos de los aspectos.

Un tema también puede contener un conjunto de estilos nuevo. Para crear una declaración de estilo global o cualquier otra declaración de estilo adicional, es preciso escribir código ActionScript. Para más información, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 27](#).

Aplicación de un tema a un documento

Para aplicar un tema nuevo a un documento, abra un archivo FLA del tema como biblioteca externa y arrastre la carpeta Theme desde la biblioteca externa hasta la biblioteca del documento. Los pasos siguientes explican el proceso en detalle.

Para aplicar un tema a un documento:

- 1 Seleccione Archivo > Abrir y abra el documento que utiliza componentes de v2 en Flash o seleccione Archivo > Nuevo y cree un documento nuevo que utilice componentes de v2.
- 2 Seleccione Archivo > Guardar y elija un nombre exclusivo como **ThemeApply fla**.
- 3 Seleccione Archivo > Importar > Abrir biblioteca externa y seleccione el archivo FLA del tema que desee aplicar al documento.

Si no ha creado un tema nuevo, puede utilizar el tema Sample de la carpeta Flash 2004/es/Configuration/SampleFLA.

- 4 En el panel Biblioteca del tema, seleccione Flash UI Components 2 > Themes > MMDefault y arrastre la carpeta Assets de cualquier componente del documento a la biblioteca ThemeApply fla.

Si no está seguro de qué componentes se encuentran en los documentos, puede arrastrar la carpeta Themes completa al escenario. Los aspectos que se encuentran en la carpeta Themes de la biblioteca se asignan automáticamente a componentes del documento.

Nota: la vista previa dinámica de los componentes del escenario no reflejará el tema nuevo.

- 5 Seleccione Control > Probar película para ver el documento con el tema nuevo aplicado.

Creación de un tema nuevo

Si no desea utilizar el tema Halo ni el tema Sample, puede modificar cualquiera de ellos para crear un tema nuevo.

Algunos aspectos de los temas tienen un tamaño fijo. Puede agrandarlos o reducirlos y los componentes cambiarán de tamaño automáticamente para coincidir con ellos. Otros aspectos están formados por varias piezas, algunas de ellas estáticas y otras no.

Algunos aspectos (por ejemplo RectBorder y ButtonSkin) utilizan la interfaz API de dibujo de ActionScript para dibujar sus gráficos, ya que es más eficiente desde el punto de vista del tamaño y del rendimiento. En dichos aspectos, puede utilizar código ActionScript como plantilla para ajustar los aspectos a sus necesidades.

Para crear un tema nuevo:

- 1 Seleccione el archivo FLA del tema que desee utilizar como plantilla y haga una copia.
Asigne a la copia un nombre exclusivo como **MyTheme fla**.
- 2 Seleccione Archivo > Abrir MyTheme fla en Flash.
- 3 Seleccione Ventana > Biblioteca para abrir la biblioteca si aún no está abierta.
- 4 Haga doble clic en el símbolo de aspecto que desee modificar para abrirlo en el modo Editar símbolo.
Los aspectos se encuentran en la carpeta Themes > MMDefault > *Componente Assets* (en este ejemplo, Themes > MMDefault > RadioButton Assets).
- 5 Modifique el símbolo o elimine los gráficos y cree gráficos nuevos.
Puede que necesite seleccionar Ver > Acercar para subir el grado de aumento. Si se edita un aspecto, es preciso mantener el punto de registro para que el aspecto se vea correctamente. La esquina superior izquierda de todos los símbolos editados debe estar en (0,0).
- 6 Cuando haya terminado de editar el símbolo del aspecto, haga clic en el botón Atrás situado en la parte izquierda de la barra de información en el área superior del escenario para volver al modo Editar documento.
- 7 Repita los pasos 4, 5 y 6 hasta que haya terminado de editar todos los aspectos que desee cambiar.
- 8 Aplique MyTheme fla a un documento siguiendo los pasos indicados en la sección anterior, [“Aplicación de un tema a un documento” en la página 35](#).

Aplicación de aspectos a los componentes

Los aspectos son símbolos que un componente utiliza para mostrar su apariencia. Pueden ser símbolos gráficos o símbolos de clip de película. La mayoría de los aspectos contienen formas que representan la apariencia del componente. Algunos aspectos contienen sólo código ActionScript que dibuja el componente en el documento.

Los componentes de Macromedia v2 son clips compilados, cuyos elementos no se pueden ver en la biblioteca. No obstante, con Flash se instalan archivos FLA que contienen todos los aspectos de componente. Estos archivos FLA reciben el nombre de *temas*. Cada tema tiene un comportamiento y una apariencia diferentes, aunque contiene aspectos con los mismos nombres de símbolo e identificadores de vínculo. Esto le permite arrastrar un tema al escenario de un documento para cambiar su apariencia. Para más información sobre los temas, consulte [“Temas” en la página 35](#). Los archivos FLA del tema también pueden utilizarse para editar aspectos de componente. Los aspectos se encuentran en la carpeta Themes del panel Biblioteca de cada FLA del tema.

Cada componente está formado por varios aspectos. Por ejemplo, la flecha abajo del componente ScrollBar está formada por tres aspectos: ScrollDownArrowDisabled, ScrollDownArrowUp y ScrollDownArrowDown. Algunos componentes comparten aspectos. Los componentes que utilizan barras de desplazamiento (incluidos ComboBox, List, ScrollBar y ScrollPane) comparten los aspectos de la carpeta ScrollBar Skins. Puede editar los aspectos ya existentes y crear aspectos nuevos para cambiar la apariencia de un componente.

El archivo .as que define cada clase de componente contiene código que carga aspectos específicos del componente. Cada aspecto de un componente tiene una propiedad de aspecto que se asigna al identificador de vinculación del símbolo de un aspecto. Por ejemplo, el estado Presionado (abajo) de la flecha abajo de ScrollBar tiene el nombre de propiedad de aspecto `downArrowDownName`. El valor predeterminado de la propiedad `downArrowDownName` es `"ScrollDownArrowDown"`, que es el identificador de vinculación del símbolo de aspecto. Los aspectos se pueden editar y aplicar a un componente mediante las propiedades de dichos aspectos. No tiene que editar el archivo .as del componente para cambiar sus propiedades de aspecto; simplemente, puede pasar los valores de propiedad del aspecto a la función constructora del componente cuando éste se crea en el documento.

Seleccione uno de los siguientes procedimientos para aplicar aspectos a un componente según lo que desee hacer:

- Para sustituir todos los aspectos de un documento por un conjunto nuevo (en el que cada tipo de componente comparta la misma apariencia), aplique un tema (véase [“Temas” en la página 35](#)).

Nota: este método de aplicación de aspectos está recomendado para principiantes, ya que no es preciso crear ningún script.

- Para utilizar diferentes aspectos para varias instancias del mismo componente, edite los aspectos existentes y defina las propiedades del aspecto (véase [“Edición de aspectos de componentes” en la página 38](#) y [“Aplicación de un aspecto editado a un componente” en la página 39](#)).
- Para cambiar aspectos de un subcomponente (como una barra de desplazamiento de un componente List), ponga el componente en una subclase (véase [“Aplicación de un aspecto editado a un subcomponente” en la página 40](#)).

- Para cambiar aspectos de un subcomponente a los que no se tiene acceso directo desde el componente principal (como un componente List de un componente ComboBox), sustituya las propiedades del aspecto en el prototipo (véase [“Cambio de propiedades de aspecto en el prototipo” en la página 42](#)).

Nota: los métodos indicados están enumerados de mayor a menor, basándose en su facilidad de uso.

Edición de aspectos de componentes

Si desea utilizar un aspecto determinado para una instancia de un componente, pero dispone de otro aspecto de otra instancia del componente, debe abrir un archivo FLA del tema y crear un símbolo de aspecto nuevo. Los componentes se han diseñado para facilitar el uso de aspectos diferentes para instancias diferentes.

Para editar un aspecto, siga este procedimiento:

- 1 Seleccione Archivo > Abrir y abra el archivo FLA del tema que desea utilizar como plantilla.
- 2 Seleccione Archivo > Guardar como y, a continuación, seleccione un nombre exclusivo como **MyTheme.fla**.
- 3 Seleccione el aspecto o aspectos que desea editar (en este ejemplo, se trata de RadioTrueUp). Los aspectos se encuentran en la carpeta Themes > MMDefault > *Componente* Assets (en este ejemplo, Themes > MMDefault > RadioButton Assets > States).
- 4 Seleccione Duplicar en el menú Opciones de Biblioteca (o haga clic con el botón derecho del ratón sobre el símbolo) y asígnele un nombre exclusivo como MyRadioTrueUp.
- 5 Seleccione el botón Avanzado en el cuadro de diálogo Propiedades de símbolo y seleccione Exportar para ActionScript.
Se introducirá automáticamente un identificador de vinculación que coincida con el nombre de símbolo.
- 6 Haga doble clic en el aspecto nuevo en la biblioteca para abrirlo en el modo Editar símbolo.
- 7 Modifique el clip de película o elimínelo y cree uno nuevo.
Puede que necesite seleccionar Ver > Acercar para subir el grado de aumento. Si se edita un aspecto, es preciso mantener el punto de registro para que el aspecto se vea correctamente. La esquina superior izquierda de todos los símbolos editados debe estar en (0,0).
- 8 Cuando haya terminado de editar el símbolo del aspecto, haga clic en el botón Atrás situado en la parte izquierda de la barra de información en el área superior del escenario para volver al modo Editar documento.
- 9 Seleccione Archivo > Guardar, pero no cierre MyTheme.fla. Ahora debe crear un documento nuevo donde aplicar el aspecto editado a un componente.
Para más información, consulte [“Aplicación de un aspecto editado a un componente” en la página 39](#), [“Aplicación de un aspecto editado a un subcomponente” en la página 40](#) o [“Cambio de propiedades de aspecto en el prototipo” en la página 42](#). Para obtener información sobre cómo aplicar un aspecto nuevo, consulte [“Aplicación de aspectos a los componentes” en la página 37](#).

Nota: los cambios efectuados en los aspectos de componente no aparecen al ver componentes en el escenario mediante la función Vista previa dinámica.

Aplicación de un aspecto editado a un componente

Cuando haya acabado de editar un aspecto, tendrá que aplicarlo a un componente de un documento. Puede utilizar el método `createClassObject()` para crear dinámicamente las instancias del componente, o bien puede poner manualmente las instancias del componente en el escenario. Existen dos maneras diferentes de aplicar aspectos a instancias de un componente, según cómo añada los componentes al documento.

Para crear un componente de forma dinámica y aplicar un aspecto editado, siga este procedimiento:

- 1 Seleccione Archivo > Nuevo para crear un documento de Flash nuevo.
- 2 Seleccione Archivo > Guardar y asígnele un nombre exclusivo como **DynamicSkinning.fla**.
- 3 Arrastre los componentes desde el panel Componentes hasta el escenario, incluido el componente cuyo aspecto ha editado (en este ejemplo, se trata de `RadioButton`) y elimínelos.
De esta manera, se añaden los símbolos a la biblioteca del documento, aunque no los vuelve visibles en éste.
- 4 Arrastre `MyRadioTrueUp` y cualquier otro símbolo que haya personalizado desde `MyTheme.fla` al escenario de `DynamicSkinning.fla` y elimínelos.
De esta manera, se añaden los símbolos a la biblioteca del documento, aunque no los vuelve visibles en éste.
- 5 Abra el panel Acciones e introduzca lo siguiente en el fotograma 1:

```
import mx.controls.RadioButton
createClassObject(RadioButton, "myRadio", 0, {trueUpIcon:"MyRadioTrueUp",
    etiqueta: "My Radio Button"});
```
- 6 Seleccione Control > Probar película.

Para añadir manualmente un componente al escenario y aplicar un aspecto editado, siga este procedimiento:

- 1 Seleccione Archivo > Nuevo para crear un documento de Flash nuevo.
- 2 Seleccione Archivo > Guardar y asígnele un nombre exclusivo como **ManualSkinning.fla**.
- 3 Arrastre los componentes desde el panel Componentes hasta el escenario, incluido el componente cuyo aspecto ha editado (en este ejemplo, se trata de `RadioButton`).
- 4 Arrastre `MyRadioTrueUp` y cualquier otro símbolo que haya personalizado desde `MyTheme.fla` al escenario de `ManualSkinning.fla` y elimínelos.
De esta manera, se añaden los símbolos a la biblioteca del documento, aunque no los vuelve visibles en éste.
- 5 Seleccione el componente `RadioButton` en el escenario y abra el panel Acciones.
- 6 Asocie el código siguiente a la instancia `RadioButton`:

```
onClipEvent(initialize){
    trueUpIcon = "MyRadioTrueUp";
}
```
- 7 Seleccione Control > Probar película.

Aplicación de un aspecto editado a un subcomponente

En determinadas situaciones, es posible que desee modificar los aspectos de un subcomponente de un componente y se encuentre con que las propiedades de dichos aspectos no están disponibles directamente (por ejemplo, no hay forma directa de modificar los aspectos de la barra de desplazamiento de un componente List). El código siguiente le permite tener acceso a los aspectos de la barra de desplazamiento. Todas las barras de desplazamiento que se creen después de ejecutar este código también tendrán los aspectos nuevos.

Si un componente está formado por subcomponentes, dichos subcomponentes se identifican en la entrada del componente en el [Capítulo 4, “Diccionario de componentes”, en la página 45](#).

Para aplicar un aspecto nuevo a un subcomponente, siga este procedimiento:

- 1 Siga los pasos indicados en [“Edición de aspectos de componentes” en la página 38](#), pero edite un aspecto de la barra de desplazamiento. En este ejemplo, edite el aspecto ScrollDownArrowDown y asígnele el nombre nuevo **MyScrollDownArrowDown**.
- 2 Seleccione Archivo > Nuevo para crear un documento de Flash nuevo.
- 3 Seleccione Archivo > Guardar y asígnele un nombre exclusivo como **SubcomponentProject.fla**.
- 4 Haga doble clic en el componente List del panel Componentes para añadirlo al escenario y presione Retroceso para eliminarlo del mismo.

De esta manera, se añade el componente al panel Biblioteca, aunque no lo vuelve visible en el documento.

- 5 Arrastre MyScrollDownArrowDown y cualquier otro símbolo que haya editado desde MyTheme.fla al escenario de SubcomponentProject.fla y elimínelo.

De esta manera, se añade el componente al panel Biblioteca, aunque no lo vuelve visible en el documento.

- 6 Siga uno de estos procedimientos:

- Si desea cambiar todas las barras de desplazamiento de un documento, introduzca el código siguiente en el panel Acciones, en el fotograma 1 de la línea de tiempo:

```
import mx.controls.List
import mx.controls.scrollClasses.ScrollBar
ScrollBar.prototype.downArrowDownName = "MyScrollDownArrowDown";
```

A continuación puede introducir el código siguiente en el fotograma 1 para crear una lista de forma dinámica:

```
createClassObject(List, "myListBox", 0, {dataProvider: ["AL","AR","AZ",
    "CA","HI","ID", "KA","LA","MA"]});
```

O bien, puede arrastrar un componente List desde la biblioteca al escenario.

- Si desea cambiar una barra de desplazamiento específica de un documento, introduzca el código siguiente en el panel Acciones, en el fotograma 1 de la línea de tiempo:

```
import mx.controls.List
import mx.controls.scrollClasses.ScrollBar
var oldName = ScrollBar.prototype.downArrowDownName;
ScrollBar.prototype.downArrowDownName = "MyScrollDownArrowDown";
createClassObject(List, "myList1", 0, {dataProvider: ["AL","AR","AZ",
    "CA","HI","ID", "KA","LA","MA"]});
myList1.redraw(true);
ScrollBar.prototype.downArrowDownName = oldName;
```

Nota: se debe definir información suficiente para que las barras de desplazamiento aparezcan, o definir la propiedad `vScrollPolicy` en `true`.

7 Seleccione Control > Probar película.

También puede definir los aspectos de un subcomponente para todos los componentes de un documento definiendo la propiedad del aspecto en el objeto `prototipo` del subcomponente en la sección `#initclip` del símbolo del aspecto. Para más información sobre el objeto `prototipo`, consulte `Function.prototype` en el apartado Diccionario de ActionScript de la Ayuda.

Para utilizar `#initclip` para aplicar un aspecto editado a todos los componentes de un documento, siga este procedimiento:

- 1 Siga los pasos indicados en [“Edición de aspectos de componentes” en la página 38](#), pero edite un aspecto de la barra de desplazamiento. En este ejemplo, edite el aspecto `ScrollDownArrowDown` y asígnele el nombre nuevo `MyScrollDownArrowDown`.
- 2 Seleccione Archivo > Nuevo para crear un documento de Flash nuevo. Guárdelo con un nombre exclusivo, por ejemplo, `SkinsInitExample fla`.
- 3 Seleccione el símbolo `MyScrollDownArrowDown` en la biblioteca del ejemplo de la biblioteca del tema editado, arrástrelo al escenario de `SkinsInitExample fla` y elimínelo.
Esto añade el símbolo a la biblioteca sin hacerlo visible en el escenario.
- 4 Seleccione `MyScrollDownArrowDown` en la biblioteca `SkinsInitExample fla` y seleccione Vinculación en el menú Opciones.
- 5 Active la casilla de verificación Exportar para ActionScript. Haga clic en Aceptar.
Se seleccionará automáticamente Exportar en primer fotograma.
- 6 Haga doble clic en `MyScrollDownArrowDown` de la biblioteca para abrirlo en el modo Editar símbolo.
- 7 Introduzca el código siguiente en el fotograma 1 del símbolo `MyScrollDownArrowDown`:

```
#initclip 10
import mx.controls.scrollClasses.ScrollBar;
ScrollBar.prototype.downArrowDownName = "MyScrollDownArrowDown";
#endinitclip
```

- 8 Para añadir un componente List al documento, siga uno de estos procedimientos:
 - Arrastre un componente List desde el panel Componentes al escenario. Introduzca suficientes parámetros de Label para que aparezca la barra de desplazamiento vertical.
 - Arrastre un componente List desde el panel Componentes al escenario y elimínelo. Introduzca el código siguiente en el fotograma 1 de la línea de tiempo principal de `SkinsInitExample fla`:

```
createClassObject(mx.controls.List, "myListBox1", 0, {dataProvider:
    ["AL", "AR", "AZ", "CA", "HI", "ID", "KA", "LA", "MA"]});
```

Nota: añada información suficiente para que aparezca la barra de desplazamiento vertical, o defina `vScrollPolicy` en `true`.

En el siguiente ejemplo se explica cómo aplicar aspectos a algo que ya está en el escenario. En este ejemplo sólo se aplican aspectos a los componentes List; no se aplican aspectos a las barras de desplazamiento TextArea o ScrollPane.

Para utilizar `#initclip` para aplicar un aspecto editado a componentes específicos de un documento, siga este procedimiento:

- 1 Siga los pasos indicados en [“Edición de aspectos de componentes” en la página 38](#), pero edite un aspecto de la barra de desplazamiento. En este ejemplo, edite el aspecto `ScrollDownArrowDown` y asígnele el nombre nuevo `MyScrollDownArrowDown`.

- 2 Seleccione Archivo > Nuevo y cree un documento de Flash.
- 3 Seleccione Archivo > Guardar y asigne al archivo un nombre exclusivo como **MyVScrollTest.fla**.
- 4 Arrastre MyScrollDownArrowDown desde la biblioteca del tema a la biblioteca MyVScrollTest.fla.
- 5 Seleccione Insertar > Nuevo símbolo y asígnele un nombre exclusivo como **MyVScrollBar**.
- 6 Active la casilla de verificación Exportar para ActionScript. Haga clic en Aceptar.
Se seleccionará automáticamente Exportar en primer fotograma.
- 7 Introduzca el código siguiente en el fotograma 1 del símbolo MyScrollBar:


```
#initclip 10
import MyVScrollBar
Object.registerClass("VScrollBar", MyVScrollBar);
#endinitclip
```
- 8 Arrastre un componente List desde el panel Componentes al escenario.
- 9 En el inspector de propiedades, introduzca todos los parámetros de Label necesarios para que aparezca la barra de desplazamiento vertical.
- 10 Seleccione Archivo > Guardar.
- 11 Seleccione Archivo > Nuevo y cree un archivo de ActionScript nuevo.
- 12 Introduzca el código siguiente:


```
import mx.controls.VScrollBar
import mx.controls.List
class MyVScrollBar extends VScrollBar{
    function init():Void{
        if (_parent instanceof List){
            downArrowDownName = "MyScrollDownArrowDown";
        }
        super.init();
    }
}
```
- 13 Seleccione Archivo > Guardar y guarde este archivo como **MyVScrollBar.as**.
- 14 Haga clic en un área vacía del escenario y, en el inspector de propiedades, seleccione el botón Configuración de publicación.
- 15 Seleccione el botón Configuración de la versión ActionScript.
- 16 Haga clic en el botón Más para añadir una nueva ruta de clase y seleccione el botón Destino para examinar la ubicación del archivo MyComboBox.as en el disco duro.
- 17 Seleccione Control > Probar película.

Cambio de propiedades de aspecto en el prototipo

Si un componente no admite directamente variables del aspecto, puede definir el componente como subclase y sustituir sus aspectos. Por ejemplo, el componente ComboBox no admite directamente la aplicación de aspectos a su lista desplegable, ya que utiliza un componente List como lista desplegable.

Si un componente está formado por subcomponentes, dichos subcomponentes se identifican en la entrada del componente en el [Capítulo 4, “Diccionario de componentes”, en la página 45](#).

Para aplicar aspectos a un subcomponente, siga este procedimiento:

- 1 Siga los pasos indicados en [“Edición de aspectos de componentes” en la página 38](#), pero edite un aspecto de la barra de desplazamiento. En este ejemplo, edite el aspecto `ScrollDownArrowDown` y asígnele el nombre nuevo **MyScrollDownArrowDown**.
- 2 Seleccione Archivo > Nuevo y cree un documento de Flash.
- 3 Seleccione Archivo > Guardar y asigne al archivo un nombre exclusivo como **MyComboTest.fla**.
- 4 Arrastre `MyScrollDownArrowDown` desde la biblioteca del tema hasta el escenario de `MyComboTest.fla` y elimínelo.
Esto añade el símbolo a la biblioteca, pero no lo hace visible en el escenario.
- 5 Seleccione Insertar > Nuevo símbolo y asígnele un nombre exclusivo como **MyComboBox**.
- 6 Seleccione la casilla de verificación Exportar para ActionScript y haga clic en Aceptar.
Se seleccionará automáticamente Exportar en primer fotograma.
- 7 Introduzca el código siguiente en el panel Acciones en las acciones del fotograma 1 de `MyComboBox`:

```
#initclip 10
import MyComboBox
Object.registerClass("ComboBox", MyComboBox);
#endinitclip
```
- 8 Arrastre un componente `ComboBox` al escenario.
- 9 En el inspector de propiedades, introduzca todos los parámetros de Label necesarios para que aparezca la barra de desplazamiento vertical.
- 10 Seleccione Archivo > Guardar.
- 11 Seleccione Archivo > Nuevo y cree un archivo de ActionScript nuevo (sólo en Flash Professional).
- 12 Introduzca el código siguiente:

```
import mx.controls.ComboBox
import mx.controls.scrollClasses.ScrollBar
class MyComboBox extends ComboBox{
    function getDropdown():Object{
        var oldName = ScrollBar.prototype.downArrowDownName;
        ScrollBar.prototype.downArrowDownName = "MyScrollDownArrowDown";
        var r = super.getDropdown();
        ScrollBar.prototype.downArrowDownName = oldName;
        return r;
    }
}
```
- 13 Seleccione Archivo > Guardar y guarde este archivo como **MyComboBox.as**.
- 14 Haga clic en un área vacía del escenario y, en el inspector de propiedades, seleccione el botón Configuración de publicación.
- 15 Seleccione el botón Configuración de la versión ActionScript.
- 16 Haga clic en el botón Más para añadir una nueva ruta de clase y seleccione el botón Destino para examinar la ubicación del archivo `MyComboBox.as` en el disco duro.
- 17 Seleccione Control > Probar película.

CAPÍTULO 4

Diccionario de componentes

En este capítulo se describen los componentes y la interfaz API (interfaz de programación de aplicaciones) de cada uno de ellos.

La descripción de cada componente contiene la información siguiente:

- Interacción con el teclado
- Vista previa dinámica
- Accesibilidad
- Configuración de los parámetros del componente
- Utilización del componente en una aplicación
- Personalización del componente con estilos y aspectos
- Métodos, propiedades y eventos de `JavaScript`

Los componentes se presentan en orden alfabético, pero en las tablas siguientes podrá encontrarlos por categoría:

Componentes de la interfaz de usuario (IU)

Componente	Descripción
Componente Accordion	Conjunto de vistas verticales superpuestas con botones en la parte superior que permiten a los usuarios cambiar de vista.
Componente Alert	Ventana que presenta al usuario una pregunta y botones para elegir la respuesta.
Componente Button	Botón que puede cambiarse de tamaño y personalizarse con un icono personalizado.
Componente CheckBox	Permite hacer una elección booleana (<code>true</code> o <code>false</code>).
Componente ComboBox	Permite seleccionar una opción en una lista de desplazamiento de opciones. En ocasiones, este componente contiene un campo de texto editable en la parte superior de la lista para realizar búsquedas.
Componente DateChooser	Permite a los usuarios elegir una o varias fechas en un calendario.

Componente	Descripción
Componente DateField	Campo de texto no editable con un icono de calendario. Si el usuario hace clic en cualquier punto del recuadro de delimitación, aparece un componente DateChooser.
Componente DataGrid	Permite a los usuarios mostrar y manipular varias columnas de datos.
Componente Label	Campo de texto no editable de una línea.
Componente List	Permite seleccionar una o varias opciones en una lista de desplazamiento.
Componente Loader	Contenedor de un archivo SWF o JPEG cargado.
Componente Menu	Permite seleccionar un comando en una lista; menú de aplicación estándar del escritorio.
Componente MenuBar	Barra horizontal de menús. Permite que los menús actúen como un grupo, de manera que pueda controlar entradas de ratón y de teclado.
Componente NumericStepper	Flechas para aumentar o reducir una cifra con el ratón.
Componente ProgressBar	Muestra el progreso de un proceso, normalmente de carga.
Componente RadioButton	Permite elegir entre opciones que se excluyen entre sí.
Componente ScrollPane	Muestra películas, mapas de bits y archivos SWF en un área limitada con barras de desplazamiento automático.
Componente TextArea	Campo de texto de varias líneas opcionalmente editable.
Componente TextInput	Campo de texto de una línea opcionalmente editable.
Componente Tree	Permite manipular información jerárquica.
Componente Window	Ventana desplazable con barra de título, texto, borde y botón de cierre que muestra contenido al usuario.

Componentes de medios

Componente	Descripción
Componente MediaController	Controla la reproducción de flujo de medios en una aplicación.
Componente MediaDisplay	Muestra los flujos de medios en una aplicación
Componente MediaPlayer	Combinación de componentes MediaDisplay y MediaController.

Componentes de datos

Componente	Descripción
Paquete DataBinding	Estas clases implementan la funcionalidad de vinculación de datos en tiempo de ejecución de Flash.
Componente DataHolder	Almacena información y puede utilizarse como conector entre componentes.
Componente DataProvider	Este componente es el modelo para las listas de datos de acceso lineal. Este modelo proporciona capacidades sencillas de manipulación de matrices que difunden sus cambios.
Componente DataSet	Bloque para crear aplicaciones gestionadas por datos.
Componente RDBMSResolver	Permite guardar datos en cualquier fuente de datos admitida. Este componente resolver convierte el XML que se puede recibir y analizar mediante un servicio Web, JavaBean, servlet o página ASP.
Componente WebServiceConnector	Proporciona acceso sin script a llamadas de método de servicio Web.
Componente XMLConnector	Lee y graba documentos XML con los métodos HTTP GET y POST.
Componente XUpdateResolver	Permite guardar datos en cualquier fuente de datos admitida. Este componente resolver convierte DeltaPacket a XUpdate.

Administradores

Componente	Descripción
Clase DepthManager	Administra la profundidad de apilamiento de los objetos.
Clase FocusManager	Gestiona el desplazamiento entre componentes de la pantalla con el tabulador. También maneja los cambios de selección cada vez que el usuario hace clic en la aplicación.
Clase PopUpManager	Permite crear y eliminar ventanas emergentes.
Clase StyleManager	Permite registrar estilos y administra estilos heredados.

Pantallas

Componente	Descripción
Clase Slide	Permite manipular las pantallas de presentación de diapositivas en tiempo de ejecución.
Clase Form	Permite manipular las pantallas de aplicaciones de formularios en tiempo de ejecución.

Componente Accordion

Para ver la información más reciente sobre esta función, haga clic en el botón Actualizar que se encuentra en la parte superior de la ficha Ayuda.

Componente Alert

Para ver la información más reciente sobre esta función, haga clic en el botón Actualizar que se encuentra en la parte superior de la ficha Ayuda.

Componente Button

El componente Button es un botón rectangular de la interfaz de usuario que puede cambiarse de tamaño. También se le puede añadir un icono personalizado y cambiar su comportamiento de botón de comando a conmutador. El conmutador permanece presionado cuando se hace clic en él y recupera su estado original al repetirse el clic.

Un botón puede estar activado o desactivado en una aplicación. Si está desactivado, el botón no recibe la entrada del ratón ni del teclado. Si está activado, el botón se selecciona cuando el usuario hace clic sobre él o presiona el tabulador hasta su posición. Cuando una instancia de Button está seleccionada, es posible utilizar las siguientes teclas para controlarla:

Tecla	Descripción
Mayúsculas + Tabulador	Desplaza la selección al objeto anterior.
Barra espaciadora	Presiona o libera el componente y activa el evento <code>click</code> .
Tabulador	Desplaza la selección al objeto siguiente.

Para más información sobre el control de la selección, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 24](#) o [“Clase FocusManager” en la página 101](#).

La vista previa dinámica de cada instancia de Button refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o el panel Inspector de componentes. Sin embargo, en la vista previa dinámica los iconos personalizados se representan en el escenario con un cuadrado gris.

Cuando se añade el componente Button a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de pantalla puedan acceder al mismo. En primer lugar, debe añadir la línea de código siguiente para activar la accesibilidad del componente Button:

```
mx.accessibility.ButtonAccImpl.enableAccessibility();
```

La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias. Para más información, consulte [“Creación de contenido accesible” en el apartado Utilización de Flash de la Ayuda](#). Es posible que tenga que actualizar el sistema de Ayuda para consultar esta información.

Utilización del componente Button

Un botón es una parte fundamental de cualquier formulario o aplicación Web. Utilice botones siempre que necesite que el usuario inicie un evento. Por ejemplo, la mayoría de los formularios contienen el botón “Enviar” y las presentaciones suelen llevar los botones “Anterior” y “Siguiente”.

Para añadir un icono a un botón, es necesario seleccionar o crear un clip de película o un símbolo gráfico para utilizarlo como icono. El símbolo debe registrarse en la posición 0, 0 para que el botón tenga el diseño adecuado. Seleccione el símbolo del icono en el panel Biblioteca, abra el cuadro de diálogo Vinculación desde el menú Opciones e introduzca un identificador de vinculación. Éste es el valor que debe introducir como parámetro icon en el inspector de propiedades o el panel Inspector de componentes. También puede introducirlo en la propiedad [Button.icon](#) de ActionScript.

Nota: si el icono es más grande que el botón, sobresaldrá por los bordes de éste.

Parámetros de Button

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente Button en el inspector de propiedades o el panel Inspector de componentes:

label define el valor del texto del botón; el valor predeterminado es Button.

icon añade un icono personalizado al botón. El valor es el identificador de vinculación de un clip de película o símbolo gráfico de la biblioteca; no hay valor predeterminado.

toggle convierte el botón en un conmutador. Si el valor es true, el botón permanece en el estado Presionado cuando está presionado y recupera el estado Arriba cuando se vuelve a presionar. Si el valor es false, el botón se comporta como un botón de comando normal; el valor predeterminado es false.

selected especifica si el botón está presionado (true) o no (false) cuando el parámetro toggle es true. El valor predeterminado es false.

labelPlacement orienta el texto de la etiqueta del botón con respecto al icono. Este parámetro puede tener uno de estos cuatro valores: left, right, top o bottom; el valor predeterminado es right. Para más información, consulte [Button.labelPlacement](#).

Puede escribir código ActionScript para controlar éstas y otras opciones de componentes Button con sus propiedades, métodos y eventos. Para más información, consulte [Clase Button](#).

Creación de aplicaciones con el componente Button

El siguiente procedimiento explica cómo añadir un componente Button a una aplicación durante la edición. En este ejemplo se trata de un botón de Ayuda con un icono personalizado que abre el sistema de Ayuda cuando el usuario lo presiona.

Para crear una aplicación con el componente Window, siga este procedimiento:

- 1 Arrastre un componente Button desde el panel Componentes al escenario.
- 2 En el inspector de propiedades, introduzca el nombre de instancia **helpBtn**.
- 3 En el inspector de propiedades, siga este procedimiento:

- Introduzca **Help** en el parámetro label.
- Introduzca **HelpIcon** en el parámetro icon.

Para utilizar un icono, en la biblioteca debe haber un clip de película o un símbolo gráfico con un identificador de vinculación para emplearlo como parámetro icon. En este ejemplo, el identificador de vinculación es HelpIcon.

- Defina la propiedad toggle en true.

- 4 Seleccione el fotograma 1 de la línea de tiempo, abra el panel Acciones e introduzca el código siguiente:

```
clippyListener = new Object();
clippyListener.click = function (evt){
    clippyHelper.enabled = evt.target.selected;
}
helpBtn.addEventListener("click", clippyListener);
```

La última línea de código añade un controlador de eventos `click` a la instancia `helpBtn`. El controlador activa y desactiva la instancia `clippyHelper`, que podría ser un panel Ayuda de alguna clase.

Personalización del componente Button

El componente Button puede transformarse horizontal y verticalmente durante la edición o en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `setSize()` (véase [UIObject.setSize\(\)](#)) o cualquier método o propiedad aplicable de la clase Button (véase [Clase Button](#)). Cuando se modifica el tamaño del botón, no cambia el tamaño del icono ni de la etiqueta.

El recuadro de delimitación de una instancia de Button es invisible y designa el área activa de la instancia. Si se aumenta el tamaño de la instancia, también aumenta el tamaño del área activa. Si el recuadro de delimitación es demasiado pequeño para que encaje la etiqueta, ésta se recorta para ajustarse al espacio disponible.

Si el icono es más grande que el botón, sobresaldrá por los bordes de éste.

Utilización de estilos con el componente Button

Es posible definir propiedades de estilo para cambiar la apariencia de una instancia de Button. Si el nombre de una propiedad de estilo termina por “Color”, significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 27](#).

El componente Button admite los siguientes estilos de Halo:

Estilo	Descripción
themeColor	Fondo de un componente. Es el único estilo de color que no hereda su valor. Los valores posibles son "haloGreen", "haloBlue" y "haloOrange".
color	Texto de la etiqueta de un componente.
disabledColor	Color desactivado para el texto.
fontFamily	Nombre de la fuente del texto.
fontSize	Tamaño de la fuente en puntos.
fontStyle	Estilo de la fuente: puede ser "normal" o "italic".
fontWeight	Grosor de la fuente: puede ser "normal" o "bold".

Utilización de aspectos con el componente Button

El componente Button utiliza la interfaz API de dibujo de ActionScript para dibujar estados de botón. Para aplicar aspectos al componente Button durante la edición, modifique el código ActionScript del archivo ButtonSkin.as que se encuentra en la carpeta First Run\Classes\mx\skins\halo.

Si utiliza el método `UIObject.createClassObject()` para crear dinámicamente una instancia del componente Button (en tiempo de ejecución), podrá asignarle aspectos dinámicamente. Para asignar aspectos a un componente en tiempo de ejecución, defina las propiedades de aspecto en el parámetro `initObject` que pasa al método `createClassObject()`. Estas propiedades de aspecto definen los nombres de los símbolos que se utilizarán como estados del botón, ya sea con o sin icono.

Si define el parámetro `icon` durante la edición o la propiedad `icon` de ActionScript en tiempo de ejecución, se asignará el mismo identificador de vinculación a tres estados del icono: `falseUpIcon`, `falseDownIcon` y `trueUpIcon`. Si desea designar un icono exclusivo para alguno de los ocho estados del icono (por ejemplo, para que aparezca un icono diferente cuando el usuario presione el botón), debe definir las propiedades del parámetro `initObject` que pasa al método `createClassObject()`.

El código siguiente crea un objeto denominado `initObject` que se utilizará como parámetro `initObject` y define las propiedades de aspecto en identificadores de vinculación de símbolos nuevos. La última línea de código llama al método `createClassObject()` para crear una instancia nueva de la clase Button con las propiedades pasadas en el parámetro `initObject`:

```
var initObject = new Object();
initObject.falseUpIcon = "MyFalseUpIcon";
initObject.falseDownIcon = "MyFalseDownIcon";
initObject.trueUpIcon = "MyTrueUpIcon";
createClassObject(mx.controls.Button, "ButtonInstance", 0, initObject);
```

Para más información, consulte [“Aplicación de aspectos a los componentes” en la página 37](#) y `UIObject.createClassObject()`.

Si un botón está activado, muestra su estado Sobre cuando el puntero pasa sobre él. El botón recibe la selección de entrada y muestra el estado Presionado cuando se hace clic en él. El botón recupera el estado Sobre cuando se suelta el ratón. Si el puntero se retira del botón estando el ratón presionado, el botón recupera su estado original y conserva la selección de entrada. Si el parámetro `toggle` está definido en `true`, el estado del botón no cambia hasta que el ratón se suelta estando sobre él.

Si el botón está desactivado, muestra el estado Desactivado sea cual sea la acción del usuario.

El componente Button utiliza las siguientes propiedades de aspecto:

Propiedad	Descripción
<code>falseUpSkin</code>	Estado Arriba. El valor predeterminado es <code>RectBorder</code> .
<code>falseDownSkin</code>	Estado Presionado. El valor predeterminado es <code>RectBorder</code> .
<code>falseOverSkin</code>	Estado Sobre. El valor predeterminado es <code>RectBorder</code> .
<code>falseDisabledSkin</code>	Estado Desactivado. El valor predeterminado es <code>RectBorder</code> .
<code>trueUpSkin</code>	Estado Conmutado. El valor predeterminado es <code>RectBorder</code> .

Propiedad	Descripción
<code>trueDownSkin</code>	Estado Presionado conmutado. El valor predeterminado es <code>RectBorder</code> .
<code>trueOverSkin</code>	Estado Sobre conmutado. El valor predeterminado es <code>RectBorder</code> .
<code>trueDisabledSkin</code>	Estado Desactivado conmutado. El valor predeterminado es <code>RectBorder</code> .
<code>falseUpIcon</code>	Estado Arriba del icono. El valor predeterminado es <code>undefined</code> .
<code>falseDownIcon</code>	Estado Presionado del icono. El valor predeterminado es <code>undefined</code> .
<code>falseOverIcon</code>	Estado Sobre del icono. El valor predeterminado es <code>undefined</code> .
<code>falseDisabledIcon</code>	Estado Desactivado del icono. El valor predeterminado es <code>undefined</code> .
<code>trueUpIcon</code>	Estado Conmutado del icono. El valor predeterminado es <code>undefined</code> .
<code>trueOverIcon</code>	Estado Sobre conmutado del icono. El valor predeterminado es <code>undefined</code> .
<code>trueDownIcon</code>	Estado Presionado conmutado del icono. El valor predeterminado es <code>undefined</code> .
<code>trueDisabledIcon</code>	Estado Desactivado conmutado del icono. El valor predeterminado es <code>undefined</code> .

Clase Button

Herencia `UIObject > UIComponent > SimpleButton > Button`

Namespace de clase de ActionScript `mx.controls.Button`

Las propiedades de la clase `Button` permiten añadir un icono a un botón, crear una etiqueta de texto o indicar si el botón actúa como botón de comando o como conmutador en tiempo de ejecución.

Si una propiedad de la clase `Button` se define con `ActionScript`, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o el panel Inspector de componentes.

El componente `Button` utiliza `FocusManager` para sustituir el rectángulo de selección predeterminado de `Flash Player` por un rectángulo de selección personalizado con esquinas redondeadas. Para más información, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 24](#).

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.controls.Button.version);
```

Nota: el código siguiente devuelve `undefined`: `trace(myButtonInstance.version);`.

La clase del componente `Button` es distinta del objeto `Button` incorporado de `ActionScript`.

Resumen de métodos de la clase Button

Hereda todos los métodos de [Clase UIObject](#) y [Clase UIComponent](#).

Resumen de propiedades de la clase Button

Método	Descripción
<code>SimpleButton.emphasized</code>	Indica si el botón tiene el aspecto de un botón de comando predeterminado.
<code>SimpleButton.emphasizedStyleDeclaration</code>	Declaración de estilos cuando la propiedad <code>emphasized</code> está definida en <code>true</code> .
<code>Button.icon</code>	Especifica un icono para una instancia del botón.
<code>Button.label</code>	Especifica el texto que aparece dentro de un botón.
<code>Button.labelPlacement</code>	Especifica la orientación del texto de la etiqueta con respecto a un icono.
<code>Button.selected</code>	Cuando la propiedad <code>toggle</code> es <code>true</code> , especifica si el botón está presionado (<code>true</code>) o no (<code>false</code>).
<code>Button.toggle</code>	Indica si el botón se comporta como un conmutador.

Hereda todas las propiedades de [Clase UIObject](#) y [Clase UIComponent](#).

Resumen de eventos de la clase Button

Método	Descripción
<code>Button.click</code>	Se difunde cuando se presiona el ratón sobre una instancia del botón o cuando se presiona la barra espaciadora.

Hereda todos los eventos de [Clase UIObject](#) y [Clase UIComponent](#).

Button.click

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(click){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.click = function(eventObject){  
    ...  
}  
buttonInstance.addEventListener("click", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se hace clic con el ratón (se suelta el botón del ratón) en un botón o cuando éste está seleccionado y se presiona la barra espaciadora.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente `Button`. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado a la instancia del componente `Button` `myButtonComponent`, envía “_level0.myButtonComponent” al panel Salida:

```
on(click){  
    trace(this);  
}
```

Observe que `this` se comporta de forma diferente cuando se utiliza dentro de un controlador `on()` asociado a un símbolo de botón normal de Flash. Cuando `this` se utiliza dentro de un controlador `on()` asociado a un símbolo de botón, hace referencia a la línea de tiempo que contiene el botón. Por ejemplo, el código siguiente, asociado a la instancia `myButton` del símbolo de botón, envía “_level0” al panel Salida:

```
on(release) {  
    trace(this);  
}
```

Nota: el objeto incorporado `Button` de `ActionScript` no tiene eventos `click`; el evento más próximo es `release`.

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*buttonInstance*) distribuye un evento (en este caso `click`) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (*eventObject*) al método del objeto detector. El objeto `Event` tiene un conjunto de propiedades que contiene información acerca del mismo. Estas propiedades sirven para escribir el código que gestiona el evento.

Finalmente, se llama al método `addEventListener()` (véase [UIEventDispatcher.addEventListener\(\)](#)) de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 236](#).

Ejemplo

En este ejemplo, escrito en un fotograma de la línea de tiempo, se envía un mensaje al panel Salida cuando se hace clic en un botón denominado `buttonInstance`. La primera línea del código etiqueta el botón. La segunda línea especifica que el botón actúe como conmutador. La tercera línea crea un objeto detector denominado `form`. La cuarta línea define una función para el evento `click` en el objeto detector. Dentro de la función hay una acción `trace` que utiliza el objeto `Event` pasado automáticamente a la función (en este ejemplo `eventObj`) para generar un mensaje. La propiedad `target` de un objeto `Event` es el componente que ha generado el evento (en este ejemplo `buttonInstance`). A la propiedad `Button.selected` se accede desde la propiedad `target` del objeto `Event`. La última línea llama al método `addEventListener()` desde `buttonInstance` y pasa como parámetros el evento `click` y el objeto detector `form`, como en el ejemplo siguiente:

```
buttonInstance.label = "Click Test"
buttonInstance.toggle = true;
form = new Object();
form.click = function(eventObj){
    trace("La propiedad seleccionada ha cambiado a " +
        eventObj.target.selected);
}
buttonInstance.addEventListener("click", form);
```

El código siguiente también envía un mensaje al panel Salida cuando se hace clic en `buttonInstance`. El controlador `on()` debe asociarse directamente a `buttonInstance`, como en el ejemplo siguiente:

```
on(click){
    trace("clic en componente de botón");
}
```

Véase también

[UIEventDispatcher.addEventListener\(\)](#)

SimpleButton.emphasized

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

buttonInstance.emphasized

Descripción

Propiedad; indica si el botón está en estado `emphasized` (`true`) o no (`false`). El estado `emphasized` equivale a la apariencia de un botón de comando predeterminado. En general, utilice la propiedad [FocusManager.defaultPushButton](#) en lugar de definir directamente la propiedad `emphasized`. El valor predeterminado es `false`.

La propiedad `emphasized` es una propiedad estática de la clase `SimpleButton`. Por tanto, debe acceder a ella directamente desde `SimpleButton`, como en el ejemplo siguiente:

```
SimpleButton.emphasizedStyleDeclaration = "foo";
```

Si está utilizando `FocusManager.defaultPushButton`, es posible que sólo desee definir un botón en el estado `emphasized`, o utilizar el estado `emphasized` para cambiar el texto de un color a otro. En el siguiente ejemplo se define la propiedad `emphasized` de la instancia del botón `myButton`:

```
_global.styles.foo = new CSSStyleDeclaration();
_global.styles.foo.color = 0xFF0000;
SimpleButton.emphasizedStyleDeclaration = "foo";
myButton.emphasized = true;
```

Véase también

[SimpleButton.emphasizedStyleDeclaration](#)

SimpleButton.emphasizedStyleDeclaration

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

buttonInstance.emphasizedStyleDeclaration

Descripción

Propiedad; cadena que indica la declaración de estilos que da formato a un botón cuando la propiedad `emphasized` se define en `true`.

Véase también

[Window.titleStyleDeclaration](#), [SimpleButton.emphasized](#)

Button.icon

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

buttonInstance.icon

Descripción

Propiedad; cadena que especifica el identificador de vinculación de un símbolo de la biblioteca que se utilizará como icono en una instancia del botón. El icono puede ser un símbolo de clip de película o un símbolo gráfico con un punto de registro en la esquina superior izquierda. Si el icono es demasiado grande para encajarlo en el botón, deberá modificar el tamaño del botón; ni el botón ni el icono cambian de tamaño automáticamente. Si el icono es más grande que el botón, sobresaldrá por los bordes del botón.

Para crear un icono personalizado, cree un clip de película o un símbolo gráfico. Seleccione el símbolo en el escenario en modo Editar símbolos e introduzca 0 en los cuadros X e Y del inspector de propiedades. En el panel Biblioteca, seleccione el clip de película y elija Vinculación en el menú Opciones. Seleccione Exportar para ActionScript e introduzca un identificador en el cuadro de texto Identificador.

El valor predeterminado es una cadena vacía (" ") que indica que no hay icono.

Utilice la propiedad `labelPlacement` para definir la posición del icono con respecto al botón.

Ejemplo

El código siguiente asigna el clip de película del panel Biblioteca que tiene el identificador de vinculación `happiness` a la instancia `Button` como icono:

```
myButton.icon = "happiness"
```


Véase también

[Button.labelPlacement](#)

Button.label

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

buttonInstance.label

Descripción

Propiedad; especifica la etiqueta de texto para una instancia del botón. De forma predeterminada, la etiqueta aparece centrada en el botón. Cuando se llama este método, se sustituye el parámetro de edición de etiquetas especificado en el inspector de propiedades o el panel Inspector de componentes. El valor predeterminado es "Button".

Ejemplo

El código siguiente define la etiqueta en "Eliminar de la lista":

```
buttonInstance.label = "Eliminar de la lista";
```

Véase también

[Button.labelPlacement](#)

Button.labelPlacement

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

buttonInstance.labelPlacement

Descripción

Propiedad; define la posición de la etiqueta con respecto al icono. El valor predeterminado es "right". Los siguientes son los cuatro valores posibles; el icono y la etiqueta están siempre centrados vertical y horizontalmente en el área de delimitación del botón:

- "right" La etiqueta aparece a la derecha del icono.
- "left" La etiqueta aparece a la izquierda del icono.
- "bottom" La etiqueta aparece debajo del icono.
- "top" La etiqueta aparece encima del icono.

Ejemplo

El código siguiente coloca la etiqueta a la izquierda del icono. La segunda línea del código envía el valor de la propiedad `labelPlacement` al panel Salida:

```
iconInstance.labelPlacement = "left";  
trace(iconInstance.labelPlacement);
```

Button.selected

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

buttonInstance.selected

Descripción

Propiedad; valor booleano que especifica si un botón está presionado (`true`) o no (`false`). El valor de la propiedad `toggle` debe ser `true` para definir la propiedad `selected` en `true`. Si la propiedad `toggle` es `false`, asignar el valor `true` a la propiedad `selected` no tiene efecto alguno. El valor predeterminado es `false`.

El evento `click` no se activa cuando el valor de la propiedad `selected` cambia con `ActionScript`. Se activa cuando el usuario interactúa con el botón.

Ejemplo

En el siguiente ejemplo, la propiedad `toggle` está definida en `true` y la propiedad `selected` también está definida en `true`, lo que coloca el botón en estado Presionado. La acción `trace` envía el valor `true` al panel Salida:

```
ButtonInstance.toggle = true; // toggle ha de ser true para poder definir la  
propiedad selected  
ButtonInstance.selected = true; //muestra el estado Conmutado del botón  
trace(ButtonInstance.selected); //acción trace envía el valor true
```

Véase también

[Button.toggle](#)

Button.toggle

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

buttonInstance.toggle

Descripción

Propiedad; valor booleano que especifica si un botón actúa como conmutador (`true`) o como botón de comando (`false`); el valor predeterminado es `false`. Cuando se presiona un conmutador, permanece en estado Presionado hasta que se vuelve a hacer clic sobre él.

Ejemplo

El código siguiente define la propiedad `toggle` en `true`, por lo que la instancia `myButton` se comporta como un conmutador:

```
myButton.toggle = true;
```

Interfaz CellRenderer

Para obtener información reciente sobre esta función, haga clic en el botón Actualizar en la parte superior de la ficha Ayuda.

Componente CheckBox

Una casilla de verificación es un cuadrado que puede seleccionarse o no seleccionarse. Cuando se selecciona, en el cuadro aparece una marca. A una casilla de verificación se le puede añadir una etiqueta de texto, que puede colocarse a la izquierda, derecha, arriba o abajo.

Una casilla de verificación puede estar activada o desactivada en una aplicación. Si está activada y el usuario hace clic en ella o en su etiqueta, la casilla recibe la selección de entrada y muestra su apariencia presionada. Si el usuario desplaza el puntero fuera del área de delimitación de la casilla o de su etiqueta mientras presiona el botón del ratón, el componente recupera su apariencia original y conserva la selección de entrada. El estado de una casilla de verificación no cambia hasta que se suelta el ratón sobre el componente. Además, una casilla de verificación tiene dos estados Desactivado (seleccionado y no seleccionado) que no permiten la interacción con el ratón ni el teclado.

Si la casilla de verificación está desactivada, muestra su apariencia desactivada sea cual sea la acción del usuario. Si está desactivado, el botón no recibe la entrada del ratón ni del teclado.

La instancia de `CheckBox` recibe la selección cuando el usuario hace clic sobre ella o presiona el tabulador hasta su posición. Cuando una instancia de `CheckBox` está seleccionada, se pueden utilizar las siguientes teclas para controlarla:

Tecla	Descripción
Mayúsculas + Tabulador	Desplaza la selección al elemento anterior.
Barra espaciadora	Selecciona o anula la selección del componente y activa el evento <code>click</code> .
Tabulador	Desplaza la selección al elemento siguiente.

Para más información sobre el control de la selección, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 24](#) o [“Clase FocusManager” en la página 101](#).

La vista previa dinámica de cada instancia de `CheckBox` refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o el panel Inspector de componentes.

Cuando se añade el componente `CheckBox` a una aplicación, se puede utilizar el panel Accesibilidad para que los lectores de la pantalla puedan acceder a él. En primer lugar, debe añadir la línea de código siguiente para activar la accesibilidad:

```
mx.accessibility.CheckBoxAccImpl.enableAccessibility();
```

La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias. Para más información, consulte “Creación de contenido accesible” en el apartado Utilización de Flash de la Ayuda. Es posible que tenga que actualizar el sistema de Ayuda para consultar esta información.

Utilización del componente `CheckBox`

Una casilla de verificación es una parte fundamental de cualquier formulario o aplicación Web. Utilice casillas de verificación siempre que necesite reunir un conjunto de valores `true` o `false` que no se excluyan mutuamente. Por ejemplo, un formulario que recopila información personal sobre un cliente podría contener una lista de aficiones que el cliente debe elegir; cada afición llevaría al lado una casilla de verificación.

Parámetros de `CheckBox`

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente `CheckBox` en el inspector de propiedades o el panel Inspector de componentes:

label define el valor del texto de la casilla de verificación; el valor predeterminado es `defaultValue`.

selected define el valor inicial de la casilla de verificación en marcada (`true`) o sin marcar (`false`).

labelPlacement orienta el texto de la etiqueta en la casilla de verificación. Este parámetro puede tener uno de estos cuatro valores: `left`, `right`, `top` o `bottom`; el valor predeterminado es `right`. Para más información, consulte [CheckBox.labelPlacement](#).

Puede escribir código `ActionScript` para controlar éstas y otras opciones adicionales de componentes `CheckBox` con sus propiedades, métodos y eventos. Para más información, consulte [Clase CheckBox](#).

Creación de aplicaciones con el componente `CheckBox`

El siguiente procedimiento explica cómo añadir un componente `CheckBox` a una aplicación durante la edición. El ejemplo siguiente es un formulario para una aplicación de citas en línea. El formulario es un cuestionario que busca citas posibles para el cliente. El formulario debe llevar una casilla de verificación con la etiqueta "Restrict Age" que permita al cliente limitar la búsqueda a un grupo de edad determinado. Si selecciona la casilla "Restrict Age", el cliente puede introducir la edad máxima y mínima en dos campos de texto que sólo se activan cuando se selecciona la casilla "Restrict Age".

Para crear una aplicación con el componente `CheckBox`, siga este procedimiento:

- 1 Arrastre dos componentes `TextInput` desde el panel Componentes al escenario.
- 2 En el inspector de propiedades, introduzca los nombres de instancia `minimumAge` y `maximumAge`.
- 3 Arrastre un componente `CheckBox` desde el panel Componentes al escenario.
- 4 En el inspector de propiedades, siga este procedimiento:
 - Introduzca **restrictAge** como nombre de instancia.
 - Introduzca **Restrict Age** como parámetro `label`.

- 5 Seleccione el fotograma 1 de la línea de tiempo, abra el panel Acciones e introduzca el código siguiente:

```
restrictAgeListener = new Object();
restrictAgeListener.click = function (evt){
    minimumAge.enabled = evt.target.selected;
    maximumAge.enabled = evt.target.selected;
}
restrictAge.addEventListener("click", restrictAgeListener);
```

Este código crea un controlador de eventos `click` que activa y desactiva los componentes de los campos de texto `minimumAge` y `maximumAge`, ya situados en el escenario. Para más información sobre el evento `click`, consulte [CheckBox.click](#). Para más información sobre el componente `TextInput`, consulte [“Componente TextInput” en la página 216](#).

Personalización del componente CheckBox

El componente `CheckBox` puede transformarse horizontal y verticalmente durante la edición o en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `setSize()` ([UIObject.setSize\(\)](#)) o cualquier método o propiedad aplicable de la clase `CheckBox` (véase [Clase CheckBox](#)). Cuando se modifica el tamaño de la casilla de verificación, no cambia el tamaño de la etiqueta ni del icono de la casilla; sólo cambia el tamaño del recuadro de delimitación.

El recuadro de delimitación de una instancia de `CheckBox` es invisible y designa el área activa de la instancia. Si se aumenta el tamaño de la instancia, también aumenta el tamaño del área activa. Si el recuadro de delimitación es demasiado pequeño para que encaje la etiqueta, ésta se recorta para ajustarse al espacio disponible.

Utilización de estilos con el componente CheckBox

Es posible definir propiedades de estilo para cambiar la apariencia de una instancia de `CheckBox`. Si el nombre de una propiedad de estilo termina por “Color”, significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 27](#).

El componente `CheckBox` admite los siguientes estilos de Halo:

Estilo	Descripción
<code>themeColor</code>	Fondo de un componente. Es el único estilo de color que no hereda su valor. Los valores posibles son “haloGreen”, “haloBlue” y “haloOrange”.
<code>color</code>	Texto de la etiqueta de un componente.
<code>disabledColor</code>	Color desactivado para el texto.
<code>fontFamily</code>	Nombre de la fuente del texto.
<code>fontSize</code>	Tamaño de la fuente en puntos.
<code>fontStyle</code>	Estilo de la fuente: puede ser “normal” o “italic”.
<code>fontWeight</code>	Grosor de la fuente: puede ser “normal” o “bold”.
<code>textDecoration</code>	Decoración del texto: puede ser “none” o “underline”.

Utilización de aspectos con el componente CheckBox

El componente CheckBox utiliza símbolos del panel Biblioteca para representar estados de botón. Para aplicar aspectos al componente CheckBox durante la edición, modifique los símbolos del panel Biblioteca. Los aspectos del componente CheckBox se encuentran en la carpeta Flash UI Components 2/Themes/MMDefault/CheckBox Assets/states de la biblioteca del archivo HaloTheme.fla o SampleTheme.fla. Para más información, consulte [“Aplicación de aspectos a los componentes” en la página 37](#).

El componente CheckBox utiliza las siguientes propiedades de aspecto:

Propiedad	Descripción
falseUpSkin	Estado Arriba. El valor predeterminado es RectBorder.
falseDownSkin	Estado Presionado. El valor predeterminado es RectBorder.
falseOverSkin	Estado Sobre. El valor predeterminado es RectBorder.
falseDisabledSkin	Estado Desactivado. El valor predeterminado es RectBorder.
trueUpSkin	Estado Conmutado. El valor predeterminado es RectBorder.
trueDownSkin	Estado Presionado conmutado. El valor predeterminado es RectBorder.
trueOverSkin	Estado Sobre conmutado. El valor predeterminado es RectBorder.
trueDisabledSkin	Estado Desactivado conmutado. El valor predeterminado es RectBorder.

Clase CheckBox

Herencia UIObject > UIComponent > SimpleButton > Button > CheckBox

Namespace de clase de ActionScript mx.controls.CheckBox

Las propiedades de la clase CheckBox permiten crear etiquetas de texto y colocarlas a la izquierda, derecha, encima o debajo de una casilla de verificación en tiempo de ejecución.

Si una propiedad de la clase CheckBox se define con ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o el panel Inspector de componentes.

El componente CheckBox utiliza FocusManager para sustituir el rectángulo de selección predeterminado de Flash Player por un rectángulo de selección personalizado con esquinas redondeadas. Para más información, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 24](#).

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.controls.CheckBox.version);
```

Nota: el código siguiente devuelve undefined: `trace(myCheckBoxInstance.version);`.

Resumen de propiedades de la clase `CheckBox`

Propiedad	Descripción
<code>CheckBox.label</code>	Especifica el texto que aparece junto a una casilla de verificación.
<code>CheckBox.labelPlacement</code>	Especifica la orientación del texto de la etiqueta con respecto a la casilla de verificación.
<code>CheckBox.selected</code>	Especifica si la casilla de verificación está seleccionada (<code>true</code>) o no (<code>false</code>).

Hereda todas las propiedades de [Clase `UIObject`](#) y [Clase `UIComponent`](#).

Resumen de métodos de la clase `CheckBox`

Hereda todos los métodos de [Clase `UIObject`](#) y [Clase `UIComponent`](#).

Resumen de eventos de la clase `CheckBox`

Evento	Descripción
<code>CheckBox.click</code>	Se activa cuando se presiona el ratón sobre una instancia de botón.

Hereda todos los eventos de [Clase `UIObject`](#) y [Clase `UIComponent`](#).

`CheckBox.click`

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(click){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.click = function(eventObject){  
    ...  
}  
checkBoxInstance.addEventListener("click", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se hace clic con el ratón (se suelta el botón del ratón) en la casilla de verificación o cuando ésta está seleccionada y se presiona la barra espaciadora.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente `CheckBox`. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado a la casilla de verificación `myCheckBox`, envía “_level0.myCheckBox” al panel Salida:

```
on(click){
    trace(this);
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*checkBoxInstance*) distribuye un evento (en este caso `click`) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (*eventObject*) al método del objeto detector. El objeto `Event` tiene un conjunto de propiedades que contiene información acerca del mismo. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `addEventListener()` (véase [UIEventDispatcher.addEventListener\(\)](#)) de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en la página 236.

Ejemplo

Este ejemplo, escrito en un fotograma de la línea de tiempo, envía un mensaje al panel Salida cuando se hace clic en un botón denominado `checkBoxInstance`. La primera línea de código crea un objeto detector denominado `form`. La segunda línea define una función para el evento `click` en el objeto detector. Dentro de la función hay una acción `trace` que utiliza el objeto `Event` pasado automáticamente a la función (en este ejemplo `eventObj`) para generar un mensaje. La propiedad `target` de un objeto `Event` es el componente que ha generado el evento (en este ejemplo `checkBoxInstance`). A la propiedad `CheckBox.selected` se accede desde la propiedad `target` del objeto `Event`. La última línea llama al método `addEventListener()` desde `checkBoxInstance` y pasa como parámetros el evento `click` y el objeto detector `form`, como en el ejemplo siguiente:

```
form = new Object();
form.click = function(eventObj){
    trace("La propiedad seleccionada ha cambiado a " +
        eventObj.target.selected);
}
checkBoxInstance.addEventListener("click", form);
```

El código siguiente también envía un mensaje al panel Salida cuando se hace clic en `checkBoxInstance`. El controlador `on()` debe asociarse directamente a `checkBoxInstance`, como en el ejemplo siguiente:

```
on(click){
    trace("clic en componente de casilla de verificación");
}
```

Véase también

[UIEventDispatcher.addEventListener\(\)](#)

CheckBox.label

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

checkBoxInstance.label

Descripción

Propiedad; indica la etiqueta de texto de la casilla de verificación. De forma predeterminada, la etiqueta aparece a la derecha de la casilla de verificación. Cuando se define, esta propiedad sustituye al parámetro label especificado en el panel de parámetros del clip.

Ejemplo

El código siguiente define el texto que aparece junto al componente CheckBox y envía el valor al panel Salida:

```
checkBox.label = "Eliminar de la lista";  
trace(checkBox.label)
```

Véase también

[CheckBox.labelPlacement](#)

CheckBox.labelPlacement

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

checkBoxInstance.labelPlacement

Descripción

Propiedad; cadena que indica la posición de la etiqueta con respecto a la casilla de verificación. A continuación se indican los cuatro valores posibles (las líneas discontinuas representan el área de delimitación del componente; son invisibles en el documento):

- "right" La casilla de verificación se fija en la esquina superior izquierda del área de delimitación. La etiqueta aparece a la derecha de la casilla. Éste es el valor predeterminado.



- "left" La casilla de verificación se fija en la esquina superior derecha del área de delimitación. La etiqueta aparece a la izquierda de la casilla.



- "bottom" La etiqueta se coloca debajo de la casilla de verificación. El conjunto de casilla y etiqueta se centra horizontal y verticalmente.



- "top" La etiqueta se coloca encima de la casilla de verificación. El conjunto de casilla y etiqueta se centra horizontal y verticalmente.



El área de delimitación del componente se puede cambiar durante la edición con el comando Transformar o en tiempo de ejecución con la propiedad `UIObject.setSize()`. Para más información, consulte [“Personalización del componente CheckBox” en la página 61](#).

Ejemplo

En el ejemplo siguiente se define la posición de la etiqueta a la izquierda de la casilla de verificación:

```
checkBox_mc.labelPlacement = "left";
```

Véase también

[CheckBox.label](#)

CheckBox.selected

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
checkBoxInstance.selected
```

Descripción

Propiedad; valor booleano que selecciona (`true`) o anula la selección (`false`) de la casilla de verificación.

Ejemplo

En el ejemplo siguiente se selecciona la instancia `checkbox1`:

```
checkbox1.selected = true;
```

Componente ComboBox

Un cuadro combinado puede ser estático o editable. Un cuadro combinado estático permite al usuario hacer una sola selección en una lista desplegable. Un cuadro combinado editable permite al usuario introducir texto directamente en un campo de texto de la parte superior de la lista y seleccionar una opción en una lista desplegable. Si la lista desplegable llega al final del documento, se abre hacia arriba en lugar de hacia abajo. El cuadro combinado se compone de tres subcomponentes: un componente Button, un componente TextInput y un componente List.

Cuando se hace la selección en la lista, la etiqueta de la selección se copia en el campo de texto de la parte superior del cuadro combinado. No importa si la selección se realiza con el ratón o con el teclado.

El componente ComboBox se selecciona al hacer clic en el cuadro de texto o en el botón. Cuando un componente ComboBox está seleccionado y es editable, todas las pulsaciones del teclado se dirigen al cuadro de texto y se gestionan según las reglas del componente TextInput (véase [“Componente TextInput” en la página 216](#)), con excepción de las teclas siguientes:

Tecla	Descripción
Control + Flecha abajo	Abre la lista desplegable y la selecciona.
Mayúsculas + Tabulador	Desplaza la selección al objeto anterior.
Tabulador	Desplaza la selección al objeto siguiente.

Cuando un componente ComboBox está seleccionado y es estático, las pulsaciones alfanuméricas desplazan la selección hacia arriba y hacia abajo en la lista desplegable al siguiente elemento cuya inicial coincida con el carácter introducido. También puede utilizar las siguientes teclas para controlar un cuadro combinado estático:

Tecla	Descripción
Control + Flecha abajo	Abre la lista desplegable y la selecciona.
Control + Flecha arriba	Cierra la lista desplegable si está abierta.
Flecha abajo	La selección desciende un elemento.
Fin	La selección se desplaza al final de la lista.
Escape	Cierra la lista desplegable y vuelve a seleccionar el cuadro combinado.
Intro	Cierra la lista desplegable y vuelve a seleccionar el cuadro combinado.
Inicio	La selección se desplaza al principio de la lista.
Av Pág	La selección avanza una página.
Re Pág	La selección retrocede una página.
Mayúsculas + Tabulador	Desplaza la selección al objeto anterior.
Tabulador	Desplaza la selección al objeto siguiente.

Cuando la lista desplegable de un cuadro combinado está seleccionada, las pulsaciones alfanuméricas desplazan la selección hacia arriba y hacia abajo en la lista al siguiente elemento cuya inicial coincida con el carácter introducido. También puede utilizar las siguientes teclas para controlar una lista desplegable:

Tecla	Descripción
Control + Flecha arriba	Si la lista desplegable está abierta, se vuelve a seleccionar el cuadro de texto y se cierra la lista desplegable.
Flecha abajo	La selección desciende un elemento.
Fin	El punto de inserción se desplaza al final del cuadro de texto.
Intro	Si la lista desplegable está abierta, se vuelve a seleccionar el cuadro de texto y se cierra la lista desplegable.
Escape	Si la lista desplegable está abierta, se vuelve a seleccionar el cuadro de texto y se cierra la lista desplegable.
Inicio	El punto de inserción se desplaza al principio del cuadro de texto.
Av Pág	La selección avanza una página.
Re Pág	La selección retrocede una página.
Tabulador	Desplaza la selección al objeto siguiente.
Mayús + Fin	Selecciona el texto desde el punto de inserción a la posición de fin.
Mayús + Inicio	Selecciona el texto desde el punto de inserción a la posición de inicio.
Mayús + Tabulador	Desplaza la selección al objeto anterior.
Flecha arriba	La selección asciende un elemento.

Nota: para las teclas Av Pág y Re Pág, una página está formada por los elementos que caben en pantalla menos uno. Por ejemplo, para avanzar por una lista desplegable que presenta las líneas de diez en diez, la pantalla muestra los elementos 0-9, 9-18, 18-27 y así sucesivamente, solapando un elemento por página.

Para más información sobre el control de la selección, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 24](#) o [“Clase FocusManager” en la página 101](#).

La vista previa dinámica de cada instancia del componente ComboBox del escenario refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o el panel Inspector de componentes. Sin embargo, la lista desplegable no se abre en la vista previa dinámica, y el primer elemento aparece como elemento seleccionado.

Cuando se añade el componente ComboBox a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de la pantalla puedan acceder a él. En primer lugar, debe añadir la línea de código siguiente para activar la accesibilidad:

```
mx.accessibility.ComboBoxAccImpl.enableAccessibility();
```

La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias. Para más información, consulte [“Creación de contenido accesible” en el apartado Utilización de Flash de la Ayuda](#). Es posible que tenga que actualizar el sistema de Ayuda para consultar esta información.

Utilización del componente ComboBox

Utilice el componente ComboBox en formularios o aplicaciones en los que haya que elegir una sola opción en una lista. Por ejemplo, puede emplear una lista desplegable de provincias en un formulario de dirección de cliente. En contextos más complejos, utilice cuadros combinados editables. Por ejemplo, en una aplicación de cálculo de itinerarios, podría emplear un cuadro combinado editable para que el usuario introduzca las direcciones de origen y de destino. La lista desplegable contendría las direcciones introducidas con anterioridad.

Parámetros de ComboBox

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente ComboBox en el inspector de propiedades o el panel Inspector de componentes:

editable determina si el componente ComboBox es editable (true) o sólo seleccionable (false). El valor predeterminado es false.

labels rellena el componente ComboBox con una matriz de valores de texto.

data asocia un valor de datos a cada elemento del componente ComboBox. El parámetro data es una matriz.

rowCount establece el número de elementos que se pueden mostrar de una vez sin utilizar la barra de desplazamiento. El valor predeterminado es 5.

Puede escribir código ActionScript para definir opciones adicionales para instancias de ComboBox con los métodos, propiedades y eventos de la clase ComboBox. Para más información, consulte [Clase ComboBox](#).

Creación de aplicaciones con el componente ComboBox

En el siguiente procedimiento se explica cómo añadir un componente ComboBox a una aplicación durante la edición. En este ejemplo, el cuadro combinado presenta una lista desplegable de ciudades para elegir.

Para crear una aplicación con el componente ComboBox, siga este procedimiento:

- 1 Arrastre un componente ComboBox desde el panel Componentes al escenario.
- 2 Seleccione la herramienta Transformación y cambie el tamaño del componente en el escenario. Sólo es posible cambiar el tamaño de un cuadro combinado en el escenario durante la edición. Normalmente sólo hace falta modificar la anchura del cuadro para adaptarla al contenido.
- 3 Seleccione el cuadro combinado y, en el inspector de propiedades, introduzca el nombre de instancia **comboBox**.
- 4 En el panel Inspector de componentes o el inspector de propiedades, siga este procedimiento:
 - Introduzca Minneapolis, Portland y Keene en el parámetro label. Haga doble clic en el campo del parámetro label para abrir el cuadro de diálogo Valores. A continuación, haga clic en el signo más para añadir elementos.
 - Introduzca MN.swf, OR.swf y NH.swf en el parámetro data.
Son archivos SWF imaginarios que, por ejemplo, podrían cargarse cuando el usuario seleccionara una ciudad en el cuadro combinado.
- 5 Seleccione el fotograma 1 de la línea de tiempo, abra el panel Acciones e introduzca el código siguiente:

```

form = new Object();
form.change = function (evt){
    trace(evt.target.selectedItem.label);
}
comboBox.addEventListener("change", form);

```

La última línea de código añade un controlador de eventos `change` a la instancia `comboBox`. Para más información, consulte [ComboBox.change](#).

Personalización del componente ComboBox

El componente `ComboBox` puede transformarse horizontal y verticalmente durante la edición. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar.

Si el texto es demasiado largo para encajar en el cuadro combinado, se trunca. Para encajar el texto de la etiqueta, deberá cambiar el tamaño del cuadro combinado durante la edición.

En los cuadros combinados editables, el área activa es sólo el botón, no el cuadro de texto. En los cuadros combinados estáticos, el área activa está formada por el botón y el cuadro de texto.

Utilización de estilos con el componente ComboBox

Es posible definir propiedades de estilo para cambiar la apariencia de un componente `ComboBox`. Si el nombre de una propiedad de estilo termina por “Color”, significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte “Utilización de estilos para personalizar el texto y el color de un componente” en la página 27.

Los cuadros combinados sólo tienen dos estilos exclusivos. Los estilos del botón, el cuadro de texto y la lista desplegable del cuadro combinado se asignan a través de dichos componentes, como se indica a continuación:

- El botón es una instancia de `Button` y utiliza sus estilos. (Véase “Utilización de estilos con el componente `Button`” en la página 50.)
- El texto es una instancia de `TextInput` y utiliza sus estilos. (Véase “Utilización de estilos con el componente `TextInput`” en la página 218.)
- La lista desplegable es una instancia de `List` y utiliza sus estilos. (Véase “Utilización de estilos con el componente `List`” en la página 115.)

El componente `ComboBox` utiliza los siguientes estilos de Halo:

Estilo	Descripción
<code>themeColor</code>	Fondo de un componente. Es el único estilo de color que no hereda su valor. Los valores posibles son "haloGreen", "haloBlue" y "haloOrange".
<code>color</code>	Texto de la etiqueta de un componente.
<code>disabledColor</code>	Color desactivado para el texto.
<code>fontFamily</code>	Nombre de la fuente del texto.
<code>fontSize</code>	Tamaño de la fuente en puntos.
<code>fontStyle</code>	Estilo de la fuente: puede ser "normal" o "italic".

Estilo	Descripción
fontWeight	Grosor de la fuente: puede ser “normal” o “bold”.
textDecoration	Decoración del texto: puede ser “none” o “underline”.
openDuration	Número de milisegundos para abrir la lista desplegable. El valor predeterminado es 250.
openEasing	Referencia a la función de interpolación que controla la animación de la lista desplegable. De forma predeterminada, va a la pantalla de inicio y luego a la pantalla de desconexión del sitio Web. Para obtener más funciones, descargue la lista del sitio Web de Robert Penner .

Utilización de aspectos con el componente ComboBox

El componente ComboBox utiliza símbolos del panel Biblioteca para representar estados de botón. El componente ComboBox tiene variables de aspecto para la flecha abajo. Aparte, utiliza la barra de desplazamiento y los aspectos de lista. Para asignar aspectos al componente ComboBox durante la edición, modifique símbolos del panel Biblioteca y vuelva a exportar el componente como SWC. Los aspectos del componente CheckBox se encuentran en la carpeta Flash UI Components 2/Themes/MMDefault/ComboBox Assets/states de la biblioteca del archivo HaloTheme.fla o SampleTheme.fla. Para más información, consulte [“Aplicación de aspectos a los componentes” en la página 37](#).

El componente ComboBox utiliza las siguientes propiedades de aspecto:

Propiedad	Descripción
ComboDownArrowDisabledName	Estado Desactivado de la flecha abajo. El valor predeterminado es RectBorder.
ComboDownArrowDownName	Estado Presionado de la flecha abajo. El valor predeterminado es RectBorder.
ComboDownArrowUpName	Estado Arriba de la flecha abajo. El valor predeterminado es RectBorder.
ComboDownArrowOverName	Estado Sobre de la flecha abajo. El valor predeterminado es RectBorder.

Clase ComboBox

Herencia UIObject > UIComponent > ComboBase > ComboBox

Namespace de clase de ActionScript mx.controls.ComboBox

El componente ComboBox combina tres subcomponentes independientes: Button, TextInput y List. La mayoría de las interfaces API de cada subcomponente están disponibles directamente en el componente ComboBox y se enumeran en las tablas de métodos, propiedades y eventos de la clase ComboBox.

La lista desplegable de un cuadro combinado se obtiene con un objeto Array o un objeto DataProvider. Si utiliza un objeto DataProvider, la lista cambia en tiempo de ejecución. El origen de los datos de ComboBox puede modificarse dinámicamente cambiando a otro objeto Array o DataProvider.

Los elementos de una lista de un cuadro combinado se indexan por posición a partir del número 0. Un elemento puede ser:

- Un tipo de datos primitivo.
- Un objeto que contiene una propiedad `label` y una propiedad `data`.

Nota: un objeto puede utilizar la propiedad `ComboBox.labelFunction` o `ComboBox.labelField` para determinar la propiedad `label`.

Si el elemento es un tipo de datos primitivo distinto de una cadena, se convierte en una cadena. Si el elemento es un objeto, la propiedad `label` debe ser una cadena y la propiedad `data` puede ser cualquier valor de `ActionScript`.

Los métodos del componente `ComboBox` a los que se suministran elementos tienen dos parámetros, `label` y `data`, que hacen referencia a las propiedades del mismo nombre. Los métodos que devuelven elementos devuelven objetos.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.controls.ComboBox.version);
```

Nota: el código siguiente devuelve `undefined`: `trace(myComboBoxInstance.version);`.

Resumen de métodos de la clase `ComboBox`

Propiedad	Descripción
<code>ComboBox.addItem()</code>	Añade un elemento al final de la lista.
<code>ComboBox.addItemAt()</code>	Añade un elemento al final de la lista en el índice especificado.
<code>ComboBox.close()</code>	Cierra la lista desplegable.
<code>ComboBox.getItemAt()</code>	Devuelve el elemento del índice especificado.
<code>ComboBox.open()</code>	Abre la lista desplegable.
<code>ComboBox.removeAll()</code>	Elimina todos los elementos de la lista.
<code>ComboBox.removeItemAt()</code>	Elimina un elemento de la lista en la posición especificada.
<code>ComboBox.replaceItemAt()</code>	Sustituye un elemento de la lista por otro elemento especificado.

Hereda todos los métodos de [Clase `UIObject`](#) y [Clase `UIComponent`](#).

Resumen de propiedades de la clase `ComboBox`

Propiedad	Descripción
<code>ComboBox.dataProvider</code>	Modelo de datos de los elementos de la lista.
<code>ComboBox.dropdown</code>	Devuelve una referencia al componente <code>List</code> contenido en el cuadro combinado.
<code>ComboBox.dropdownWidth</code>	Anchura de la lista desplegable, expresada en píxeles.
<code>ComboBox.editable</code>	Indica si el cuadro combinado es editable o no.

Propiedad	Descripción
<code>ComboBox.labelField</code>	Indica qué campo de datos ha de utilizarse como etiqueta para la lista desplegable.
<code>ComboBox.labelFunction</code>	Especifica una función para calcular el campo de etiqueta de la lista desplegable.
<code>ComboBox.length</code>	Sólo lectura. Longitud de la lista desplegable.
<code>ComboBox.rowCount</code>	Número máximo de elementos de la lista que se muestran de una vez.
<code>ComboBox.selectedIndex</code>	Índice del elemento seleccionado en la lista desplegable.
<code>ComboBox.selectedItem</code>	Valor del elemento seleccionado en la lista desplegable.
<code>ComboBox.text</code>	Cadena de texto del cuadro de texto.
<code>ComboBox.textField</code>	Referencia al componente TextInput del cuadro combinado.
<code>ComboBox.value</code>	Valor del cuadro de texto (editable) o lista desplegable (estático).

Hereda todas las propiedades de [Clase UIObject](#) y [Clase UIComponent](#).

Resumen de eventos de la clase ComboBox

Evento	Descripción
<code>ComboBox.change</code>	Se difunde cuando el valor del cuadro combinado cambia como resultado de la interacción del usuario.
<code>ComboBox.close</code>	Se difunde cuando la lista desplegable comienza a cerrarse.
<code>ComboBox.enter</code>	Se difunde cuando se presiona la tecla Intro.
<code>ComboBox.itemRollOut</code>	Se difunde cuando el puntero se desplaza fuera de un elemento de la lista desplegable.
<code>ComboBox.itemRollOver</code>	Se difunde cuando el ratón se desplaza sobre un elemento de la lista desplegable.
<code>ComboBox.open</code>	Se difunde cuando la lista desplegable comienza a abrirse.
<code>ComboBox.scroll</code>	Se difunde cuando se recorre la lista desplegable.

Hereda todos los eventos de [Clase UIObject](#) y [Clase UIComponent](#).

ComboBox.addItem()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
comboBoxInstance.addItem(label[, data])
```

Sintaxis 2:

```
comboBoxInstance.addItem({label:label[, data:data]})
```

Sintaxis 3:

```
comboBoxInstance.addItem(obj);
```

Parámetros

label Cadena que indica la etiqueta del elemento nuevo.

data Datos del elemento; pueden ser cualquier tipo de datos. Este parámetro es opcional.

obj Objeto con una propiedad *label* y una propiedad *data* opcional.

Valor devuelto

Índice en el que se ha añadido el elemento.

Descripción

Método; añade un elemento nuevo al final de la lista.

Ejemplo

El código siguiente añade un elemento a la instancia *myComboBox*:

```
myComboBox.addItem("esto es un elemento");
```

ComboBox.addItemAt()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
comboBoxInstance.addItemAt(index, label[, data])
```

Parámetros

index Número 0 o superior que indica la posición en la que insertar el elemento (índice del elemento nuevo).

label Cadena que indica la etiqueta del elemento nuevo.

data Datos del elemento; pueden ser cualquier tipo de datos. Este parámetro es opcional.

Valor devuelto

Índice en el que se ha añadido el elemento.

Descripción

Método; añade un elemento nuevo al final de la lista en el índice especificado con el parámetro *index*. Los índices superiores a `ComboBox.length` se omiten.

Ejemplo

El código siguiente inserta un elemento en el índice 3, que es la cuarta posición de la lista del cuadro combinado (la primera posición es 0):

```
myBox.addItemAt(3, "éste es el cuarto elemento");
```

ComboBox.change

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(change){  
    // aquí código propio  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.change = function(eventObject){  
    // aquí código propio  
}  
comboBoxInstance.addEventListener("change", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando el valor del cuadro combinado cambia como resultado de la interacción del usuario.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente `ComboBox`. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado a la instancia `myBox` del componente `ComboBox`, envía “_level0.myBox” al panel Salida:

```
on(change){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*comboBoxInstance*) distribuye un evento (en este caso *change*) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto Event (*eventObject*) al método del objeto detector. Cada objeto Event tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `addEventListener()` (véase `UIEventDispatcher.addEventListener()`) de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en la página 236.

Ejemplo

En el ejemplo siguiente se envía al panel Salida el nombre de instancia del componente que ha generado el evento *change*:

```
form.change = function(eventObj){
    trace("Valor cambiado a " + eventObj.target.value);
}
myCombo.addEventListener("change", form);
```

Véase también

`UIEventDispatcher.addEventListener()`

ComboBox.close()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
myComboBox.close()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; cierra la lista desplegable.

Ejemplo

En el ejemplo siguiente se cierra la lista desplegable del cuadro combinado `myBox`:

```
myBox.close();
```

Véase también

[ComboBox.open\(\)](#)

ComboBox.close

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(close){  
    // aquí código propio  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.close = function(eventObject){  
    // aquí código propio  
}  
comboBoxInstance.addEventListener("close", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando la lista del cuadro combinado comienza a plegarse.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente `ComboBox`. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado a la instancia `myBox` del componente `ComboBox`, envía “_level0.myBox” al panel Salida:

```
on(close){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*comboBoxInstance*) distribuye un evento (en este caso `close`) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (*eventObject*) al método del objeto detector. Cada objeto `Event` tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 236](#).

Ejemplo

En el ejemplo siguiente se envía un mensaje al panel Salida cuando la lista desplegable comienza a cerrarse:

```
form.close = function(){
    trace("El cuadro combinado se ha cerrado");
}
myCombo.addEventListener("close", form);
```

Véase también

[UIEventDispatcher.addEventListener\(\)](#)

ComboBox.dataProvider

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

comboBoxInstance.dataProvider

Descripción

Propiedad; modelo de datos de los elementos que se ven en una lista. El valor de esta propiedad puede ser una matriz o cualquier objeto que implemente la interfaz `DataProvider`. El valor predeterminado es `[]`. Ésta es una propiedad del componente `List`, pero se puede acceder a ella directamente desde una instancia del componente `ComboBox`.

El componente `List` y otros componentes para datos añaden métodos al prototipo del objeto `Array` para ceñirse a la interfaz `DataProvider` (para más detalles, consulte `DataProvider.as`). Por lo tanto, cualquier matriz que exista a la vez que la lista tiene automáticamente todos los métodos necesarios (`addItem()`, `getItemAt()`, etc.) para ser el modelo de la lista y puede utilizarse para difundir cambios de modelo a varios componentes.

Si la matriz contiene objetos, accede a las propiedades `labelField` o `labelFunction` para determinar qué partes del elemento deben mostrarse. El valor predeterminado es `"label"`, por lo que, si existe este campo, es el campo que se muestra; en caso contrario, se muestra una lista de todos los campos separados por comas.

Nota: si la matriz contiene cadenas y no objetos en todos los índices, la lista no puede ordenar los elementos y mantener el estado de la selección. Al ordenar se pierde la selección.

Como proveedor de datos para `List` pueden elegirse todas las instancias que implementen la interfaz `DataProvider`. Entre ellas se encuentran los juegos de registros de `Flash Remoting`, los conjuntos de datos de `Firefly`, etc.

Ejemplo

Este ejemplo utiliza una matriz de cadenas para rellenar la lista desplegable:

```
comboBox.dataProvider = ["Envío estándar", "Aéreo dos días", "Aéreo exprés"];
```

En este ejemplo se crea una matriz proveedora de datos y se asigna a la propiedad `dataProvider`:

```
myDP = new Array();
list.dataProvider = myDP;

for (var i=0; i<accounts.length; i++) {
    // estos cambios en DataProvider se difundirán a la lista
    myDP.addItem({ label: accounts[i].name,
                    data: accounts[i].accountID });
}
```

ComboBox.dropdown

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

myComboBox.dropdown

Descripción

Propiedad (sólo lectura); devuelve una referencia al componente List contenido en el cuadro combinado. No se crea una instancia en el cuadro combinado para el subcomponente List hasta que se tenga que visualizar. No obstante, cuando se accede a la propiedad `dropdown`, se crea la lista.

Véase también

[ComboBox.dropdownWidth](#)

ComboBox.dropdownWidth

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

myComboBox.change

Descripción

Propiedad; límite de anchura de la lista desplegable, expresado en píxeles. El valor predeterminado es la anchura del componente ComboBox (instancia de TextInput más instancia de SimpleButton).

Ejemplo

El código siguiente define `dropdownWidth` en 150 píxeles:

```
myComboBox.dropdownWidth = 150;
```

Véase también

[ComboBox.dropdown](#)

ComboBox.editable

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

myComboBox.editable

Descripción

Propiedad; indica si el cuadro combinado es editable (`true`) o no (`false`). Un cuadro combinado editable permite introducir en el cuadro de texto valores que no se ven en la lista desplegable. Si el cuadro combinado no es editable, en el cuadro de texto sólo pueden introducirse los valores de la lista desplegable. El valor predeterminado es `false`.

Si el cuadro combinado se define en `editable`, el campo de texto del cuadro combinado se borra. Además, el índice (y el elemento) seleccionado se define en un valor no definido. Para convertir un cuadro combinado en editable conservando el elemento seleccionado, utilice el código siguiente:

```
var ix = myComboBox.selectedIndex;
myComboBox.editable = true; // borra el campo de texto.
myComboBox.selectedIndex = ix; // vuelve a copiar la etiqueta en el campo de
    texto.
```

Ejemplo

El código siguiente convierte `myComboBox` en editable:

```
myComboBox.editable = true;
```

ComboBox.enter

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(enter){
    // aquí código propio
}
```


Sintaxis 2:

```
listenerObject = new Object();
listenerObject.enter = function(eventObject){
    // aquí código propio
}
comboBoxInstance.addEventListener("enter", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se presiona la tecla Intro en el cuadro de texto. Este evento sólo se difunde desde cuadros combinados editables. Éste es un evento `TextInput` que se difunde desde un cuadro combinado. Para más información, consulte [TextInput.enter](#).

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente `ComboBox`. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado a la instancia `myBox` del componente `ComboBox`, envía “_level0.myBox” al panel Salida:

```
on(enter){
    trace(this);
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*comboBoxInstance*) distribuye un evento (en este caso `enter`) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (*eventObject*) al método del objeto detector. Cada objeto `Event` tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en [la página 236](#).

Ejemplo

En el ejemplo siguiente se envía un mensaje al panel Salida cuando la lista desplegable comienza a cerrarse:

```
form.enter = function(){
    trace("Evento enter del cuadro combinado activado");
}
myCombo.addEventListener("enter", form);
```

Véase también

[UIEventDispatcher.addEventListener\(\)](#)

ComboBox.getItemAt()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
comboBoxInstance.getItemAt(index)
```

Parámetros

index Número mayor o igual que 0 y menor que `ComboBox.length`. Índice del elemento que debe recuperarse.

Valor devuelto

Valor u objeto del elemento indexado. Si el índice está fuera de rango, el valor es undefined.

Descripción

Método; recupera el elemento del índice especificado.

Ejemplo

El código siguiente muestra el elemento de la posición 4 del índice:

```
trace(myBox.getItemAt(4).label);
```

ComboBox.itemRollOut

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(itemRollOut){  
    // aquí código propio  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.itemRollOut = function(eventObject){  
    // aquí código propio  
}  
comboBoxInstance.addEventListener("itemRollOut", listenerObject)
```

Objeto Event

Además de las propiedades estándar del objeto Event, el evento `itemRollOut` tiene una propiedad adicional: `index`. `index` es el número del elemento del que ha salido el puntero.

Descripción

Evento; se difunde a todos los detectores registrados cuando el puntero sale de los elementos de la lista desplegable. Éste es un evento List que se difunde desde un cuadro combinado. Para más información, consulte [List.itemRollOut](#).

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente `ComboBox`. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado a la instancia `myBox` del componente `ComboBox`, envía “_level0.myBox” al panel Salida:

```
on(itemRollOut){
    trace(this);
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*comboBoxInstance*) distribuye un evento (en este caso `itemRollOut`) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (*eventObject*) al método del objeto detector. Cada objeto `Event` tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Para más información acerca de los objetos de eventos, consulte “Objetos `Event`” en la página 236.

Finalmente, se llama al método `addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Ejemplo

En el ejemplo siguiente se envía un mensaje al panel Salida para indicar de qué número de índice de elemento ha salido el puntero:

```
form.itemRollOut = function (eventObj) {
    trace("Elemento n.º" + eventObj.index + " de donde ha salido el puntero.");
}
myCombo.addEventListener("itemRollOut", form);
```

Véase también

`ComboBox.itemRollOver`, `UIEventDispatcher.addEventListener()`

ComboBox.itemRollOver

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(itemRollOver){
    // aquí código propio
}
```

Sintaxis 2:

```
listenerObject = new Object();
listenerObject.itemRollOver = function(eventObject){
    // aquí código propio
}
comboBoxInstance.addEventListener("itemRollOver", listenerObject)
```

Objeto Event

Además de las propiedades estándar del objeto Event, el evento `itemRollOver` tiene una propiedad adicional: `index`. `index` es el número del elemento sobre el que ha pasado el puntero.

Descripción

Evento; se difunde a todos los detectores registrados cuando el puntero se desplaza por los elementos de la lista desplegable. Éste es un evento List que se difunde desde un cuadro combinado. Para más información, consulte [List.itemRollOver](#).

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente `ComboBox`. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado a la instancia `myBox` del componente `ComboBox`, envía “_level0.myBox” al panel Salida:

```
on(itemRollOver){
    trace(this);
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*comboBoxInstance*) distribuye un evento (en este caso `itemRollOver`) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto Event (*eventObject*) al método del objeto detector. Cada objeto Event tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Para más información acerca de los objetos de eventos, consulte “Objetos Event” en la página 236.

Finalmente, se llama al método `addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Ejemplo

En el ejemplo siguiente se envía un mensaje al panel Salida para indicar por qué número de índice de elemento ha pasado el puntero:

```
form.itemRollOver = function (eventObj) {
    trace("Elemento n.º" + eventObj.index + " por donde ha pasado el puntero.");
}
myCombo.addEventListener("itemRollOver", form);
```

Véase también

[ComboBox.itemRollOut](#), [UIEventDispatcher.addEventListener\(\)](#)

ComboBox.labelField

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

myComboBox.labelField

Descripción

Propiedad; nombre del campo de los objetos de matriz dataProvider que se utilizará como campo de etiqueta. Ésta es una propiedad del componente List que está disponible desde una instancia del componente ComboBox. Para más información, consulte [List.labelField](#).

El valor predeterminado es undefined.

Ejemplo

En el ejemplo siguiente se define la propiedad dataProvider en una matriz de cadenas y la propiedad labelField para que indique que el campo name debe utilizarse como etiqueta de la lista desplegable:

```
myComboBox.dataProvider = [
    {name:"Gary", gender:"male"},
    {name:"Susan", gender:"female"} ];
```

```
myComboBox.labelField = "name";
```

Véase también

[List.labelFunction](#)

ComboBox.labelFunction

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

myComboBox.labelFunction

Descripción

Propiedad; función que calcula la etiqueta de un elemento dataProvider. Es preciso definir la función. El valor predeterminado es undefined.

Ejemplo

En el ejemplo siguiente se crea un proveedor de datos y después define una función que especifica qué debe utilizarse como etiqueta en la lista desplegable:

```
myComboBox.dataProvider = [
    {firstName:"Nigel", lastName:"Pegg", age:"really young"},
    {firstName:"Gary", lastName:"Grossman", age:"young"},
    {firstName:"Chris", lastName:"Walcott", age:"old"},
    {firstName:"Greg", lastName:"Yachuk", age:"really old" }];

myComboBox.labelFunction = function(itemObj){
    return (itemObj.lastName + ", " + itemObj.firstName);
}
```

Véase también

[List.labelField](#)

ComboBox.length

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

myComboBox.length

Descripción

Propiedad (sólo lectura); longitud de la lista desplegable. Ésta es una propiedad del componente List que está disponible desde una instancia de ComboBox. Para más información, consulte [List.length](#). El valor predeterminado es 0.

Ejemplo

En el ejemplo siguiente se almacena el valor `length` en una variable:

```
dropdownItemCount = myBox.length;
```

ComboBox.open()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

myComboBox.open()

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Propiedad; abre la lista desplegable.

Ejemplo

El código siguiente abre la lista desplegable de la instancia `combo1`:

```
combo1.open();
```

Véase también

[ComboBox.close\(\)](#)

ComboBox.open

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(open){  
    // aquí código propio  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.open = function(eventObject){  
    // aquí código propio  
}  
comboBoxInstance.addEventListener("open", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando la lista desplegable comienza a aparecer.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente `ComboBox`. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado a la instancia `myBox` del componente `ComboBox`, envía “_level0.myBox” al panel Salida:

```
on(open){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*comboBoxInstance*) distribuye un evento (en este caso *open*) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto Event (*eventObject*) al método del objeto detector. Cada objeto Event tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Para más información acerca de los objetos de eventos, consulte “Objetos Event” en la página 236.

Finalmente, se llama al método `addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Ejemplo

En el ejemplo siguiente se envía un mensaje al panel Salida para indicar de qué número de índice de elemento ha salido el puntero:

```
form.open = function () {  
    trace("El cuadro combinado se ha abierto con texto " + myBox.text);  
}  
myBox.addEventListener("open", form);
```

Véase también

`ComboBox.close`, `UIEventDispatcher.addEventListener()`

ComboBox.removeAll()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
comboBoxInstance.removeAll()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; elimina todos los elementos de la lista. Éste es un método del componente List que está disponible desde una instancia del componente ComboBox.

Ejemplo

El código siguiente borra la lista:

```
myCombo.removeAll();
```


Véase también

[ComboBox.removeItemAt\(\)](#), [ComboBox.replaceItemAt\(\)](#)

ComboBox.removeItemAt()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
listInstance.removeItemAt(index)
```

Parámetros

index Número que indica la posición del elemento que va a eliminarse. El valor comienza a partir de cero.

Valor devuelto

Objeto; elemento eliminado (undefined si no hay elementos).

Descripción

Método; elimina el elemento de la posición especificada en *index*. Los índices de la lista a partir del índice que indica el parámetro *index* se contraen una posición. Éste es un método del componente List que está disponible desde una instancia del componente ComboBox.

Ejemplo

El código siguiente elimina el elemento de la posición 3 del índice:

```
myCombo.removeItemAt(3);
```

Véase también

[ComboBox.removeAll\(\)](#), [ComboBox.replaceItemAt\(\)](#)

ComboBox.replacelItemAt()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
comboBoxInstance.replaceItemAt(index, label[, data])
```

Parámetros

index Número 0 o superior que indica la posición en la que insertar el elemento (índice del elemento nuevo).

label Cadena que indica la etiqueta del elemento nuevo.

data Datos del elemento. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; sustituye el contenido del elemento del índice que especifica el parámetro *index*. Éste es un método del componente List que está disponible desde el componente ComboBox.

Ejemplo

En el ejemplo siguiente se cambia la posición del tercer índice:

```
myCombo.replaceItemAt(3, "new label");
```

Véase también

`ComboBox.removeAll()`, `ComboBox.removeItemAt()`

ComboBox.rowCount

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
myComboBox.rowCount
```

Descripción

Propiedad; número máximo de filas visibles en la lista desplegable. El valor predeterminado es 5.

Si la lista desplegable contiene un número de elementos mayor o igual que la propiedad `rowCount`, cambia de tamaño y, si es necesario, muestra una barra de desplazamiento. Si la lista desplegable contiene menos elementos que la propiedad `rowCount`, adapta su tamaño al número de elementos de la lista.

Este comportamiento difiere del componente List, que siempre muestra el número de filas que especifique la propiedad `rowCount` aunque quede espacio vacío.

Si el valor es negativo o fraccionario, el comportamiento es undefined.

Ejemplo

En el ejemplo siguiente se especifica que el cuadro combinado tenga 20 filas visibles o menos:

```
myComboBox.rowCount = 20;
```

ComboBox.scroll

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(scroll){  
    // aquí código propio  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.scroll = function(eventObject){  
    // aquí código propio  
}  
comboBoxInstance.addEventListener("scroll", listenerObject)
```

Objeto Event

Además de las propiedades estándar del objeto Event, el evento scroll tiene una propiedad adicional, *direction*. Es una cadena con dos valores posibles: "horizontal" o "vertical". En los eventos scroll de ComboBox, el valor siempre es "vertical".

Descripción

Evento; se difunde a todos los detectores registrados cuando se recorre la lista desplegable. Éste es un evento del componente List disponible para ComboBox.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente ComboBox. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado a la instancia `myBox` del componente ComboBox, envía “_level0.myBox” al panel Salida:

```
on(scroll){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*comboBoxInstance*) distribuye un evento (en este caso `scroll`) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto Event (*eventObject*) al método del objeto detector. Cada objeto Event tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 236](#).

Finalmente, se llama al método `addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Ejemplo

En el ejemplo siguiente se envía un mensaje al panel Salida para indicar hasta qué número de índice de elemento se ha desplazado el puntero:

```
form.scroll = function (eventObj) {  
    trace("La lista se ha desplazado al elemento n.º " +  
        eventObj.target.vPosition);  
}  
myCombo.addEventListener("scroll", form);
```

Véase también

[UIEventDispatcher.addEventListener\(\)](#)

ComboBox.selectedIndex

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

myComboBox.selectedIndex

Descripción

Propiedad; índice (número) del elemento seleccionado en la lista desplegable. El valor predeterminado es 0. Cuando se asigna esta propiedad, se borra la selección actual, se selecciona el elemento indicado y el cuadro de texto del cuadro combinado muestra la etiqueta del elemento indicado.

Si se asigna un valor `selectedIndex` fuera de rango, el valor se omite. Cuando se introduce texto en el campo de texto de un cuadro combinado editable, `selectedIndex` se establece en `undefined`.

Ejemplo

El siguiente código selecciona el último elemento de la lista:

```
myComboBox.selectedIndex = myComboBox.length-1;
```

Véase también

[ComboBox.selectedItem](#)

ComboBox.selectedItem

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

myComboBox.selectedItem

Descripción

Propiedad; valor del elemento seleccionado en la lista desplegable.

Si el cuadro combinado es editable y se introduce texto en el cuadro de texto, `selectedItem` devuelve `undefined`. Sólo contiene un valor cuando se selecciona un elemento en la lista desplegable o si el valor se define mediante `ActionScript`. Si el cuadro combinado es estático, el valor de `selectedItem` siempre es válido.

Ejemplo

En el ejemplo siguiente se muestra el valor de `selectedItem` si el proveedor de datos contiene tipos primitivos:

```
var item = myComboBox.selectedItem;
trace("Ha seleccionado el elemento " + item);
```

En el ejemplo siguiente se muestra el valor de `selectedItem` si el proveedor de datos contiene objetos con las propiedades `label` y `data`:

```
var obj = myComboBox.selectedItem;
trace("Ha seleccionado el color denominado: " + obj.label);
trace("El valor hexadecimal de este color es: " + obj.data);
```

Véase también

[ComboBox.dataProvider](#), [ComboBox.selectedIndex](#)

ComboBox.text

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

myComboBox.text

Descripción

Propiedad; texto del cuadro de texto. Este valor se puede obtener y definir con cuadros combinados editables. Con cuadros combinados estáticos, el valor es de sólo lectura.

Ejemplo

En el ejemplo siguiente se define el valor actual de `text` de un cuadro combinado editable:

```
myComboBox.text = "California";
```

ComboBox.textField

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

myComboBox.textField

Descripción

Propiedad (sólo lectura); referencia al componente TextInput contenido en el componente ComboBox.

Esta propiedad permite acceder al componente TextInput subyacente para manipularlo. Por ejemplo, quizá desee cambiar la selección del cuadro de texto o limitar los caracteres que se pueden introducir en él.

Ejemplo

El código siguiente restringe el cuadro de texto de *myComboBox* a los números aceptados:

```
myComboBox.textField.restrict = "0-9";
```

ComboBox.value

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

myComboBox.value

Descripción

Propiedad (sólo lectura); si el cuadro combinado es editable, *value* devuelve el valor del cuadro de texto. Si el cuadro de texto es estático, *value* devuelve el valor de la lista desplegable. El valor de la lista desplegable es el campo *data* o, si el campo *data* no existe, el campo *label*.

Ejemplo

En el ejemplo siguiente se introducen los datos en el cuadro combinado al definir la propiedad *dataProvider*. Después muestra *value* en el panel Salida. Finalmente, selecciona "California" y lo muestra en el cuadro de texto:

```
cb.dataProvider = [
    {label:"Alaska", data:"AZ"},
    {label:"California", data:"CA"},
    {label:"Washington", data:"WA"}];

cb.editable = true;
cb.selectedIndex = 1;
trace('El valor editable es "California": ' + cb.value);

cb.editable = false;
cb.selectedIndex = 1;
trace('El valor no editable es "CA": ' + cb.value);
```

Paquete DataBinding

Para obtener información reciente sobre esta función, haga clic en el botón Actualizar en la parte superior de la ficha Ayuda.

Componente DataGrid

Para obtener información reciente sobre esta función, haga clic en el botón Actualizar en la parte superior de la ficha Ayuda.

Componente DataHolder

Para obtener información reciente sobre esta función, haga clic en el botón Actualizar en la parte superior de la ficha Ayuda.

Componente DataProvider

Para obtener información reciente sobre esta función, haga clic en el botón Actualizar en la parte superior de la ficha Ayuda.

Componente DataSet

Para obtener información reciente sobre esta función, haga clic en el botón Actualizar en la parte superior de la ficha Ayuda.

Componente DateChooser

Para obtener información reciente sobre esta función, haga clic en el botón Actualizar en la parte superior de la ficha Ayuda.

Componente DateField

Para obtener información reciente sobre esta función, haga clic en el botón Actualizar en la parte superior de la ficha Ayuda.

Clase DepthManager

Namespace de clase de ActionScript mx.managers.DepthManager

La clase DepthManager añade funcionalidad a la clase MovieClip de ActionScript, la cual permite gestionar las asignaciones de profundidad de cualquier componente o clip de película, incluido `_root`. También permite gestionar profundidades reservadas en un clip especial de profundidad máxima situado en `_root` de servicios del sistema, como el cursor o las sugerencias.

Los métodos siguientes componen la interfaz API de clasificación de profundidad relativa:

- `DepthManager.createChildAtDepth()`
- `DepthManager.createClassChildAtDepth()`
- `DepthManager.setDepthAbove()`
- `DepthManager.setDepthBelow()`
- `DepthManager.setDepthTo()`

Los métodos siguientes componen la interfaz API de espacio de profundidad reservada:

- `DepthManager.createClassObjectAtDepth()`
- `DepthManager.createObjectAtDepth()`

Resumen de métodos de la clase DepthManager

Método	Descripción
<code>DepthManager.createChildAtDepth()</code>	Crea un elemento secundario del símbolo especificado en la profundidad especificada.
<code>DepthManager.createClassChildAtDepth()</code>	Crea un objeto de la clase especificada en la profundidad especificada.
<code>DepthManager.createClassObjectAtDepth()</code>	Crea una instancia de la clase especificada en la profundidad especificada en el clip especial de profundidad máxima.
<code>DepthManager.createObjectAtDepth()</code>	Crea un objeto en la profundidad especificada en el clip de profundidad máxima.
<code>DepthManager.setDepthAbove()</code>	Establece la profundidad por encima de la instancia especificada.
<code>DepthManager.setDepthBelow()</code>	Establece la profundidad por debajo de la instancia especificada.
<code>DepthManager.setDepthTo()</code>	Establece la profundidad en la instancia especificada en el clip de profundidad máxima.

DepthManager.createChildAtDepth()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
movieClipInstance.createChildAtDepth(linkageName, depthFlag[, initObj])
```

Parámetros

linkageName Identificador de vinculación. Este parámetro es una cadena.

depthFlag Uno de los valores siguientes: `DepthManager.kTop`, `DepthManager.kBottom`, `DepthManager.kTopmost`, `DepthManager.kNotopmost`. Todas las etiquetas de profundidad son propiedades estáticas de la clase `DepthManager`. Debe hacer referencia al paquete `DepthManager` (por ejemplo, `mx.managers.DepthManager.kTopmost`), o utilizar la sentencia `import` para importar el paquete `DepthManager`.

initObj Objeto de inicialización. Este parámetro es opcional.

Valor devuelto

Referencia al objeto creado.

Descripción

Método; crea una instancia secundaria del símbolo que especifica el parámetro *linkageName* en la profundidad que especifica el parámetro *depthFlag*.

Ejemplo

El siguiente ejemplo crea una instancia *minuteHand* del clip de película *MinuteSymbol* y la coloca encima del reloj:

```
import mx.managers.DepthManager;
minuteHand = clock.createClassChildAtDepth("MinuteSymbol", DepthManager.kTop);
```

DepthManager.createClassChildAtDepth()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004 y Flash MX Professional 2004

Sintaxis

```
movieClipInstance.createClassChildAtDepth( className, depthFlag[, initObj] )
```

Parámetros

className Nombre de clase.

depthFlag Uno de los valores siguientes: *DepthManager.kTop*, *DepthManager.kBottom*, *DepthManager.kTopmost*, *DepthManager.kNotopmost*. Todas las etiquetas de profundidad son propiedades estáticas de la clase *DepthManager*. Debe hacer referencia al paquete *DepthManager* (por ejemplo, *mx.managers.DepthManager.kTopmost*), o utilizar la sentencia *import* para importar el paquete *DepthManager*.

initObj Objeto de inicialización. Este parámetro es opcional.

Valor devuelto

Referencia al elemento secundario creado.

Descripción

Método; crea un elemento secundario de la clase que especifica el parámetro *className* en la profundidad que especifica el parámetro *depthFlag*.

Ejemplo

El siguiente código dibuja un rectángulo de selección encima de todos los objetos *NoTopmost*:

```
import mx.managers.DepthManager
this.ring = createClassChildAtDepth(mx.skins.RectBorder, DepthManager.kTop);
```

El siguiente código crea una instancia de la clase *Button* y le atribuye un valor a su propiedad *label* como parámetro *initObj*:

```
import mx.managers.DepthManager
button1 = createClassChildAtDepth(mx.controls.Button, DepthManager.kTop,
    {label: "Top Button"});
```

DepthManager.createClassObjectAtDepth()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004 y Flash MX Professional 2004

Sintaxis

```
DepthManager.createClassObjectAtDepth(className, depthSpace[, initObj])
```

Parámetros

className Nombre de clase.

depthSpace Uno de los valores siguientes: `DepthManager.kCursor`, `DepthManager.kTooltip`. Todas las etiquetas de profundidad son propiedades estáticas de la clase `DepthManager`. Debe hacer referencia al paquete `DepthManager` (por ejemplo, `mx.managers.DepthManager.kCursor`), o utilizar la sentencia `import` para importar el paquete `DepthManager`.

initObj Objeto de inicialización. Este parámetro es opcional.

Valor devuelto

Referencia al objeto creado.

Descripción

Método; crea un objeto de la clase que especifica el parámetro *className* en la profundidad que especifica el parámetro *depthSpace*. Este método se utiliza para acceder a los espacios de profundidad reservada en el clip de profundidad máxima.

Ejemplo

En el ejemplo siguiente se crea un objeto de la clase `Button`:

```
import mx.managers.DepthManager
myCursorButton = createClassObjectAtDepth(mx.controls.Button,
    DepthManager.kCursor, {label: "Cursor"});
```

DepthManager.createObjectAtDepth()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004 y Flash MX Professional 2004

Sintaxis

```
DepthManager.createObjectAtDepth(linkageName, depthSpace[, initObj])
```

Parámetros

linkageName Identificador de vinculación.

depthSpace Uno de los valores siguientes: `DepthManager.kCursor`, `DepthManager.kTooltip`. Todas las etiquetas de profundidad son propiedades estáticas de la clase `DepthManager`. Debe hacer referencia al paquete `DepthManager` (por ejemplo, `mx.managers.DepthManager.kCursor`), o utilizar la sentencia `import` para importar el paquete `DepthManager`.

initObj Objeto de inicialización.

Valor devuelto

Referencia al objeto creado.

Descripción

Método; crea un objeto en la profundidad especificada. Este método se utiliza para acceder a los espacios de profundidad reservada en el clip de profundidad máxima.

Ejemplo

En el ejemplo siguiente se crea una instancia del símbolo `TooltipSymbol` y se coloca en la profundidad reservada para las sugerencias:

```
import mx.managers.DepthManager
myCursorTooltip = createObjectAtDepth("TooltipSymbol", DepthManager.kTooltip);
```

DepthManager.setDepthAbove()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004 y Flash MX Professional 2004

Sintaxis

```
movieClipInstance.setDepthAbove(instance)
```

Parámetros

instance Nombre de instancia.

Valor devuelto

Ninguno.

Descripción

Método; establece la profundidad de un clip de película o una instancia de componente por encima de la profundidad de la instancia que especifica el parámetro *instance*.

DepthManager.setDepthBelow()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004 y Flash MX Professional 2004

Sintaxis

```
movieClipInstance.setDepthBelow(instance)
```

Parámetros

instance Nombre de instancia.

Valor devuelto

Ninguno.

Descripción

Método; establece la profundidad de un clip de película o una instancia de componente por debajo de la profundidad de la instancia que especifica el parámetro *instance*.

Ejemplo

El código siguiente establece la profundidad de la instancia `textInput` por debajo de la profundidad de `button`:

```
textInput.setDepthBelow(button);
```

DepthManager.setDepthTo()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004 y Flash MX Professional 2004

Sintaxis

```
movieClipInstance.setDepthTo(depth)
```

Parámetros

depth Nivel de profundidad.

Valor devuelto

Ninguno.

Descripción

Método; establece la profundidad de *movieClipInstance* en el valor que especifica *depth*. Este método desplaza una instancia a otra profundidad para dar cabida a otro objeto.

Ejemplo

En el ejemplo siguiente se define la profundidad de la instancia `mc1` en 10:

```
mc1.setDepthTo(10);
```

Para más información sobre la profundidad y el orden de apilamiento, consulte “Determinación del siguiente valor superior de profundidad disponible” en el apartado Diccionario de ActionScript de la Ayuda.

Clase FocusManager

FocusManager se utiliza para especificar el orden en el que se seleccionan los componentes cuando el usuario presiona el tabulador para desplazarse por una aplicación. Es posible emplear la interfaz API de FocusManager para definir un botón en el documento que reciba la entrada del teclado cuando el usuario presione Intro (Windows) o Retorno (Macintosh). Por ejemplo, cuando un usuario rellena un formulario, debe pasar de un campo a otro con el tabulador y presionar Intro (Windows) o Retorno (Macintosh) para enviar el formulario.

Todos los componentes implementan compatibilidad FocusManager; no es necesario escribir código alguno para invocarlo. FocusManager también interactúa con System Manager, que activa y desactiva instancias de FocusManager a medida que se activan y desactivan ventanas emergentes. Cada ventana modal tiene una instancia de FocusManager, por lo que los componentes de la ventana se adscriben a un conjunto de tabuladores propio, lo que impide que el usuario pase a componentes de otras ventanas al tabular.

FocusManager reconoce los grupos de botones de opción (los que tienen definida una propiedad `RadioButton.groupName`) y selecciona la instancia del grupo que tiene una propiedad `selected` definida en `true`. Cuando se presiona el tabulador, FocusManager comprueba si el objeto siguiente tiene el mismo `groupName` que el objeto activo. Si es así, desplaza automáticamente la selección al objeto siguiente que tenga un `groupName` diferente. También pueden utilizar esta función los grupos de componentes que admitan la propiedad `groupName`.

FocusManager gestiona los cambios de selección producidos con clics del ratón. Si el usuario hace clic en un componente, el componente se selecciona.

FocusManager no selecciona automáticamente ningún componente de una aplicación. La ventana principal y cualquier ventana emergente no seleccionará ningún componente de forma predeterminada a menos que llame a `focusManager.setFocus` de un componente.

Utilización de FocusManager

Para que la selección pueda desplazarse por la aplicación, defina la propiedad `tabIndex` en los objetos que deban seleccionarse (botones incluidos). Cuando el usuario presione el tabulador, FocusManager buscará el objeto activado cuya propiedad `tabIndex` sea superior al valor actual de `tabIndex`. Cuando FocusManager alcance la propiedad `tabIndex` más alta, volverá a cero. De este modo, en el ejemplo siguiente el objeto `comment` (probablemente un componente `TextArea`) es el primero en seleccionarse y después se selecciona el objeto `okButton`:

```
comment.tabIndex = 1;  
okButton.tabIndex = 2;
```

Para crear un botón que se seleccione cuando el usuario presione la tecla Intro (Windows) o Retorno (Macintosh), defina la propiedad `FocusManager.defaultPushButton` en el nombre de instancia del botón deseado, como en el ejemplo siguiente:

```
focusManager.defaultPushButton = okButton;
```

Nota: FocusManager detecta cuándo se colocan los objetos en el escenario (el orden de profundidad de los objetos) pero no sus posiciones relativas en el mismo. Es diferente al modo en el que Flash Player gestiona la tabulación.

Parámetros de FocusManager

No hay parámetros de edición para FocusManager. Es preciso utilizar los métodos y propiedades de ActionScript de la clase FocusManager en el panel Acciones. Para más información, consulte [Clase FocusManager](#).

Creación de aplicaciones con FocusManager

El siguiente procedimiento crea un plan de selección en una aplicación de Flash.

- 1 Arrastre el componente TextInput desde el panel Componentes al escenario.
- 2 En el inspector de propiedades, asígnele el nombre de instancia **comment**.
- 3 Arrastre el componente Button desde el panel Componentes al escenario.
- 4 En el inspector de propiedades, asígnele el nombre de instancia **okButton** y defina el parámetro label en **OK**.
- 5 En el fotograma 1 del panel Acciones, introduzca lo siguiente:

```
comment.tabIndex = 1;
okButton.tabIndex = 2;
focusManager.setFocus(comment);
focusManager.defaultPushButton = okButton;
lo = new Object();
lo.click = function(){
    trace("clic en botón");
}
okButton.addEventListener("click", lo);
```

Este código establece el orden de tabulación y especifica el botón predeterminado que recibirá el evento `click` cuando el usuario presione Intro (Windows) o Retorno (Macintosh).

Personalización de FocusManager

Se puede cambiar el color del anillo de selección en el tema Halo cambiando el valor del estilo `themeColor`.

FocusManager utiliza un aspecto FocusRect para seleccionar. Este aspecto puede reemplazarse o modificarse, y es posible sustituir `UIComponent.drawFocus` con subclases para emplear indicadores de selección personalizados.

Clase FocusManager

Herencia UIObject > UIComponent > FocusManager

Namespace de clase de ActionScript mx.managers.FocusManager

Resumen de métodos de la clase FocusManager

Método	Descripción
FocusManager.getFocus()	Devuelve una referencia al objeto seleccionado.
FocusManager.sendDefaultPushButtonEvent()	Envía un evento <code>click</code> a los objetos detectores registrados en el botón de comando predeterminado.
FocusManager.setFocus()	Selecciona el objeto especificado.

Resumen de propiedades de la clase FocusManager

Método	Descripción
<code>FocusManager.defaultPushButton</code>	Objeto que recibe el evento <code>click</code> cuando el usuario presiona la tecla Intro (Windows) o Retorno (Macintosh).
<code>FocusManager.defaultPushButtonEnabled</code>	Indica si la gestión con el teclado está activada (<code>true</code>) o desactivada (<code>false</code>) para el botón de comando predeterminado. El valor predeterminado es <code>true</code> .
<code>FocusManager.enabled</code>	Indica si la gestión con el tabulador está activada (<code>true</code>) o desactivada (<code>false</code>). El valor predeterminado es <code>true</code> .
<code>FocusManager.nextTabIndex</code>	Valor siguiente de la propiedad <code>tabIndex</code> .

FocusManager.defaultPushButton

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004 y Flash MX Professional 2004.

Sintaxis

```
focusManager.defaultPushButton
```

Descripción

Propiedad; especifica el botón de comando predeterminado para una aplicación. Cuando el usuario presiona la tecla Intro (Windows) o Retorno (Macintosh), los detectores del botón de comando predeterminado reciben un evento `click`. El valor predeterminado es `undefined` y el tipo de datos de esta propiedad es objeto.

FocusManager utiliza la declaración de estilo `emphasized` de la clase `SimpleButton` para indicar visualmente el botón de comando predeterminado actual.

El valor de la propiedad `defaultPushButton` es siempre el botón seleccionado. El botón de comando predeterminado no se selecciona inicialmente por haber definido la propiedad `defaultPushButton`. Si hay varios botones en una aplicación, el botón seleccionado actualmente recibe el evento `click` cuando se presiona la tecla Intro o Retorno. Si hay algún otro componente seleccionado cuando se presiona Intro o Retorno, se restablece el valor original de la propiedad `defaultPushButton`.

Ejemplo

El código siguiente define el botón de comando predeterminado en la instancia `OKButton`:

```
FocusManager.defaultPushButton = OKButton;
```

Véase también

`FocusManager.defaultPushButtonEnabled`,
`FocusManager.sendDefaultPushButtonEvent()`

FocusManager.defaultPushButtonEnabled

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
focusManager.defaultPushButtonEnabled
```

Descripción

Propiedad; valor booleano que determina si la gestión con el teclado está activada (`true`) o no (`false`) para el botón de comando predeterminado. Si `defaultPushButtonEnabled` se define en `false`, los componentes pueden recibir la entrada de la tecla Intro (Windows) o Retorno (Macintosh) y gestionarla internamente. Deberá volver a activar la gestión con el botón de comando predeterminado observando el método `onKillFocus()` del componente (véase `MovieClip.onKillFocus` en el apartado Diccionario de ActionScript de la Ayuda) o del evento `focusOut`. El valor predeterminado es `true`.

Ejemplo

El código siguiente desactiva la gestión con el botón de comando predeterminado:

```
focusManager.defaultPushButtonEnabled = false;
```

FocusManager.enabled

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
focusManager.enabled
```

Descripción

Propiedad; Valor booleano que determina si una gestión con el tabulador está activada (`true`) o no (`false`) para un grupo determinado de objetos de selección. (Por ejemplo, otra ventana emergente podría tener su propio `FocusManager`.) Si `enabled` se define en `false`, los componentes pueden recibir la entrada del tabulador y gestionarla internamente. Deberá volver a activar la gestión de `FocusManager` observando el método `onKillFocus()` del componente (véase `MovieClip.onKillFocus` en el apartado Diccionario de ActionScript de la Ayuda) o del evento `focusOut`. El valor predeterminado es `true`.

Ejemplo

El código siguiente desactiva la tabulación:

```
focusManager.enabled = false;
```


FocusManager.setFocus()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004 y Flash MX Professional 2004

Sintaxis

```
focusManager.setFocus()
```

Parámetros

Ninguno.

Valor devuelto

Referencia al objeto seleccionado.

Descripción

Método; devuelve una referencia al objeto seleccionado actualmente.

Ejemplo

El código siguiente selecciona myOKButton si el objeto actualmente seleccionado es myInputText:

```
if (focusManager.setFocus() == myInputText)
{
    focusManager.setFocus(myOKButton);
}
```

Véase también

[FocusManager.setFocus\(\)](#)

FocusManager.nextTabIndex

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
FocusManager.nextTabIndex
```

Descripción

Propiedad; siguiente número de índice de tabulador disponible. Utilice esta propiedad para definir dinámicamente la propiedad `tabIndex` de un objeto.

Ejemplo

El código siguiente asigna a la instancia mycheckbox el siguiente valor `tabIndex` más alto:

```
mycheckbox.tabIndex = focusManager.nextTabIndex;
```

Véase también

[UIComponent.tabIndex](#)

FocusManager.sendDefaultPushButtonEvent()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004 y Flash MX Professional 2004

Sintaxis

```
focusManager.sendDefaultPushButtonEvent()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; envía un evento `click` a los objetos detectores registrados en el botón de comando predeterminado. Utilice este método para enviar un evento `click` mediante programación.

Ejemplo

El código siguiente activa el evento `click` del botón de comando predeterminado y rellena los campos de nombre de usuario y contraseña cuando un usuario selecciona la instancia `CheckBox` `chb` (la casilla de verificación debe tener la etiqueta “Conexión automática”):

```
name_txt.tabIndex = 1;
password_txt.tabIndex = 2;
chb.tabIndex = 3;
submit_ib.tabIndex = 4;

focusManager.defaultPushButton = submit_ib;

chbObj = new Object();
chbObj.click = function(o){
    if (chb.selected == true){
        name_txt.text = "Jody";
        password_txt.text = "foobar";
        focusManager.sendDefaultPushButtonEvent();
    } else {
        name_txt.text = "";
        password_txt.text = "";
    }
}
chb.addEventListener("click", chbObj);

submitObj = new Object();
submitObj.click = function(o){
    if (password_txt.text != "foobar"){
        trace("error al enviar");
    } else {
```

```
        trace("¡Bien! isendDefaultPushButtonEvent ha funcionado!");
    }
}
submit_ib.addEventListener("click", submitObj);
```

Véase también

[FocusManager.defaultPushButton](#), [FocusManager.sendDefaultPushButtonEvent\(\)](#)

FocusManager.setFocus()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004 y Flash MX Professional 2004

Sintaxis

```
focusManager.setFocus(object)
```

Parámetros

object Referencia al objeto que debe seleccionarse.

Valor devuelto

Ninguno.

Descripción

Método; selecciona el objeto especificado.

Ejemplo

El código siguiente selecciona myOKButton:

```
focusManager.setFocus(myOKButton);
```

Véase también

[FocusManager.getFocus\(\)](#)

Clase Form

Para obtener información reciente sobre esta función, haga clic en el botón Actualizar en la parte superior de la ficha Ayuda.

Componente Label

Un componente Label es una línea de texto. Es posible especificar que el componente Label se formatee con HTML. También se puede controlar la alineación y tamaño de una etiqueta. Los componentes Label no tienen bordes, no se pueden seleccionar y no difunden eventos.

La vista previa dinámica de cada instancia de Label refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o el panel Inspector de componentes. Label no tiene bordes, por lo que el único modo de previsualizarlo dinámicamente es definir su parámetro text. Si el texto es demasiado largo y prefiere definir el parámetro autoSize, recuerde que éste no es compatible con la vista previa dinámica y que el recuadro de delimitación de la etiqueta no cambia de tamaño. Para seleccionar la etiqueta en el escenario, haga clic dentro del recuadro de delimitación.

Cuando se añade el componente Label a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de pantalla puedan acceder al mismo. En primer lugar, debe añadir la línea de código siguiente para activar la accesibilidad:

```
mx.accessibility.LabelAccImpl.enableAccessibility();
```

La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias. Para más información, consulte “Creación de contenido accesible” en el apartado Utilización de Flash de la Ayuda. Es posible que tenga que actualizar el sistema de Ayuda para consultar esta información.

Utilización del componente Label

Utilice el componente Label para crear etiquetas de texto para otros componentes de un formulario, como la etiqueta “Nombre:” situada a la izquierda de un campo TextInput que acepta el nombre del usuario. Si está creando una aplicación con componentes basados en la versión 2 (v2) de la arquitectura de componentes de Macromedia, es recomendable utilizar un componente Label en lugar de un campo de texto normal, ya que así podrá utilizar estilos para lograr una apariencia uniforme.

Parámetros de Label

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente Label en el inspector de propiedades o el panel Inspector de componentes:

text indica el texto de la etiqueta; el valor predeterminado es Label.

html indica si la etiqueta va a formatearse con HTML (`true`) o no (`false`). Si el parámetro `html` se define en `true`, el componente Label no podrá formatearse con estilos. El valor predeterminado es `false`.

autoSize indica cómo se cambia de tamaño y se alinea la etiqueta para que encaje el texto. El valor predeterminado es `none`. El parámetro puede tener cualquiera de estos cuatro valores:

- `none`: la etiqueta no cambia de tamaño ni se alinea para que encaje el texto.
- `left`: los lados derecho e inferior de la etiqueta cambian de tamaño para que encaje el texto. Los lados izquierdo y superior no cambian de tamaño.
- `center`: el lado inferior de la etiqueta cambia de tamaño para que encaje el texto. El centro horizontal de la etiqueta permanece fijo en su posición central horizontal original.
- `right`: los lados izquierdo e inferior de la etiqueta cambian de tamaño para que encaje el texto. Los lados superior y derecho no cambian de tamaño.

Nota: la propiedad `autoSize` del componente Label es diferente de la propiedad `autoSize` del objeto TextField incorporado de ActionScript.

Puede escribir código ActionScript para definir opciones adicionales para instancias de Label con sus métodos, propiedades y eventos. Para más información, consulte [Clase Label](#).

Creación de aplicaciones con el componente Label

El siguiente procedimiento explica cómo añadir un componente Label a una aplicación durante la edición. En este ejemplo, la etiqueta se encuentra junto a un cuadro combinado de fechas en una aplicación de carrito de la compra.

Para crear una aplicación con el componente Label, siga este procedimiento:

- 1 Arrastre un componente Label desde el panel Componentes al escenario.
- 2 En el panel Inspector de componentes, siga este procedimiento:
 - Introduzca **Expiration Date** en el parámetro label.

Personalización del componente Label

El componente Label puede transformarse horizontal y verticalmente durante la edición o en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. También es posible definir el parámetro de edición autoSize; en tal caso, el recuadro de delimitación de la vista previa dinámica no varía, pero la etiqueta sí cambia de tamaño. Para más información, consulte [“Parámetros de Label” en la página 108](#). En tiempo de ejecución, utilice el método `setSize()` (véase `UIObject.setSize()`) o `Label.autoSize`.

Utilización de estilos con el componente Label

Es posible definir propiedades de estilo para cambiar la apariencia de una instancia de Label. Todo el texto de la instancia del componente Label debe compartir el mismo estilo. Por ejemplo, no es posible definir el estilo `color` en "blue" para una palabra de la etiqueta y en "red" para la segunda palabra de la misma etiqueta.

Si el nombre de una propiedad de estilo termina por "Color", significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son.

Para más información sobre estilos, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 27](#).

Un componente Label admite los siguientes estilos:

Estilo	Descripción
<code>color</code>	Color predeterminado del texto.
<code>embedFonts</code>	Fuentes que incorporar al documento.
<code>fontFamily</code>	Nombre de la fuente del texto.
<code>fontSize</code>	Tamaño de la fuente en puntos.
<code>fontStyle</code>	Estilo de la fuente; puede ser "normal" o "italic".
<code>fontWeight</code>	Grosor de la fuente; puede ser "normal" o "bold".
<code>textAlign</code>	Alineación del texto; puede ser "left", "right" o "center".
<code>textDecoration</code>	Decoración del texto; puede ser "none" o "underline".

Utilización de aspectos con el componente Label

El componente Label no puede utilizar aspectos.

Para más información sobre asignación de aspectos a componentes, consulte [“Aplicación de aspectos a los componentes” en la página 37](#).

Clase Label

Herencia UIObject > Label

Namespace de clase de ActionScript mx.controls.Label

Las propiedades de la clase Label permiten especificar en tiempo de ejecución el texto de la etiqueta, indicar si el texto se va a formatear con HTML e indicar si la etiqueta cambia de tamaño automáticamente para que encaje el texto.

Si una propiedad de la clase Label se define con ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o el panel Inspector de componentes.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.controls.Label.version);
```

Nota: el código siguiente devuelve undefined: `trace(myLabelInstance.version);`.

Resumen de métodos de la clase Label

Hereda todos los métodos de [Clase UIObject](#).

Resumen de propiedades de la clase Label

Propiedad	Descripción
<code>Label.autoSize</code>	Cadena que indica cómo una etiqueta cambia de tamaño y se alinea para que encaje el valor de su propiedad <code>text</code> . Hay cuatro valores posibles: "none", "left", "center" y "right". El valor predeterminado es "none".
<code>Label.html</code>	Valor booleano que indica si la etiqueta puede formatearse con HTML (<code>true</code>) o no (<code>false</code>).
<code>Label.text</code>	Texto de la etiqueta.

Hereda todas las propiedades de [Clase UIObject](#).

Resumen de eventos de la clase Label

Hereda todos los eventos de [Clase UIObject](#).

Label.autoSize

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

labelInstance.autoSize

Descripción

Propiedad; cadena que indica cómo una etiqueta cambia de tamaño y se alinea para que encaje el valor de su propiedad `text`. Hay cuatro valores posibles: "none", "left", "center" y "right". El valor predeterminado es "none".

- `none`: la etiqueta no cambia de tamaño ni se alinea para que encaje el texto.
- `left`: los lados derecho e inferior de la etiqueta cambian de tamaño para que encaje el texto. Los lados izquierdo y superior no cambian de tamaño.
- `center`: el lado inferior de la etiqueta cambia de tamaño para que encaje el texto. El centro horizontal de la etiqueta permanece fijo en su posición central horizontal original.
- `right`: los lados izquierdo e inferior de la etiqueta cambian de tamaño para que encaje el texto. Los lados superior y derecho no cambian de tamaño.

Nota: la propiedad `autoSize` del componente `Label` es diferente de la propiedad `autoSize` del objeto `TextField` incorporado de `ActionScript`.

Label.html

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

labelInstance.html

Descripción

Propiedad; valor booleano que indica si la etiqueta puede formatearse con HTML (`true`) o no (`false`). El valor predeterminado es `false`. Los componentes `Label` con la propiedad `html` definida en `true` no pueden formatearse con estilos.

Con el componente `Label` no puede utilizarse la etiqueta HTML `` aunque `Label.html` esté definido en `true`. Por ejemplo, en este caso el texto "Hello" aparece en negro y no en rojo, como haría si pudiera utilizarse ``:

```
lbl.html = true;
lbl.text = "<font color=\"#FF0000\">Hello</font> World";
```

Para recuperar el texto normal de un texto con formato HTML, defina la propiedad `HTML` en `false` y, a continuación, acceda a la propiedad `text`. Esto eliminará el formato HTML. Tal vez desee copiar el texto de la etiqueta en un componente `Label` o `TextArea` de fuera de la pantalla antes de recuperar el texto normal.

Ejemplo

En el ejemplo siguiente se define la propiedad `html` en `true` para que la etiqueta pueda formatearse con HTML. A continuación, la propiedad `text` se define en una cadena que incluye formato con HTML:

```
labelControl.html = true;  
labelControl.text = "The <b>Royal</b> Nonesuch";
```

La palabra “Royal” se muestra en negrita.

Label.text

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

labelInstance.text

Descripción

Propiedad; texto de una etiqueta. El valor predeterminado es "Label".

Ejemplo

El código siguiente define la propiedad `text` de la instancia `labelControl` de `Label` y envía el valor al panel Salida:

```
labelControl.text = "The Royal Nonesuch";  
trace(labelControl.text);
```

Componente List

El componente `List` es un cuadro de lista de desplazamiento de selección única o múltiple. Una lista también puede mostrar gráficos, incluidos otros componentes. Los elementos que se visualizan en el componente `List` se añaden con el cuadro de diálogo Valores que aparece al hacer clic en los campos de parámetros `data` o `labels`. Para añadir elementos a la lista también pueden utilizarse los métodos `List.addItem()` y `List.addItemAt()`.

El componente `List` utiliza un índice con base cero donde el elemento con el índice 0 aparece en primer lugar. Si añade, elimina o reemplaza elementos de lista mediante los métodos y propiedades de la clase `List`, es posible que tenga que especificar el índice del elemento de lista.

El componente `List` se selecciona cuando se hace clic en él o se presiona el tabulador hasta su posición y, a continuación, se pueden utilizar las siguientes teclas para controlarlo:

Tecla	Descripción
Teclas alfanuméricas	Salta al siguiente elemento con <code>Key.getAscii()</code> como el primer carácter de su etiqueta.
Control	Tecla conmutadora. Permite seleccionar y anular la selección de varios elementos no contiguos.

Tecla	Descripción
Flecha abajo	La selección desciende un elemento.
Inicio	La selección se desplaza al principio de la lista.
Av Pág	La selección avanza una página.
Re Pág	La selección retrocede una página.
Mayús	Tecla de selección contigua. Permite seleccionar elementos contiguos.
Flecha arriba	La selección asciende un elemento.

Nota: para las teclas Av Pág y Re Pág, una página está formada por los elementos que caben en pantalla menos uno. Por ejemplo, para avanzar por una lista desplegable que presenta las líneas de diez en diez, la pantalla muestra los elementos 0-9, 9-18, 18-27 y así sucesivamente, solapando un elemento por página.

Para más información sobre el control de la selección, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 24](#) o [“Clase FocusManager” en la página 101](#).

La vista previa dinámica de cada instancia del componente List refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o el panel Inspector de componentes.

Cuando se añade el componente List a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de la pantalla puedan acceder a él. En primer lugar, debe añadir la línea de código siguiente para activar la accesibilidad:

```
mx.accessibility.ListAccImpl.enableAccessibility();
```

La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias. Para más información, consulte [“Creación de contenido accesible” en el apartado Utilización de Flash de la Ayuda](#). Es posible que tenga que actualizar el sistema de Ayuda para consultar esta información.

Utilización del componente List

Utilice una lista cuando el usuario deba elegir una o varias opciones entre muchas. Por ejemplo, un usuario visita un sitio Web de comercio electrónico y necesita elegir el artículo que va a comprar. Hay 30 artículos; el usuario recorre la lista y hace clic en el que desea seleccionar.

También puede diseñar una lista que utilice clips de película personalizados como filas para poder mostrar más información al usuario. Por ejemplo, en una aplicación de correo electrónico, cada buzón podría ser un componente List y cada fila podría tener iconos para indicar la prioridad y el estado.

Parámetros del componente List

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente List en el inspector de propiedades o en el panel Inspector de componentes:

data Matriz de valores que rellenan los datos de la lista. El valor predeterminado es [] (matriz vacía). No hay propiedad en tiempo de ejecución equivalente.

labels Matriz de valores de texto que rellenan los valores de etiqueta de la lista. El valor predeterminado es [] (matriz vacía). No hay propiedad en tiempo de ejecución equivalente.

multipleSelection Valor booleano que indica si pueden seleccionarse varios valores (true) o no (false). El valor predeterminado es false.

rowHeight indica la altura de cada fila, expresada en píxeles. El valor predeterminado es 20. Si se define una fuente, la altura de la fila no cambia.

Puede escribir código ActionScript para definir opciones adicionales para instancias de List con sus métodos, propiedades y eventos. Para más información, consulte [Clase List](#).

Creación de aplicaciones con el componente List

El siguiente procedimiento explica cómo añadir un componente List a una aplicación durante la edición. En este ejemplo, la lista tiene tres elementos.

Para añadir un componente List simple a una aplicación, siga este procedimiento:

- 1 Arrastre un componente List desde el panel Componentes al escenario.
- 2 Seleccione la lista y elija Modificar > Transformar para cambiar su tamaño y adaptarlo a la aplicación.
- 3 En el inspector de propiedades, siga este procedimiento:
 - Introduzca el nombre de instancia **myList**.
 - Introduzca Item1, Item2 y Item3 como parámetro labels.
 - Introduzca item1.html, item2.html, item3.html como parámetro data.
- 4 Elija Control > Probar película para ver la lista con sus elementos.

En la aplicación podría utilizar los valores de la propiedad data para abrir archivos HTML.

El siguiente procedimiento explica cómo añadir un componente List a una aplicación durante la edición. En este ejemplo, la lista tiene tres elementos.

Para añadir un componente List a una aplicación, siga este procedimiento:

- 1 Arrastre un componente List desde el panel Componentes al escenario.
- 2 Seleccione la lista y elija Modificar > Transformar para cambiar su tamaño y adaptarlo a la aplicación.
- 3 En el panel Acciones, introduzca el nombre de instancia **myList**.
- 4 Seleccione el fotograma 1 de la línea de tiempo y, en el panel Acciones, introduzca lo siguiente:
`myList.dataProvider = myDP;`
Si ha definido un proveedor de datos denominado `myDP`, la lista se llenará de datos. Para más información sobre proveedores de datos, consulte [List.dataProvider](#).
- 5 Elija Control > Probar película para ver la lista con sus elementos.

Personalización del componente List

El componente List puede transformarse horizontal y verticalmente durante la edición o en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `List.setSize()` (véase [UIObject.setSize\(\)](#)).

Cuando se cambia el tamaño de una lista, las filas se contraen horizontalmente y recortan el texto que contienen. Verticalmente, la lista añade o elimina las filas necesarias. Las barras de desplazamiento se colocan automáticamente. Para más información sobre barras de desplazamiento, consulte [“Componente ScrollPane” en la página 186](#).

Utilización de estilos con el componente List

Es posible definir propiedades de estilo para cambiar la apariencia de un componente List.

El componente List utiliza los siguientes estilos de Halo:

Estilo	Descripción
<code>alternatingRowColors</code>	Especifica los colores de las filas en un patrón alterno. El valor puede ser una matriz de más de dos colores, por ejemplo, <code>0xFF00FF</code> , <code>0xCC6699</code> y <code>0x996699</code> .
<code>backgroundColor</code>	Color de fondo de la lista. Este estilo se define en una declaración de estilo de clase, <code>ScrollSelectList</code> .
<code>borderColor</code>	Sección negra de un borde tridimensional o sección en color de un borde bidimensional.
<code>borderStyle</code>	Estilo del recuadro de delimitación. Los valores posibles son: "none", "solid", "inset" y "outset". Este estilo se define en una declaración de estilo de clase, <code>ScrollSelectList</code> .
<code>defaultIcon</code>	Nombre del icono predeterminado que se utilizará en las filas de la lista. El valor predeterminado es <code>undefined</code> .
<code>rollOverColor</code>	Color de una fila por la que se desplaza el puntero.
<code>selectionColor</code>	Color de una fila seleccionada.
<code>selectionEasing</code>	Referencia a la ecuación de aceleración (función) utilizada para controlar la interpolación programada.
<code>disabledColor</code>	Color desactivado para el texto.
<code>textRollOverColor</code>	Color del texto cuando el puntero se desplaza sobre él.
<code>textSelectedColor</code>	Color del texto cuando está seleccionado.
<code>selectionDisabledColor</code>	Color de una fila si se ha seleccionado y desactivado.
<code>selectionDuration</code>	Duración de las transiciones al seleccionar elementos.
<code>useRollOver</code>	Determina si el resaltado se activa cuando el puntero se desplaza sobre una fila.

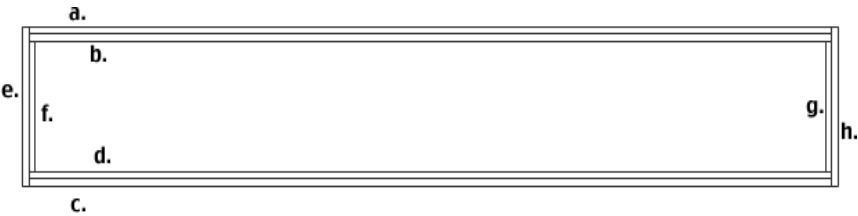
El componente List también utiliza las propiedades de estilo del componente Label (véase [“Utilización de estilos con el componente Label” en la página 109](#)), `ScrollBar` y `RectBorder`.

Utilización de aspectos con el componente List

Todos los aspectos del componente List están incluidos en los subcomponentes que forman la lista ([Componente ScrollPane](#) y [RectBorder](#)). Para más información, consulte “[Componente ScrollPane](#)” en la página 186. Puede utilizar el método `setStyle()` (véase `UIObject.setStyle()`) para cambiar las propiedades de estilo [RectBorder](#) siguientes:

Estilos RectBorder	Posición en el borde
<code>borderColor</code>	a
<code>highlightColor</code>	b
<code>borderColor</code>	c
<code>shadowColor</code>	d
<code>borderCapColor</code>	e
<code>shadowCapColor</code>	f
<code>shadowCapColor</code>	g
<code>borderCapColor</code>	h

Las propiedades de estilo definen las siguientes posiciones en el borde:



Clase List

Herencia `UIObject > UIComponent > View > ScrollView > ScrollSelectList > List`

Namespace de clase de `ActionScript` `mx.controls.List`

El componente List consta de tres partes:

- Elementos
- Filas
- Proveedor de datos

Un elemento es un objeto de `ActionScript` que sirve para almacenar las unidades de información de la lista. Una lista puede concebirse como una matriz; cada espacio indexado de la matriz es un elemento. Un elemento es un objeto que, normalmente, tiene una propiedad `label` que se muestra y una propiedad `data` que se utiliza para almacenar datos.

Una fila es un componente que se utiliza para mostrar un elemento. Las filas las suministra de forma predeterminada la lista (se utiliza la clase `SelectableRow`) o las suministra el usuario, normalmente con una subclase de la clase `SelectableRow`. La clase `SelectableRow` implementa la interfaz `CellRenderer`, un conjunto de propiedades y métodos que permiten que la lista manipule las filas y envíe datos e información de estado (por ejemplo, resaltada, seleccionada, etc.) a la fila que va a mostrarse.

Un proveedor de datos es el modelo de datos de la lista de elementos de una lista. Si en un mismo fotograma hay una lista y una matriz, ésta recibe automáticamente métodos que permiten manipular datos y difundir cambios a varias vistas. Puede crear una instancia de `Array` u obtener una en un servidor y utilizarla como modelo de datos para varios componentes `List`, `ComboBox`, `DataGrid`, etc. El componente `List` posee un conjunto de métodos proxy para su proveedor de datos (por ejemplo, `addItem()` y `removeItem()`). Si no se proporciona ningún proveedor de datos externo a la lista, estos métodos crean automáticamente una instancia de proveedor de datos que se presenta mediante `List.dataProvider`.

Para añadir un componente `List` al orden de tabulación de una aplicación, defina su propiedad `tabIndex` (véase `UIComponent.tabIndex`). El componente `List` utiliza `FocusManager` para sustituir el rectángulo de selección predeterminado de Flash Player por un rectángulo de selección personalizado con esquinas redondeadas. Para más información, consulte “[Creación de un desplazamiento personalizado de la selección](#)” en la página 24.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.controls.List.version);
```

Nota: el código siguiente devuelve undefined: `trace(myListInstance.version);`.

Resumen de métodos de la clase `List`

Método	Descripción
<code>List.addItem()</code>	Añade un elemento al final de la lista.
<code>List.addItemAt()</code>	Añade un elemento a la lista en el índice especificado.
<code>List.getItemAt()</code>	Devuelve el elemento del índice especificado.
<code>List.removeAll()</code>	Elimina todos los elementos de la lista.
<code>List.removeItemAt()</code>	Elimina el elemento en el índice especificado.
<code>List.replaceItemAt()</code>	Sustituye por otro el elemento del índice especificado.
<code>List.setPropertiesAt()</code>	Aplica las propiedades especificadas al elemento especificado.
<code>List.sortItems()</code>	Ordena los elementos de la lista según la función de comparación especificada.
<code>List.sortItemsBy()</code>	Ordena los elementos de la lista según la propiedad especificada.

Hereda todos los métodos de [Clase `UIObject`](#) y [Clase `UIComponent`](#).

Resumen de propiedades de la clase List

Propiedad	Descripción
List.cellRenderer	Asigna el procesador de celdas que va a utilizarse en cada fila de la lista.
List.dataProvider	Origen de los elementos de la lista.
List.hPosition	Posición horizontal de la lista.
List.hScrollPolicy	Indica si la barra de desplazamiento horizontal se muestra ("on") o no ("off").
List.iconField	Campo de cada elemento que se utiliza para especificar iconos.
List.iconFunction	Función que determina qué icono utilizar.
List.labelField	Especifica el campo de cada elemento que se utilizará como texto de etiqueta.
List.labelFunction	Función que determina qué campos de cada elemento se utilizarán para el texto de etiqueta.
List.length	Longitud de la lista en elementos. Es una propiedad de sólo lectura.
List.maxHPosition	Especifica el número de píxeles que la lista puede desplazarse a la derecha cuando List.hScrollPolicy está definida en "on".
List.multipleSelection	Indica si en la lista se permite la selección múltiple (<code>true</code>) o no (<code>false</code>).
List.rowCount	Número de filas que son al menos parcialmente visibles en la lista.
List.rowHeight	Altura de las filas de la lista, expresada en píxeles.
List.selectable	Indica si la lista es seleccionable (<code>true</code>) o no (<code>false</code>).
List.selectedIndex	Índice de una selección en una lista de selección única.
List.selectedIndices	Matriz de los elementos seleccionados en una lista de selección múltiple.
List.selectedItem	Elemento seleccionado en una lista de selección única. Es una propiedad de sólo lectura.
List.selectedItems	Objetos del elemento seleccionado en una lista de selección múltiple. Es una propiedad de sólo lectura.
List.vPosition	Desplaza la lista para que el elemento visible en la parte superior sea el número asignado.
List.vScrollPolicy	Indica si la barra de desplazamiento vertical se muestra ("on"), no se muestra ("off") o se muestra cuando es necesario ("auto").

Hereda todas las propiedades de [Clase UIObject](#) y [Clase UIComponent](#).

Resumen de eventos de la clase List

Evento	Descripción
<code>List.change</code>	Se difunde siempre que la selección cambia por la interacción del usuario.
<code>List.itemRollOut</code>	Se difunde cuando el puntero se desplaza por elementos de la lista y después sale de ellos.
<code>List.itemRollOver</code>	Se difunde cuando el puntero se desplaza por elementos de la lista.
<code>List.scroll</code>	Se difunde cuando se recorre la lista.

Hereda todos los eventos de [Clase UIObject](#) y [Clase UIComponent](#).

List.addItem()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
listInstance.addItem(label[, data])  
listInstance.addItem(itemObject)
```

Parámetros

label Cadena que indica la etiqueta del elemento nuevo.
data Datos del elemento. Este parámetro es optativo y puede ser cualquier tipo de datos.
itemObject Objeto de elemento que normalmente tiene propiedades *label* y *data*.

Valor devuelto

Índice en el que se ha añadido el elemento.

Descripción

Método; añade un elemento nuevo al final de la lista.

En el primer ejemplo de sintaxis, se crea siempre un objeto de elemento con la propiedad *label* especificada y, si se ha especificado, la propiedad *data*.

En el segundo ejemplo de sintaxis, se añade el objeto de elemento especificado.

Cuando se llama a este método, se modifica el proveedor de datos del componente List. Si el proveedor de datos se comparte con otros componentes, éstos también se actualizan.

Ejemplo

Las dos siguientes líneas de código añaden un elemento a la instancia `myList`. Para probar este código, arrastre un componente `List` al escenario y asígnele el nombre de instancia **myList**. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
myList.addItem("esto es un elemento");  
myList.addItem({label:"Gordon",age:"very old",data:123});
```

List.addItemAt()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
listInstance.addItemAt(index, label[, data])  
listInstance.addItemAt(index, itemObject)
```

Parámetros

label Cadena que indica la etiqueta del elemento nuevo.

data Datos del elemento. Este parámetro es optativo y puede ser cualquier tipo de datos.

index Número mayor o igual que cero que indica la posición del elemento.

itemObject Objeto de elemento que normalmente tiene propiedades `label` y `data`.

Valor devuelto

Índice en el que se ha añadido el elemento.

Descripción

Método; añade un elemento nuevo en la posición especificada por el parámetro *index*.

En el primer ejemplo de sintaxis, se crea siempre un objeto de elemento con la propiedad `label` especificada y, si se ha especificado, la propiedad `data`.

En el segundo ejemplo de sintaxis, se añade el objeto de elemento especificado.

Cuando se llama a este método, se modifica el proveedor de datos del componente `List`. Si el proveedor de datos se comparte con otros componentes, éstos también se actualizan.

Ejemplo

La siguiente línea de código añade un elemento en la posición del tercer índice, que es el cuarto elemento de la lista:

```
myList.addItemAt(3,{label:'Red',data:0xFF0000});
```

List.cellRenderer

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
listInstance.cellRenderer
```

Descripción

Propiedad; asigna el procesador de celdas que va a utilizarse en cada fila de la lista. Esta propiedad debe ser una referencia a un objeto de clase o un identificador de vinculación del símbolo que debe utilizar el procesador de celdas. Todas las clases que se utilicen con esta propiedad deben implementar “[Interfaz CellRenderer](#)” en la [página 59](#).

Ejemplo

En el ejemplo siguiente se utiliza un identificador de vinculación para definir un nuevo procesador de celdas:

```
myList.cellRenderer = "ComboBoxCell";
```

List.change

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(change){  
    // aquí código propio  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.change = function(eventObject){  
    // aquí código propio  
}  
listInstance.addEventListener("change", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando el índice seleccionado de la lista cambia como resultado de la interacción del usuario.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente List. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado a la instancia `myBox` del componente List, envía “_level0.myBox” al panel Salida:

```
on(click){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*listInstance*) distribuye un evento (en este caso *change*) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto *Event* (*eventObject*) al método del objeto detector. Cada objeto *Event* tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 236](#).

Finalmente, se llama al método `addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Ejemplo

En el ejemplo siguiente se envía al panel Salida el nombre de instancia del componente que ha generado el evento *change*:

```
form.change = function(eventObj){
    trace("Valor cambiado a " + eventObj.target.value);
}
myList.addEventListener("change", form);
```

Véase también

`UIEventDispatcher.addEventListener()`

List.dataProvider

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

listInstance.dataProvider

Descripción

Propiedad; modelo de datos de los elementos que se ven en una lista. El valor de esta propiedad puede ser una matriz o cualquier objeto que implemente la interfaz *DataProvider*. El valor predeterminado es []. Para más información sobre la interfaz *DataProvider*, consulte [“Componente DataProvider” en la página 95](#).

El componente *List* y otros componentes para datos añaden métodos al prototipo del objeto *Array* para ceñirse a la interfaz *DataProvider*. Por lo tanto, cualquier matriz que exista a la vez que la lista tiene automáticamente todos los métodos necesarios (`addItem()`, `getItemAt()`, etc.) para ser el modelo de datos de la lista y puede utilizarse para difundir cambios de modelo a varios componentes.

Si la matriz contiene objetos, accede a las propiedades `List.labelField` o `List.labelFunction` para determinar qué partes del elemento deben mostrarse. El valor predeterminado es "label", por lo que si existe un campo `label`, es el campo que se muestra; en caso contrario, se muestra una lista de todos los campos separados por comas.

Nota: si la matriz contiene cadenas y no objetos en todos los índices, la lista no puede ordenar los elementos y mantener el estado de la selección. Al ordenar se pierde la selección.

Como proveedor de datos del componente `List` pueden elegirse todas las instancias que implementen la interfaz `DataProvider`. Entre ellas se encuentran los juegos de registros de Flash Remoting, los conjuntos de datos de Firefly, etc.

Ejemplo

Este ejemplo utiliza una matriz de cadenas para rellenar la lista:

```
list.dataProvider = ["Envío estándar", "Aéreo dos días", "Aéreo exprés"];
```

Este ejemplo crea una matriz proveedora de datos y la asigna a la propiedad `dataProvider`:

```
myDP = new Array();
list.dataProvider = myDP;

for (var i=0; i<accounts.length; i++) {
    // estos cambios en DataProvider se difundirán a la lista
    myDP.addItem({ label: accounts[i].name,
                    data: accounts[i].accountID });
}
```

List.getItemAt()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
listInstance.getItemAt(index)
```

Parámetros

index Número mayor o igual que 0 y menor que `List.length`. Índice del elemento que debe recuperarse.

Valor devuelto

Objeto del elemento indexado. Undefined si el índice está fuera de rango.

Descripción

Método; recupera el elemento del índice especificado.

Ejemplo

El código siguiente muestra la etiqueta del elemento de la posición 4 del índice:

```
trace(myList.getItemAt(4).label);
```

List.hPosition

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

listInstance.hPosition

Descripción

Propiedad; desplaza la lista el número de píxeles especificado en sentido horizontal. `hPosition` no puede definirse a menos que el valor de `hScrollPolicy` sea "on" y la lista tenga un `maxHPosition` superior a 0.

Ejemplo

En el ejemplo siguiente se obtiene la posición de desplazamiento horizontal de `myList`:

```
var scrollPos = myList.hPosition;
```

En el ejemplo siguiente se define la posición de desplazamiento horizontal en el extremo izquierdo:

```
myList.hPosition = 0;
```

List.hScrollPolicy

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

listInstance.hScrollPolicy

Descripción

Propiedad; cadena que determina si se muestra o no la barra de desplazamiento horizontal; el valor puede ser "on" u "off". El valor predeterminado es "off". La barra de desplazamiento horizontal no mide el texto; es preciso definir una posición máxima de desplazamiento horizontal, véase [List.maxHPosition](#).

Nota: `List.hScrollPolicy` no admite el valor "auto".

Ejemplo

El código siguiente activa el desplazamiento horizontal de la lista hasta 200 píxeles:

```
myList.hScrollPolicy = "on";  
myList.Box.maxHPosition = 200;
```

Véase también

[List.hPosition](#), [List.maxHPosition](#)

List.iconField

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

listInstance.iconField

Descripción

Propiedad; especifica el nombre del campo que va a utilizarse como identificador de icono. Si el valor del campo es `undefined`, se utiliza el icono predeterminado especificado en el estilo `defaultIcon`. Si el estilo `defaultIcon` es `undefined`, no se utiliza ningún icono.

Ejemplo

En el ejemplo siguiente se define la propiedad `iconField` en la propiedad `icon` de cada elemento:

```
list.iconField = "icon"
```

Véase también

[List.iconFunction](#)

List.iconFunction

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

listInstance.iconFunction

Descripción

Propiedad; especifica la función que va a utilizarse para determinar qué icono utilizará cada fila para mostrar su elemento. Esta función recibe un parámetro *item*, que es el elemento que se va a presentar, y debe devolver una cadena que represente el identificador de símbolo del icono.

Ejemplo

En el ejemplo siguiente se añaden iconos que indican si el archivo es un documento de imagen o de texto. Si el campo `data.fileExtension` contiene un valor "jpg" o "gif", el icono utilizado será "pictureIcon", y así sucesivamente:

```
list.iconFunction = function(item){
    var type = item.data.fileExtension;
    if (type=="jpg" || type=="gif") {
        return "pictureIcon";
    } else if (type=="doc" || type=="txt") {
        return "docIcon";
    }
}
```

List.itemRollOut

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(itemRollOut){
    // aquí código propio
}
```

Sintaxis 2:

```
listenerObject = new Object();
listenerObject.itemRollOut = function(eventObject){
    // aquí código propio
}
listInstance.addEventListener("itemRollOut", listenerObject)
```

Objeto Event

Además de las propiedades estándar del objeto Event, el evento `itemRollOut` tiene una propiedad adicional: `index`. `index` es el número del elemento del que ha salido el puntero.

Descripción

Evento; se difunde a todos los detectores registrados cuando el puntero sale de los elementos de la lista.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente List. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado a la instancia `myList` del componente List, envía “_level0.myList” al panel Salida:

```
on(itemRollOut){
    trace(this);
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*listInstance*) distribuye un evento (en este caso *itemRollOut*) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto Event (*eventObject*) al método del objeto detector. Cada objeto Event tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 236](#).

Ejemplo

En el ejemplo siguiente se envía un mensaje al panel Salida para indicar por qué número de índice de elemento ha pasado el puntero:

```
form.itemRollOut = function (eventObj) {
    trace("Elemento n.º" + eventObj.index + " de donde ha salido el puntero.");
}
myList.addEventListener("itemRollOut", form);
```

Véase también

[List.itemRollOver](#)

List.itemRollOver

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(itemRollOver){
    // aquí código propio
}
```

Sintaxis 2:

```
listenerObject = new Object();
listenerObject.itemRollOver = function(eventObject){
    // aquí código propio
}
listInstance.addEventListener("itemRollOver", listenerObject)
```

Objeto Event

Además de las propiedades estándar del objeto Event, el evento *itemRollOver* tiene una propiedad adicional: *index*. *index* es el número del elemento sobre el que ha pasado el puntero.

Descripción

Evento; se difunde a todos los detectores registrados cuando el puntero se desplaza por los elementos de la lista.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente `List`. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado a la instancia `myList` del componente `List`, envía “_level0.myList” al panel Salida:

```
on(itemRollOver){
  trace(this);
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*listInstance*) distribuye un evento (en este caso `itemRollOver`) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (*eventObject*) al método del objeto detector. Cada objeto `Event` tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en [la página 236](#).

Ejemplo

En el ejemplo siguiente se envía un mensaje al panel Salida para indicar por qué número de índice de elemento ha pasado el puntero:

```
form.itemRollOver = function (eventObj) {
  trace("Elemento n.º" + eventObj.index + " por donde ha pasado el puntero.");
}
myList.addEventListener("itemRollOver", form);
```

Véase también

[List.itemRollOut](#)

List.labelField

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

listInstance.labelField

Descripción

Propiedad; especifica un campo en cada elemento para utilizarlo como texto que se muestra. Esta propiedad toma el valor del campo y lo utiliza como etiqueta. El valor predeterminado es "label".

Ejemplo

En el ejemplo siguiente se define la propiedad `labelField` en el campo "name" de cada elemento. "Nina" aparecería como la etiqueta del elemento añadido en la segunda línea de código:

```
list.labelField = "name";  
list.addItem({name: "Nina", age: 25});
```

Véase también

[List.labelFunction](#)

List.labelFunction

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

listInstance.labelFunction

Descripción

Propiedad; especifica la función que se debe utilizar para decidir qué campo (o combinación de campos) va a mostrarse de cada elemento. Esta función recibe un parámetro *item*, que es el elemento que se va a presentar, y debe devolver una cadena que represente el texto que va a mostrarse.

Ejemplo

En el ejemplo siguiente, la etiqueta muestra algunos detalles formateados de los elementos:

```
list.labelFunction = function(item){  
    return "El precio del producto " + item.productID + ", " + item.productName  
        + " es EUR"  
        + item.price;  
}
```

Véase también

[List.labelField](#)

List.length

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

listInstance.length

Descripción

Propiedad (sólo lectura); número de elementos de la lista.

Ejemplo

En el ejemplo siguiente se coloca el valor de `length` en una variable:

```
var len = myList.length;
```

List.maxHPosition

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

listInstance.maxHPosition

Descripción

Propiedad; especifica el número de píxeles que puede desplazarse la lista cuando [List.hScrollPolicy](#) está definida en "on". La lista no mide con precisión la anchura del texto que contiene. Es preciso definir `maxHPosition` para indicar la cantidad de desplazamiento necesario. Si no se define la propiedad, la lista no se desplazará horizontalmente.

Ejemplo

En el ejemplo siguiente se crea una lista con 400 píxeles de desplazamiento horizontal:

```
myList.hScrollPolicy = "on";  
myList.maxHPosition = 400;
```

Véase también

[List.hScrollPolicy](#)

List.multipleSelection

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

listInstance.multipleSelection

Descripción

Propiedad; indica si se permite la selección múltiple (`true`) o única (`false`). El valor predeterminado es `false`.

Ejemplo

En el ejemplo siguiente se hace una prueba para determinar si pueden seleccionarse varios elementos:

```
if (myList.multipleSelection){  
    // aquí código propio  
}
```

El ejemplo siguiente permite realizar varias selecciones en la lista:

```
myList.selectMultiple = true;
```

List.removeAll()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
listInstance.removeAll()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; elimina todos los elementos de la lista.

Cuando se llama a este método, se modifica el proveedor de datos del componente List. Si el proveedor de datos se comparte con otros componentes, éstos también se actualizan.

Ejemplo

El código siguiente borra la lista:

```
myList.removeAll();
```

List.removeItemAt()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
listInstance.removeItemAt(index)
```

Parámetros

index Cadena que indica la etiqueta del elemento nuevo. Valor mayor que cero y menor que `List.length`.

Valor devuelto

Objeto; elemento eliminado (undefined si no hay elementos).

Descripción

Método; elimina el elemento de la posición especificada en *index*. Los índices de la lista a partir del índice que indica el parámetro *index* se contraen una posición.

Cuando se llama a este método, se modifica el proveedor de datos del componente List. Si el proveedor de datos se comparte con otros componentes, éstos también se actualizan.

Ejemplo

El código siguiente elimina el elemento de la posición 3 del índice:

```
myList.removeItemAt(3);
```

List.replaceItemAt()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
listInstance.replaceItemAt(index, label[, data])  
listInstance.replaceItemAt(index, itemObject)
```

Parámetros

index Número mayor que cero y menor que `List.length` que indica la posición en la que insertar el elemento (índice del elemento nuevo).

label Cadena que indica la etiqueta del elemento nuevo.

data Datos del elemento. Este parámetro es optativo y puede ser de cualquier tipo.

itemObject. Objeto que se utiliza como el elemento, que normalmente contiene las propiedades `label` y `data`.

Valor devuelto

Ninguno.

Descripción

Método; sustituye el contenido del elemento del índice que especifica el parámetro *index*.

Cuando se llama a este método, se modifica el proveedor de datos del componente List. Si el proveedor de datos se comparte con otros componentes, éstos también se actualizan.

Ejemplo

En el ejemplo siguiente se cambia la posición del cuarto índice:

```
myList.replaceItemAt(3, "new label");
```

List.rowCount

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
listInstance.rowCount
```

Descripción

Propiedad; número de filas que son al menos parcialmente visibles en la lista. Resulta útil cuando la escala de la lista está expresada en píxeles y hace falta contar las filas. Por el contrario, si se define el número de filas, se muestra un número exacto de filas sin que aparezca la fila final partida.

El código `myList.rowCount = num` es equivalente al código `myList.setSize(myList.width, h)` (donde `h` es la altura necesaria para mostrar el número `num` de elementos).

El valor predeterminado se basa en la altura de la lista definida durante la edición o con el método `list.setSize()` (véase [UIObject.setSize\(\)](#)).

Ejemplo

En el ejemplo siguiente se revela el número de elementos visibles de una lista:

```
var rowCount = myList.rowCount;
```

En el ejemplo siguiente, la lista muestra cuatro elementos:

```
myList.rowCount = 4;
```

Este ejemplo elimina la fila parcial del final de la lista, si la hay:

```
myList.rowCount = myList.rowCount;
```

Este ejemplo define una lista con el número mínimo de filas que puede mostrar por completo:

```
myList.rowCount = 1;  
trace("myList tiene "+myList.rowCount+" filas");
```

List.rowHeight

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
listInstance.rowHeight
```

Descripción

Propiedad; altura de cada fila de la lista, expresada en píxeles. Las filas no aumentan de tamaño para que encaje la fuente configurada, por lo que definir la propiedad `rowHeight` es la mejor manera de asegurar que los elementos se muestren por completo. El valor predeterminado es 20.

Ejemplo

En el ejemplo siguiente se definen las filas en 30 píxeles:

```
myList.rowHeight = 30;
```

List.scroll

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(scroll){  
    // aquí código propio  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.scroll = function(eventObject){  
    // aquí código propio  
}  
listInstance.addEventListener("scroll", listenerObject)
```

Objeto Event

Además de las propiedades estándar del objeto Event, el evento `scroll` tiene una propiedad adicional, `direction`. Es una cadena con dos valores posibles: "horizontal" o "vertical". En los eventos `scroll` de `ComboBox`, el valor es siempre "vertical".

Descripción

Evento; se difunde a todos los detectores registrados cuando se recorre la lista.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente `List`. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado a la instancia `myList` del componente `List`, envía “_level0.myList” al panel Salida:

```
on(scroll){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*listInstance*) distribuye un evento (en este caso `scroll`) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto Event (*eventObject*) al método del objeto detector. Cada objeto Event tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en [la página 236](#).

Ejemplo

En el ejemplo siguiente se envía al panel Salida el nombre de instancia del componente que ha generado el evento `change`:

```
form.scroll = function(eventObj){
    trace("lista recorrida");
}
myList.addEventListener("scroll", form);
```

List.selectable

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

listInstance.selectable

Descripción

Propiedad; valor booleano que indica si la lista es seleccionable (`true`) o no (`false`). El valor predeterminado es `true`.

List.selectedIndex

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

listInstance.selectedIndex

Descripción

Propiedad; índice seleccionado en una lista de selección única. El valor es undefined si no se selecciona nada; el valor es igual al último elemento seleccionado si se seleccionan varios. Si se asigna un valor a `selectedIndex`, se borra la selección actual y se selecciona el elemento indicado.

Ejemplo

En este ejemplo se selecciona un elemento después del elemento seleccionado actualmente. Si no se selecciona nada, se selecciona el elemento 0:

```
var selIndex = myList.selectedIndex;
myList.selectedIndex = (selIndex==undefined ? 0 : selIndex+1);
```

Véase también

[List.selectedIndexes](#), [List.selectedItem](#), [List.selectedItems](#)

List.selectedIndexes

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

listInstance.selectedIndex

Descripción

Propiedad; matriz de índices de los elementos seleccionados. Si se asigna esta propiedad, se sustituye la selección actual. Si `selectedIndex` se define en una matriz de longitud 0 (o de valor no definido), se borra la selección actual. El valor es undefined si no se selecciona nada.

La propiedad `selectedIndex` muestra la lista en el orden en el que se han seleccionado los elementos. Si se hace clic en el segundo elemento, después en el tercero y después en el primero, `selectedIndex` devuelve [1,2,0].

Ejemplo

En el ejemplo siguiente se obtienen los índices seleccionados:

```
var selIndices = myList.selectedIndexes;
```

En el ejemplo siguiente se seleccionan cuatro elementos:

```
var myArray = new Array (1,4,5,7);
myList.selectedIndexes = myArray;
```

Véase también

[List.selectedIndex](#), [List.selectedItem](#), [List.selectedItems](#)

List.selectedItem

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

listInstance.selectedItem

Descripción

Propiedad (sólo lectura); objeto de elemento de una lista de selección única. (En una lista de selección múltiple con varios elementos seleccionados, `selectedItem` devuelve el elemento seleccionado en último lugar). Si no hay selección, el valor es `undefined`.

Ejemplo

En este ejemplo se muestra la etiqueta seleccionada:

```
trace(myList.selectedItem.label);
```

Véase también

[List.selectedIndex](#), [List.selectedIndices](#), [List.selectedItems](#)

List.selectedItems

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

listInstance.selectedItems

Descripción

Propiedad (sólo lectura); matriz de objetos del elemento seleccionado. En una lista de selección múltiple, `selectedItems` permite acceder a los objetos del conjunto de elementos seleccionados.

Ejemplo

En el ejemplo siguiente se obtiene la matriz de objetos de elemento seleccionados:

```
var myObjArray = myList.selectedItems;
```

Véase también

[List.selectedIndex](#), [List.selectedItem](#), [List.selectedIndices](#)

List.setPropertiesAt()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
listInstance.setPropertiesAt(index, styleObj)
```

Parámetros

index Número mayor que cero o menor que `List.length` que indica el índice del elemento que va a cambiarse.

styleObj Objeto que enumera las propiedades y valores que van a definirse.

Valor devuelto

Ninguno.

Descripción

Método; aplica las propiedades especificadas en el parámetro *styleObj* al elemento especificado en el parámetro *index*. Las propiedades admitidas son `icon` y `backgroundColor`.

Ejemplo

En el ejemplo siguiente se cambia el color del cuarto elemento a negro y se le asigna un icono:

```
myList.setPropertiesAt(3, {backgroundColor:0x000000, icon: "file"});
```

List.sortItems()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
listInstance.sortItems(compareFunc)
```

Parámetros

compareFunc Referencia a una función. Esta función se utiliza para comparar dos elementos y determinar su orden de clasificación.

Para más información, consulte `Array.sort()` en el apartado Diccionario de ActionScript de la Ayuda.

Valor devuelto

Índice en el que se ha añadido el elemento.

Descripción

Método; ordena los elementos de la lista según el parámetro *compareFunc*.

Ejemplo

En el ejemplo siguiente se ordenan los elementos por etiquetas en mayúsculas. Observe que los parámetros *a* y *b* que pasan a la función son elementos que tienen las propiedades `label` y `data`:

```
myList.sortItems(upperCaseFunc);
function upperCaseFunc(a,b){
    return a.label.toUpperCase() > b.label.toUpperCase();
}
```

Véase también

[List.sortItemsBy\(\)](#)

List.sortItemsBy()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
listInstance.sortItemsBy(fieldName, order)
```

Parámetros

fieldName Cadena que especifica el nombre de la propiedad que va a utilizarse para ordenar. Normalmente, este valor es "label" o "data".

order Cadena que especifica si los elementos deben clasificarse en orden ascendente ("ASC") o descendente ("DESC").

Valor devuelto

Ninguno.

Descripción

Método; clasifica los elementos de la lista alfabéticamente o numéricamente y en el orden especificado con la propiedad *fieldName*. Si los elementos de *fieldName* son una combinación de cadenas de texto y enteros, los enteros aparecen en primer lugar. El parámetro *fieldName* suele ser "label" o "data", pero puede especificarse cualquier valor de datos primitivos.

Ejemplo

En el código siguiente, los elementos de la lista `surnameMenu` se clasifican en orden ascendente por etiquetas:

```
surnameMenu.sortItemsBy("label", "ASC");
```

Véase también

[List.sortItems\(\)](#)

List.vPosition

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

listInstance.vPosition

Descripción

Propiedad; ordena la lista de tal forma que el índice es el elemento visible en la parte superior. Si el índice está fuera de límite, se utiliza el índice más próximo que esté dentro del límite. El valor predeterminado es 0.

Ejemplo

En el ejemplo siguiente se define la posición de la lista en el elemento del primer índice:

```
myList.vPosition = 0;
```

List.vScrollPolicy

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

listInstance.vScrollPolicy

Descripción

Propiedad; cadena que determina si la lista admite desplazamiento vertical o no. Esta propiedad puede tener uno de estos valores: "on", "off" o "auto". El valor "auto" muestra la barra de desplazamiento cuando es necesario.

Ejemplo

En el ejemplo siguiente se desactiva la barra de desplazamiento:

```
myList.vScrollPolicy = "off";
```

Aun así, puede crear desplazamiento con [List.vPosition](#).

Véase también

[List.vPosition](#)

Componente Loader

El componente Loader es un contenedor que puede mostrar archivos SWF o JPEG. Se puede adaptar el tamaño del contenido al contenedor o cambiar el tamaño del contenedor para que quepa el contenido. De forma predeterminada, es el contenido el que se ajusta al componente Loader. También se puede cargar el contenido en tiempo de ejecución y controlar el progreso de carga.

El componente Loader no se puede seleccionar. Sin embargo, sí se puede seleccionar el contenido cargado en el componente e interactuar con él. Para más información sobre el control de la selección, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 24](#) o [“Clase FocusManager” en la página 101](#).

La vista previa dinámica de cada instancia de Loader refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o el panel Inspector de componentes.

Es posible activar la accesibilidad del contenido cargado en un componente Loader. En tal caso, utilice el panel Accesibilidad para que los lectores de pantalla puedan acceder al mismo. Para más información, consulte [“Creación de contenido accesible” en el apartado Utilización de Flash de la Ayuda](#). Es posible que tenga que actualizar el sistema de Ayuda para consultar esta información.

Utilización del componente Loader

Utilice un componente Loader siempre que necesite captar contenido de una ubicación remota y traerlo a una aplicación de Flash. Por ejemplo, puede utilizar un componente Loader para añadir el logotipo de una empresa (archivo JPEG) a un formulario. También puede utilizar un componente Loader para aprovechar trabajo ya terminado en Flash. Por ejemplo, si ya ha creado una aplicación de Flash y desea ampliarla, puede utilizar el componente Loader para traer la aplicación antigua a una nueva, quizá como sección de una interfaz de fichas. En otro ejemplo, puede utilizar el componente Loader en una aplicación que muestra fotografías. Utilice [Loader.load\(\)](#), [Loader.percentLoaded](#) y [Loader.complete](#) para controlar el tiempo de carga de imágenes y ver barras de progreso durante la carga.

Parámetros del componente Loader

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente Loader en el inspector de propiedades o el panel Inspector de componentes:

autoload indica si el contenido debe cargarse automáticamente (true) o si es preciso esperar hasta que se llame al método [Loader.load\(\)](#) (false). El valor predeterminado es true.

content es una URL absoluta o relativa que indica el archivo que va a cargarse en el componente Loader. La ruta relativa debe hacer referencia al archivo SWF que carga el contenido. La dirección URL debe estar en el mismo subdominio que la URL donde reside actualmente el contenido de Flash. Todos los archivos SWF, ya sea para utilizarlos en Flash Player o para realizar pruebas en el modo de prueba de película, deben estar almacenados en la misma carpeta, y los nombres de archivo no pueden incluir especificaciones de carpeta ni de unidad de disco. El valor predeterminado es undefined hasta que se inicia la carga.

scaleContent indica si el contenido se redimensiona para adaptarse al componente Loader (true) o si el componente Loader se redimensiona para adaptarse al contenido (false). El valor predeterminado es true.

Puede escribir código ActionScript para definir opciones adicionales para instancias de Loader con sus métodos, propiedades y eventos. Para más información, consulte [Clase Loader](#).

Creación de aplicaciones con el componente Loader

El siguiente procedimiento explica cómo añadir un componente Loader a una aplicación durante la edición. En este ejemplo, el componente Loader carga un logotipo JPEG de una URL imaginaria.

Para crear una aplicación con el componente Loader, siga este procedimiento:

- 1 Arrastre un componente Loader desde el panel Componentes al escenario.
- 2 Seleccione un componente Loader en el escenario y utilice la herramienta Transformación libre para adaptar su tamaño a las dimensiones del logotipo empresarial.
- 3 En el inspector de propiedades, introduzca el nombre de instancia **logo**.
- 4 Seleccione el componente Loader en el escenario y siga este procedimiento en el panel Inspector de componentes:
 - Introduzca `http://corp.com/websites/logo/corplego.jpg` en el parámetro `contentPath`.

Personalización del componente Loader

El componente Loader puede transformarse horizontal y verticalmente durante la edición o en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `setSize()` (véase [UIObject.setSize\(\)](#)).

El comportamiento de ajuste de tamaño del componente Loader se controla con la propiedad `scaleContent`. Cuando `scaleContent = true`, el contenido se redimensiona para adaptarse a los límites del componente Loader (y vuelve a redimensionarse cuando se llama al método [UIObject.setSize\(\)](#)). Cuando la propiedad es `scaleContent = false`, el tamaño del componente se ajusta al tamaño del contenido y el método [UIObject.setSize\(\)](#) no tiene efecto alguno.

Utilización de estilos con el componente Loader

El componente Loader no utiliza estilos.

Utilización de aspectos con el componente Loader

El componente Loader utiliza `RectBorder`, que a su vez utiliza la interfaz API de dibujo de ActionScript. Puede utilizar el método `setStyle()` (véase [UIObject.setStyle\(\)](#)) para cambiar las propiedades de estilo `RectBorder` siguientes:

Estilos `RectBorder`

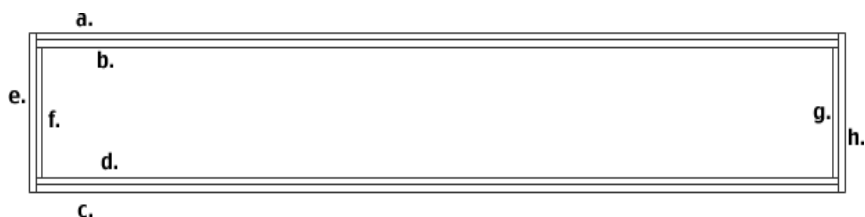
```
borderColor
highlightColor
borderColor
shadowColor
borderCapColor
shadowCapColor
```

Estilos RectBorder

shadowCapColor

borderCapColor

Las propiedades de estilo definen las siguientes posiciones en el borde:



Clase Loader

Herencia UIObject > UIComponent > View > Loader

Namespace de clase de ActionScript mx.controls.Loader

Las propiedades de la clase Loader permiten definir el contenido que va a cargarse y controlar el progreso de carga en tiempo de ejecución.

Si una propiedad de la clase Loader se define con ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o el panel Inspector de componentes.

Para más información, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 24](#).

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.controls.Loader.version);
```

Nota: el código siguiente devuelve undefined: `trace(myLoaderInstance.version);`.

Resumen de métodos de la clase Loader

Método	Descripción
<code>Loader.load()</code>	Carga el contenido especificado en la propiedad <code>contentPath</code> .

Hereda todos los métodos de [Clase UIObject](#) y [Clase UIComponent](#).

Resumen de propiedades de la clase Loader

Propiedad	Descripción
<code>Loader.autoLoad</code>	Valor booleano que indica si el contenido se carga automáticamente (<code>true</code>) o si es preciso llamar a <code>Loader.load()</code> (<code>false</code>).
<code>Loader.bytesLoaded</code>	Propiedad de sólo lectura que indica el número de bytes que se han cargado.

Propiedad	Descripción
<code>Loader.bytesTotal</code>	Propiedad de sólo lectura que indica el número total de bytes del contenido.
<code>Loader.content</code>	Referencia al contenido especificado en la propiedad <code>Loader.contentPath</code> . Es una propiedad de sólo lectura.
<code>Loader.contentPath</code>	Cadena que indica la URL del contenido que va a cargarse.
<code>Loader.percentLoaded</code>	Número que indica el porcentaje de contenido cargado. Es una propiedad de sólo lectura.
<code>Loader.scaleContent</code>	Valor booleano que indica si el contenido se redimensiona para adaptarse al componente Loader (<code>true</code>) o si el componente Loader se redimensiona para adaptarse al contenido (<code>false</code>).

Hereda todas las propiedades de [Clase UIObject](#) y [Clase UIComponent](#).

Resumen de eventos de la clase Loader

Evento	Descripción
<code>Loader.complete</code>	Se activa cuando el contenido termina de cargarse.
<code>Loader.progress</code>	Se activa mientras se carga el contenido.

Hereda todas las propiedades de [Clase UIObject](#) y [Clase UIComponent](#)

Loader.autoLoad

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
loaderInstance.autoLoad
```

Descripción

Propiedad; valor booleano que indica si el contenido se carga automáticamente (`true`) o si es preciso esperar a que se llame a `Loader.load()` (`false`). El valor predeterminado es `true`.

Ejemplo

El código siguiente define que el componente Loader espere a que se llame a `Loader.load()`:

```
loader.autoLoad = false;
```

Loader.bytesLoaded

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

loaderInstance.bytesLoaded

Descripción

Propiedad (sólo lectura); número de bytes del contenido cargado. El valor predeterminado es 0 hasta que el contenido comienza a cargarse.

Ejemplo

El código siguiente crea un componente `ProgressBar` y un componente `Loader`. A continuación crea un objeto detector con un controlador de eventos `progress` que muestra el progreso de la carga. El detector se registra en la instancia de `Loader` de este modo:

```
createClassObject(mx.controls.ProgressBar, "pBar", 0);
createClassObject(mx.controls.Loader, "loader", 1);
loadListener = new Object();
loadListener.progress = function(eventObj){
    // eventObj.target es el componente que ha generado el evento change,
    // es decir, Loader.
    pBar.setProgress(loader.bytesLoaded, loader.bytesTotal); // mostrar progreso
}
loader.addEventListener("progress", loadListener);
loader.content = "logo.swf";
```

Cuando se crea una instancia con el método `createClassObject()`, es preciso colocarla en el escenario con los métodos `move()` y `setSize()`. Véase [UIObject.move\(\)](#) y [UIObject.setSize\(\)](#).

Véase también

[Loader.bytesTotal](#), [UIObject.createClassObject\(\)](#)

Loader.bytesTotal

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

loaderInstance.bytesTotal

Descripción

Propiedad (sólo lectura); tamaño del contenido en bytes. El valor predeterminado es 0 hasta que el contenido comienza a cargarse.

Ejemplo

El código siguiente crea un componente `ProgressBar` y un componente `Loader`. A continuación crea un objeto detector de carga con un controlador de eventos `progress` que muestra el progreso de la carga. El detector se registra en la instancia de `Loader` de este modo:

```

createClassObject(mx.controls.ProgressBar, "pBar", 0);
createClassObject(mx.controls.Loader, "loader", 1);
loadListener = new Object();
loadListener.progress = function(eventObj){
    // eventObj.target es el componente que ha generado el evento change,
    // es decir, Loader.
    pBar.setProgress(loader.bytesLoaded, loader.bytesTotal); // mostrar progreso
}
loader.addEventListener("progress", loadListener);
loader.content = "logo.swf";

```

Véase también

[Loader.bytesLoaded](#)

Loader.complete

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```

on(complete){
    ...
}

```

Sintaxis 2:

```

listenerObject = new Object();
listenerObject.complete = function(eventObject){
    ...
}
loaderInstance.addEventListener("complete", listenerObject)

```

Descripción

Evento; se difunde a todos los detectores registrados cuando el contenido ha terminado de cargarse.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente `Loader`. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado a la instancia `myLoaderComponent` del componente `Loader`, envía “_level0.myLoaderComponent” al panel Salida:

```

on(complete){
    trace(this);
}

```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*loaderInstance*) distribuye un evento (en este caso *complete*) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto Event (*eventObject*) al método del objeto detector. Cada objeto Event tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en [la página 236](#).

Ejemplo

En el ejemplo siguiente se crea un componente Loader y después se define un objeto detector con un controlador de eventos *complete* que define la propiedad *visible* del componente Loader en *true*:

```
createClassObject(mx.controls.Loader, "loader", 0);
loadListener = new Object();
loadListener.complete = function(eventObj){
    loader.visible = true;
}
loader.addEventListener("complete", loadListener);
loader.contentPath = "logo.swf";
```

Loader.content

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

loaderInstance.content

Descripción

Propiedad (sólo lectura); referencia al contenido del componente Loader. El valor es *undefined* hasta que comienza la carga.

Véase también

[Loader.contentPath](#)

Loader.contentPath

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

loaderInstance.contentPath

Descripción

Propiedad; cadena que indica la URL absoluta o relativa del archivo que va a cargarse en el componente Loader. La ruta relativa debe hacer referencia al archivo SWF que carga el contenido. La URL debe pertenecer al mismo subdominio que la URL del archivo SWF que va a cargarse.

Todos los archivos SWF, ya sea para utilizarlos en Flash Player o para realizar pruebas en el modo de prueba de película, deben almacenarse en la misma carpeta, y los nombres de archivo no pueden incluir información de carpeta ni de unidad de disco.

Ejemplo

En el ejemplo siguiente se indica a la instancia de Loader que muestre el contenido del archivo "logo.swf":

```
loader.contentPath = "logo.swf";
```

Loader.load()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

loaderInstance.load(path)

Parámetros

path Parámetro optativo que especifica el valor de la propiedad *contentPath* antes de iniciarse la carga. Si no se especifica ningún valor, se utiliza el valor actual de *contentPath*.

Valor devuelto

Ninguno.

Descripción

Método; indica al componente Loader que comience a cargar su contenido.

Ejemplo

El código siguiente crea una instancia de Loader y define la propiedad *autoload* en *false* para que el componente espere a que se llame al método *load()* para comenzar a cargar el contenido. A continuación llama a *load()* e indica el contenido que debe cargarse:

```
createClassObject(mx.controls.Loader, "loader", 0);  
loader.autoload = false;  
loader.load("logo.swf");
```

Loader.percentLoaded

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

loaderInstance.percentLoaded

Descripción

Propiedad (sólo lectura); número que indica el porcentaje de contenido cargado. Normalmente, esta propiedad se utiliza para presentar el progreso de una forma de lectura fácil para el usuario. Utilice el siguiente código para redondear la cifra al número entero más cercano:

```
Math.round(bytesLoaded/bytesTotal*100))
```

Ejemplo

En el ejemplo siguiente se crea una instancia de Loader y, a continuación, un objeto detector con un controlador de progreso que averigua el porcentaje cargado y lo envía al panel Salida:

```
createClassObject(Loader, "loader", 0);
loadListener = new Object();
loadListener.progress = function(eventObj){
    // eventObj.target es el componente que ha generado el evento change,
    // es decir, Loader.
    trace("logo.swf está " + loader.percentLoaded + "% cargado."); // llevar
    seguimiento de progreso de carga
}
loader.addEventListener("complete", loadListener);
loader.content = "logo.swf";
```

Loader.progress

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(progress){
    ...
}
```

Sintaxis 2:

```
listenerObject = new Object();
listenerObject.progress = function(eventObject){
    ...
}
loaderInstance.addEventListener("progress", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados mientras se carga el contenido. Este evento se activa cuando la carga se activa con el parámetro `autoload` o llamando a `Loader.load()`. El evento `progress` no siempre se difunde. El evento `complete` puede difundirse sin que se distribuyan eventos `progress`. Esto sucede especialmente cuando el contenido cargado es un archivo local.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente `Loader`. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado a la instancia `myLoaderComponent` del componente `Loader`, envía “_level0.myLoaderComponent” al panel Salida:

```
on(progress){
    trace(this);
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*loaderInstance*) distribuye un evento (en este caso `progress`) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (*eventObject*) al método del objeto detector. Cada objeto `Event` tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en la página 236.

Ejemplo

El código siguiente crea una instancia de `Loader` y después un objeto detector con un controlador de eventos para el evento `progress` que envía un mensaje al panel Salida con el porcentaje de contenido cargado:

```
createClassObject(mx.controls.Loader, "loader", 0);
loadListener = new Object();
loadListener.progress = function(eventObj){
    // eventObj.target es el componente que ha generado el evento change,
    // es decir, Loader.
    trace("logo.swf está " + loader.percentLoaded + "% cargado."); // llevar
    seguimiento de progreso de carga
}
loader.addEventListener("progress", loadListener);
loader.contentPath = "logo.swf";
```

Loader.scaleContent

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
loaderInstance.scaleContent
```

Descripción

Propiedad; indica si el contenido se redimensiona para adaptarse al componente Loader (true) o si el componente Loader se redimensiona para adaptarse al contenido (false). El valor predeterminado es true.

Ejemplo

El código siguiente indica al componente Loader que cambie de tamaño para ajustarse al tamaño de su contenido:

```
loader.strechContent = false;
```

Componente MediaController

Para obtener información reciente sobre esta función, haga clic en el botón Actualizar en la parte superior de la ficha Ayuda.

Componente MediaDisplay

Para obtener información reciente sobre esta función, haga clic en el botón Actualizar en la parte superior de la ficha Ayuda.

Componente MediaPlayer

Para obtener información reciente sobre esta función, haga clic en el botón Actualizar en la parte superior de la ficha Ayuda.

Componente Menu

Para obtener información reciente sobre esta función, haga clic en el botón Actualizar en la parte superior de la ficha Ayuda.

Componente MenuBar

Para obtener información reciente sobre esta función, haga clic en el botón Actualizar en la parte superior de la ficha Ayuda.

Componente NumericStepper

El componente NumericStepper permite recorrer un conjunto ordenado de números. El componente muestra un número junto a pequeños botones de flecha arriba y flecha abajo. Cuando el usuario presiona estos botones, el número aumenta o disminuye de forma gradual. Si el usuario hace clic en uno de los botones de flecha, el número aumenta o disminuye según el valor del parámetro `stepSize`, hasta que el usuario suelta el ratón o hasta que se alcanza el valor máximo o mínimo.

El componente NumericStepper sólo gestiona datos numéricos. Además, para ver más de dos lugares numéricos (por ejemplo, los números 5246 o 1,34), es preciso cambiar el tamaño del componente durante la edición.

Un componente `NumericStepper` puede estar activado o desactivado en una aplicación. Si está desactivado, no recibe la entrada del ratón ni del teclado. Si está activado y la selección interna está definida en el cuadro de texto, se selecciona al presionar el tabulador hasta su posición o al hacer clic en él. Cuando una instancia de `NumericStepper` está seleccionada, es posible utilizar las siguientes teclas para controlarla:

Tecla	Descripción
Flecha abajo	El valor cambia en una unidad.
Izquierda	Desplaza el punto de inserción a la izquierda dentro del cuadro de texto.
Derecha	Desplaza el punto de inserción a la derecha dentro del cuadro de texto.
Mayúsculas + Tabulador	Desplaza la selección al objeto anterior.
Tabulador	Desplaza la selección al objeto siguiente.
Flecha arriba	El valor cambia en una unidad.

Para más información sobre el control de la selección, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 24](#) o [“Clase `FocusManager`” en la página 101](#).

La vista previa dinámica de cada instancia de `NumericStepper` refleja el valor indicado en el parámetro `value` durante la edición con el inspector de propiedades o el panel Inspector de componentes. Sin embargo, en la vista previa dinámica ni el teclado ni el ratón pueden interactuar con los botones de `NumericStepper`.

Cuando se añade el componente `NumericStepper` a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de pantalla puedan acceder al mismo. En primer lugar, debe añadir la línea de código siguiente para activar la accesibilidad:

```
mx.accessibility.NumericStepperAccImpl.enableAccessibility();
```

La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias. Para más información, consulte [“Creación de contenido accesible” en el apartado Utilización de Flash de la Ayuda](#). Es posible que tenga que actualizar el sistema de Ayuda para consultar esta información.

Utilización del componente `NumericStepper`

Utilice `NumericStepper` siempre que desee que el usuario seleccione un valor numérico. Por ejemplo, puede utilizar un componente `NumericStepper` en un formulario para que el usuario indique la fecha de caducidad de su tarjeta de crédito. En otro ejemplo, puede utilizar un componente `NumericStepper` para que el usuario aumente o disminuya el tamaño de la fuente.

Parámetros de `NumericStepper`

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente `NumericStepper` en el inspector de propiedades o el panel Inspector de componentes:

value define el valor del paso actual. El valor predeterminado es 0.

minimum define el valor mínimo del paso. El valor predeterminado es 0.

maximum define el valor máximo del paso. El valor predeterminado es 10.

stepSize define la unidad de cambio de paso. El valor predeterminado es 1.

Puede escribir código `ActionScript` para controlar éstas y otras opciones adicionales para los componentes `NumericStepper` utilizando sus propiedades, métodos y eventos. Para más información, consulte [Clase `NumericStepper`](#).

Creación de aplicaciones con el componente `NumericStepper`

El siguiente procedimiento explica cómo añadir un componente `NumericStepper` a una aplicación durante la edición. En este ejemplo, el componente `NumericStepper` permite que el usuario seleccione una película de entre 0 y 5 estrellas con incrementos de media estrella.

Para crear una aplicación con el componente `Window`, siga este procedimiento:

- 1 Arrastre un componente `NumericStepper` desde el panel Componentes al escenario.
- 2 En el inspector de propiedades, introduzca el nombre de instancia **starStepper**.
- 3 En el inspector de propiedades, siga este procedimiento:
 - Introduzca 0 en el parámetro `minimum`.
 - Introduzca 5 en el parámetro `maximum`.
 - Introduzca 0,5 en el parámetro `stepSize`.
 - Introduzca 0 en el parámetro `value`.
- 4 Seleccione el fotograma 1 de la línea de tiempo, abra el panel Acciones e introduzca el código siguiente:

```
movieRate = new Object();
movieRate.change = function (eventObject){
    starChart.value = eventObject.target.value;
}
starStepper.addEventListener("change", movieRate);
```

La última línea de código añade un controlador de eventos `change` a la instancia `starStepper`. El controlador indica al clip de película `starChart` que muestre la cantidad de estrellas que especifica la instancia `starStepper` (para ver cómo funciona este código, debe crear un clip de película `starChart` con una propiedad `value` que muestre las estrellas).

Personalización del componente `NumericStepper`

El componente `NumericStepper` puede transformarse horizontal y verticalmente durante la edición o en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos `Modificar > Transformar`. En tiempo de ejecución, utilice el método `setSize()` (véase [UIObject.setSize\(\)](#)) o cualquier método o propiedad aplicable de la clase `NumericStepper`. Véase [Clase `NumericStepper`](#).

Si se cambia el tamaño del componente `NumericStepper`, no cambia el tamaño de los botones de flecha arriba y flecha abajo. Si al componente `NumericStepper` se le asigna un tamaño superior a la altura predeterminada, los botones se fijan en la parte superior e inferior del componente. Los botones siempre aparecen a la derecha del cuadro de texto.

Utilización de estilos con el componente NumericStepper

Es posible definir propiedades de estilo para cambiar la apariencia de una instancia de NumericStepper. Si el nombre de una propiedad de estilo termina por “Color”, significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 27](#).

El componente NumericStepper admite los siguientes estilos de Halo:

Estilo	Descripción
themeColor	Fondo de un componente. Es el único estilo de color que no hereda su valor. Los valores posibles son "haloGreen", "haloBlue" y "haloOrange".
color	Texto de la etiqueta de un componente.
disabledColor	Color desactivado para el texto.
fontFamily	Nombre de la fuente del texto.
fontSize	Tamaño de la fuente en puntos.
fontStyle	Estilo de la fuente; puede ser “normal” o “italic”.
fontWeight	Grosor de la fuente; puede ser “normal” o “bold”.
textDecoration	Decoración del texto; puede ser “none” o “underline”.
textAlign	Alineación del texto; puede ser “left”, “right” o “center”.

Utilización de aspectos con el componente NumericStepper

El componente NumericStepper emplea aspectos para representar sus estados visuales. Para asignar aspectos a los componentes NumericStepper durante la edición, modifique símbolos de aspecto de la biblioteca y vuelva a exportar el componente como SWC. Los símbolos de aspecto se encuentran en la carpeta Flash UI Components 2/Themes/MMDefault/Stepper Elements/states de la biblioteca. Para más información, consulte [“Aplicación de aspectos a los componentes” en la página 37](#).

Si un componente NumericStepper está activado, los botones de flecha arriba y flecha abajo muestran el estado Sobre cuando el puntero se desplaza sobre ellos. Los botones muestran el estado Presionado cuando se hace clic en ellos y recuperan el estado Sobre cuando se suelta el ratón. Si el puntero se desplaza fuera de los botones estando el ratón presionado, los botones recuperan su estado original.

Si el componente está desactivado, muestra el estado Desactivado sea cual sea la acción del usuario.

El componente NumericStepper utiliza las siguientes propiedades de aspecto:

Propiedad	Descripción
upArrowUp	Estado Arriba de la flecha arriba. El valor predeterminado es StepUpArrowUp.
upArrowDown	Estado Presionado de la flecha arriba. El valor predeterminado es StepUpArrowDown.

Propiedad	Descripción
<code>upArrowOver</code>	Estado Sobre de la flecha arriba. El valor predeterminado es <code>StepUpArrowOver</code> .
<code>upArrowDisabled</code>	Estado Desactivado de la flecha arriba. El valor predeterminado es <code>StepUpArrowDisabled</code> .
<code>downArrowUp</code>	Estado Arriba de la flecha abajo. El valor predeterminado es <code>StepDownArrowUp</code> .
<code>downArrowDown</code>	Estado Presionado de la flecha abajo. El valor predeterminado es <code>StepDownArrowDown</code> .
<code>downArrowOver</code>	Estado Sobre de la flecha abajo. El valor predeterminado es <code>StepDownArrowOver</code> .
<code>downArrowDisabled</code>	Estado Desactivado de la flecha abajo. El valor predeterminado es <code>StepDownArrowDisabled</code> .

Clase NumericStepper

Herencia UIObject > UIComponent > NumericStepper

Nombre de clase de ActionScript mx.controls.NumericStepper

Las propiedades de la clase NumericStepper permiten indicar los valores mínimo y máximo de paso, la cantidad de cada paso en unidades y el valor actual del paso en tiempo de ejecución.

Si una propiedad de la clase NumericStepper se define con ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o el panel Inspector de componentes.

El componente NumericStepper utiliza FocusManager para sustituir el rectángulo de selección predeterminado de Flash Player por un rectángulo de selección personalizado con esquinas redondeadas. Para más información, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 24](#).

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.controls.NumericStepper.version);
```

Nota: el código siguiente devuelve undefined: `trace(myNumericStepperInstance.version);`.

Resumen de métodos de la clase NumericStepper

Hereda todas las propiedades de [Clase UIObject](#) y [Clase UIComponent](#).

Resumen de propiedades de la clase NumericStepper

Propiedad	Descripción
<code>NumericStepper.maximum</code>	Número que indica el valor máximo del rango.
<code>NumericStepper.minimum</code>	Número que indica el valor mínimo del rango.

Propiedad	Descripción
<code>NumericStepper.nextValue</code>	Número que indica el siguiente valor de la secuencia. Es una propiedad de sólo lectura.
<code>NumericStepper.previousValue</code>	Número que indica el valor anterior de la secuencia. Es una propiedad de sólo lectura.
<code>NumericStepper.stepSize</code>	Número que indica la unidad de cambio de cada paso.
<code>NumericStepper.value</code>	Número que indica el valor actual del componente.

Hereda todas las propiedades de [Clase UIObject](#) y [Clase UIComponent](#).

Resumen de eventos de la clase NumericStepper

Evento	Descripción
<code>NumericStepper.change</code>	Se activa cuando cambia el valor del paso.

Hereda todas las propiedades de [Clase UIObject](#) y [Clase UIComponent](#).

NumericStepper.change

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(click){
    ...
}
```

Sintaxis 2:

```
listenerObject = new Object();
listenerObject.change = function(eventObject){
    ...
}
stepperInstance.addEventListener("change", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se cambia el valor del componente.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente `NumericStepper`. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado a la instancia `myStepper`, envía “_level0.myStepper” al panel Salida:

```
on(click){
    trace(this);
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*stepperInstance*) distribuye un evento (en este caso *change*) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto *Event* (*eventObject*) al método del objeto detector. Cada objeto *Event* tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en la página 236.

Ejemplo

Este ejemplo, escrito en un fotograma de la línea de tiempo, envía un mensaje al panel Salida cuando se cambia un componente denominado `myNumericStepper`. La primera línea de código crea un objeto detector denominado `form`. La segunda línea define una función para el evento `change` en el objeto detector. Dentro de la función hay una acción `trace` que utiliza el objeto *Event* pasado automáticamente a la función, en este ejemplo `eventObj`, para generar un mensaje. La propiedad `target` de un objeto *Event* es el componente que ha generado el evento, en este ejemplo `myNumericStepper`. A la propiedad `NumericStepper.value` se accede desde la propiedad `target` del objeto *Event*. La última línea llama al método `UIEventDispatcher.addEventListener()` desde `myNumericStepper` y pasa como parámetros el evento `change` y el objeto detector `form`:

```
form = new Object();
form.change = function(eventObj){
    // eventObj.target es el componente que ha generado el evento change,
    //es decir, NumericStepper.
    trace("Valor cambiado a " + eventObj.target.value);
}
myNumericStepper.addEventListener("change", form);
```

NumericStepper.maximum

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

stepperInstance.maximum

Descripción

Propiedad; valor máximo del rango del componente. Esta propiedad puede contener un número de tres cifras decimales como máximo. El valor predeterminado es 10.

Ejemplo

En el ejemplo siguiente se define el valor máximo del rango del componente en 20:

```
myStepper.maximum = 20;
```

Véase también

[NumericStepper.minimum](#)

NumericStepper.minimum

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
stepperInstance.minimum
```

Descripción

Propiedad; valor mínimo del rango del componente. Esta propiedad puede contener un número de tres cifras decimales como máximo. El valor predeterminado es 0.

Ejemplo

En el ejemplo siguiente se define el valor mínimo del rango del componente en 100:

```
myStepper.minimum = 100;
```

Véase también

[NumericStepper.maximum](#)

NumericStepper.nextValue

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
stepperInstance.nextValue
```

Descripción

Propiedad (sólo lectura); siguiente valor de la secuencia. Esta propiedad puede contener un número de tres cifras decimales como máximo.

Ejemplo

En el ejemplo siguiente se define la propiedad `stepSize` en 1 y el valor inicial en 4, por lo que el valor de `nextValue` sería equivalente a 5:

```
myStepper.stepSize = 1;  
myStepper.value = 4;  
trace(myStepper.nextValue);
```

Véase también

[NumericStepper.previousValue](#)

NumericStepper.previousValue

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

stepperInstance.previousValue

Descripción

Propiedad (sólo lectura); valor anterior de la secuencia. Esta propiedad puede contener un número de tres cifras decimales como máximo.

Ejemplo

En el ejemplo siguiente se define la propiedad `stepSize` en 1 y el valor inicial en 4, por lo que el valor de `nextValue` sería equivalente a 3:

```
myStepper.stepSize = 1;  
myStepper.value = 4;  
trace(myStepper.previousValue);
```

Véase también

[NumericStepper.nextValue](#)

NumericStepper.stepSize

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

stepperInstance.stepSize

Descripción

Propiedad; cantidad en unidades que cambia con respecto al valor actual. El valor predeterminado es 1. Este valor no puede ser 0. Esta propiedad puede contener un número de tres cifras decimales como máximo.

Ejemplo

En el ejemplo siguiente se define el valor actual de `value` en 2 y la unidad `stepSize` en 2. El valor de `nextValue` es 4:

```
myStepper.value = 2;
myStepper.stepSize = 2;
trace(myStepper.nextValue);
```

NumericStepper.value

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

stepperInstance.value

Descripción

Propiedad; valor actual que se muestra en el área de texto del componente. El valor no se muestra si no corresponde al rango y al incremento de paso definidos en la propiedad `stepSize`. Esta propiedad puede contener un número de tres cifras decimales como máximo.

Ejemplo

En el ejemplo siguiente se define el valor actual de `value` en 10 y envía el valor al panel Salida:

```
myStepper.value = 10;
trace(myStepper.value);
```

Clase PopUpManager

Namespace de clase de ActionScript mx.managers.PopUpManager

La clase `PopUpManager` permite crear ventanas superpuestas que pueden ser modales o amodales (una ventana modal no permite interactuar con otras ventanas mientras está activa). Puede llamar a `PopUpManager.createPopUp()` para crear una ventana superpuesta y llamar a `PopUpManager.deletePopUp()` de la instancia de `Window` para eliminar una ventana emergente.

Resumen de métodos de la clase PopUpManager

Evento	Descripción
<code>PopUpManager.createPopUp()</code>	Crea una ventana emergente.
<code>PopUpManager.deletePopUp()</code>	Elimina una ventana emergente creada llamando a <code>PopUpManager.createPopUp()</code> .

PopUpManager.createPopUp()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004 y Flash MX Professional 2004

Sintaxis

```
PopUpManager.createPopUp(parent, class, modal [, initobj, outsideEvents])
```

Parámetros

parent Referencia para que aparezca una ventana.

class Referencia a la clase de objeto que desea crear.

modal Valor booleano que indica si la ventana es modal (`true`) o no (`false`).

initobj Objeto que contiene propiedades de inicialización. Este parámetro es opcional.

outsideEvents Valor booleano que indica si se activa un evento cuando el usuario hace clic fuera de la ventana (`true`) o no (`false`). Este parámetro es opcional.

Valor devuelto

Referencia a la ventana creada.

Descripción

Método; si es modal, una llamada a `createPopUp()` busca la ventana principal situada más arriba a partir de la principal y crea una instancia de clase. Si es amodal, una llamada a `createPopUp()` crea una instancia de clase secundaria con respecto a la ventana principal.

Ejemplo

El código siguiente crea una ventana modal cuando se hace clic en el botón:

```
lo = new Object();
lo.click = function(){
    mx.managers.PopUpManager.createPopUp(_root, mx.containers.Window, true);
}
button.addEventListener("click", lo);
```

PopUpManager.deletePopUp()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004 y Flash MX Professional 2004

Sintaxis

```
windowInstance.deletePopUp();
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; elimina una ventana emergente y quita el estado modal. La ventana superpuesta es la encargada de llamar a `PopUpManager.deletePopUp()` cuando se está eliminando la ventana.

Ejemplo

El código siguiente crea una ventana modal denominada `win` con un botón de cierre, y elimina la ventana cuando se hace clic en el botón de cierre:

```
import mx.managers.PopUpManager
import mx.containers.Window
win = PopUpManager.createPopUp(_root, Window, true, {closeButton:true});
lo = new Object();
lo.click = function(){
    win.deletePopUp();
}
win.addEventListener("click", lo);
```

Componente ProgressBar

El componente `ProgressBar` muestra el progreso de carga mientras el usuario espera a que se cargue el contenido. El proceso de carga puede ser determinado o indeterminado. Una barra de progreso determinado es la representación lineal del progreso de una tarea en el tiempo y se utiliza cuando la cantidad de contenido que va a cargarse es conocida. Una barra de progreso indeterminado se utiliza cuando la cantidad de contenido que va a cargarse es desconocida. Puede añadirse una etiqueta que muestre el progreso del contenido que se está cargando.

Los componentes están definidos de forma predeterminada en Exportar en primer fotograma. Esto significa que los componentes se cargan en la aplicación antes de que se represente el primer fotograma. Si desea crear un preloader para una aplicación, tendrá que anular la selección de Exportar en primer fotograma en el cuadro de diálogo Propiedades de vinculación de cada componente (menú de opciones del panel Biblioteca > Vinculación). El componente `ProgressBar` debe definirse en Exportar en primer fotograma, porque debe visualizarse primero, mientras el resto del contenido se carga en Flash Player.

La vista previa dinámica de cada instancia de `ProgressBar` refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o el panel Inspector de componentes. En la vista previa dinámica se reflejan los siguientes parámetros: `conversion`, `direction`, `label`, `labelPlacement`, `mode` y `source`.

Utilización del componente ProgressBar

Una barra de progreso permite ver el progreso de carga de un contenido. Esta información es fundamental para los usuarios mientras interactúan con una aplicación.

Hay varios modos de utilizar el componente `ProgressBar`; el modo se define con el parámetro `mode`. Los modos utilizados con más frecuencia son “event” (eventos) y “polled” (sondeo). Estos modos utilizan el parámetro `source` para especificar un proceso de carga que emita eventos `progress` y `complete` (modo de eventos) o que revele los métodos `getBytesLoaded` y `getBytesTotal` (modo de sondeo). También puede utilizar el componente `ProgressBar` en modo manual si define manualmente las propiedades `maximum`, `minimum` e `indeterminate` junto con las llamadas al método `ProgressBar.setProgress()`.

Parámetros de ProgressBar

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente `ProgressBar` en el inspector de propiedades o el panel Inspector de componentes:

mode Modo en el que funciona la barra de progreso. Este valor puede ser uno de los siguientes: `event`, `polled` o `manual`. El valor predeterminado es `event`.

source Cadena que va a convertirse en el objeto que represente el nombre de instancia del origen.

direction Sentido en el que se llena la barra de progreso. Este valor puede ser `right` o `left`; el valor predeterminado es `right`.

label Texto que indica el progreso de la carga. Este parámetro es una cadena con el formato “%1 de %2 cargado (%3%)”; %1 es un marcador de posición para los bytes actuales cargados, %2 es un marcador de posición para los bytes totales cargados y %3 es un marcador de posición para el porcentaje del contenido cargado. Los caracteres “%” son un marcador de posición del carácter “%”. Si el valor de %2 es desconocido, se sustituye por “?”. Si el valor es `undefined`, la etiqueta no se muestra.

labelPlacement Posición de la etiqueta con respecto a la barra de progreso. Este parámetro puede tener uno de estos valores: `top`, `bottom`, `left`, `right`, `center`. El valor predeterminado es `bottom`.

conversion Número por el que se dividen los valores %1 y %2 de la cadena de la etiqueta antes de que se muestren. El valor predeterminado es 1.

Puede escribir código `ActionScript` para controlar éstas y otras opciones de componentes `ProgressBar` con sus propiedades, métodos y eventos. Para más información, consulte [Clase ProgressBar](#).

Creación de aplicaciones con el componente ProgressBar

El siguiente procedimiento explica cómo añadir un componente `ProgressBar` a una aplicación durante la edición. En este ejemplo, la barra de progreso se utiliza en modo de eventos. En modo de eventos, el contenido que se carga debe emitir los eventos `progress` y `complete` que la barra de progreso utiliza para mostrar el progreso. El componente `Loader` emite estos eventos. Para más información, consulte [“Componente Loader” en la página 141](#).

Para crear una aplicación con el componente ProgressBar en modo de eventos, siga este procedimiento:

- 1 Arrastre un componente `ProgressBar` desde el panel Componentes al escenario.

- 2 En el inspector de propiedades, siga este procedimiento:
 - Introduzca el nombre de la instancia **pBar**.
 - Seleccione evento para el parámetro **mode**.
- 3 Arrastre un componente Loader desde el panel Componentes al escenario.
- 4 En el inspector de propiedades, introduzca el nombre de la instancia **loader**.
- 5 En el escenario, seleccione la barra de progreso; en el inspector de propiedades, introduzca **loader** para el parámetro **source**.
- 6 Seleccione el fotograma 1 en la línea de tiempo, abra el panel Acciones e introduzca el código siguiente para cargar un archivo JPEG en el componente Loader:


```
loader.autoLoad = false;
loader.content = "http://imagecache2.allposters.com/images/86/017_PP0240.jpg";
pBar.source = loader;
// no cargar hasta haber invocado el método de carga
loader.load();
```

En el ejemplo siguiente se utiliza la barra de progreso en modo de sondeo. En este modo, **ProgressBar** utiliza los métodos `getBytesLoaded` y `getBytesTotal` del objeto de origen para mostrar el progreso de la operación.

Para crear una aplicación con el componente **ProgressBar en modo de sondeo, siga este procedimiento:**

- 1 Arrastre un componente **ProgressBar** desde el panel Componentes al escenario.
- 2 En el inspector de propiedades, siga este procedimiento:
 - Introduzca el nombre de la instancia **pBar**.
 - Seleccione **polled** para el parámetro **mode**.
 - Introduzca **loader** para el parámetro **source**.
- 3 Seleccione el fotograma 1 en la línea de tiempo; abra el panel Acciones e introduzca el código siguiente para crear un objeto **Sound** denominado **loader** e invocar al método `loadSound()` para cargar un sonido en el objeto **Sound**:


```
var loader:Object = new Sound();
loader.loadSound("http://soundamerica.com/sounds/sound_fx/A-E/air.wav",
true);
```

En el ejemplo siguiente se utiliza la barra de progreso en modo manual. Para mostrar el progreso de la operación al utilizar el modo manual, debe definir las propiedades `maximum`, `minimum` e `indeterminate` junto con el método `setProgress()`, pero no necesita definir la propiedad `source`.

Para crear una aplicación con el componente **ProgressBar en modo manual, siga este procedimiento:**

- 1 Arrastre un componente **ProgressBar** desde el panel Componentes al escenario.
- 2 En el inspector de propiedades, siga este procedimiento:
 - Introduzca el nombre de la instancia **pBar**.
 - Seleccione **manual** para el parámetro **mode**.

- 3 Seleccione el fotograma 1 en la línea de tiempo; abra el panel Acciones e introduzca el código siguiente que actualiza la barra de progreso de forma manual cada vez que se descarga un archivo con una llamada al método `setProgress()`:

```
for(var:Number i=1; i <= total; i++){  
    // insertar código para cargar archivo  
    // insertar código para cargar archivo  
    pBar.setProgress(i, total);  
}
```

Personalización del componente ProgressBar

El componente `ProgressBar` puede transformarse horizontalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice `UIObject.setSize()`.

Los extremos izquierdo y derecho de la barra de progreso y el gráfico de la guía de deslizamiento tienen un tamaño fijo. Cuando se cambia el tamaño de una barra de progreso, la parte central se ajusta a la distancia entre los extremos. Si la barra de progreso es demasiado pequeña, es posible que no se pueda representar correctamente.

Utilización de estilos con el componente ProgressBar

Se pueden definir propiedades de estilo para cambiar la apariencia de una instancia de la barra de progreso. Si el nombre de una propiedad de estilo termina por “Color”, significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 27](#).

El componente `ProgressBar` admite los siguientes estilos de Halo:

Estilo	Descripción
<code>themeColor</code>	Fondo de un componente. Es el único estilo de color que no hereda su valor. Los valores posibles son “haloGreen”, “haloBlue” y “haloOrange”.
<code>color</code>	Texto de la etiqueta de un componente.
<code>disabledColor</code>	Color desactivado para el texto.
<code>fontFamily</code>	Nombre de la fuente del texto.
<code>fontSize</code>	Tamaño de la fuente en puntos.
<code>fontStyle</code>	Estilo de la fuente; puede ser “normal” o “italic”.
<code>fontWeight</code>	Grosor de la fuente; puede ser “normal” o “bold”.
<code>textDecoration</code>	Decoración del texto; puede ser “none” o “underline”.

Utilización de aspectos con el componente ProgressBar

El componente ProgressBar indica sus estados mediante los símbolos de clip de película siguientes: TrackMiddle, TrackLeftCap, TrackRightCap y BarMiddle, BarLeftCap, BarRightCap e IndBar. El símbolo IndBar se utiliza para barras de progreso indeterminado. Para aplicar aspectos al componente ProgressBar durante la edición, modifique los símbolos de la biblioteca y vuelva a exportar el componente como archivo SWC. Los símbolos se encuentran en la carpeta Flash UI Components 2/Themes/MMDefault/ProgressBar Elements de la biblioteca del archivo HaloTheme.fla o SampleTheme.fla. Para más información, consulte [“Aplicación de aspectos a los componentes” en la página 37](#).

Si utiliza el método `UIObject.createClassObject()` para crear una instancia del componente ProgressBar de forma dinámica (en tiempo de ejecución), podrá aplicar también el aspecto de forma dinámica. Para asignar aspectos a un componente en tiempo de ejecución, defina las propiedades de aspecto en el parámetro `initObject` que pasa al método `createClassObject()`. Las propiedades de aspecto definen los nombres de los símbolos que se van a utilizar como los estados de la barra de progreso.

Un componente ProgressBar utiliza las propiedades de aspecto siguientes:

Propiedad	Descripción
<code>progTrackMiddleName</code>	Parte central ampliable de la guía de deslizamiento. El valor predeterminado es <code>ProgTrackMiddle</code> .
<code>progTrackLeftName</code>	Extremo izquierdo con tamaño fijo. El valor predeterminado es <code>ProgTrackLeft</code> .
<code>progTrackRightName</code>	Extremo derecho con tamaño fijo. El valor predeterminado es <code>ProgTrackRight</code> .
<code>progBarMiddleName</code>	Gráfico de barra central ampliable. El valor predeterminado es <code>ProgBarMiddle</code> .
<code>progBarLeftName</code>	Extremo izquierdo de la barra con tamaño fijo. El valor predeterminado es <code>ProgBarLeft</code> .
<code>progBarRightName</code>	Extremo derecho de la barra con tamaño fijo. El valor predeterminado es <code>ProgBarRight</code> .
<code>progIndBarName</code>	Gráfico de barra indeterminado. El valor predeterminado es <code>ProgIndBar</code> .

Clase ProgressBar

Herencia UIObject > ProgressBar

Namespace de clase de ActionScript mx.controls.ProgressBar

Si una propiedad de la clase ProgressBar se define con ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o el panel Inspector de componentes.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.controls.ProgressBar.version);
```

Nota: el código siguiente devuelve `undefined`: `trace(myProgressBarInstance.version);`.

Resumen de métodos de la clase ProgressBar

Método	Descripción
ProgressBar.setProgress()	Define el progreso de la barra en modo manual.

Hereda todos los métodos de [Clase UIObject](#).

Resumen de propiedades de la clase ProgressBar

Propiedad	Descripción
ProgressBar.conversion	Número utilizado para convertir el valor de los bytes actuales cargados y el valor total de los bytes cargados.
ProgressBar.direction	Dirección que rellena la barra de progreso.
ProgressBar.indeterminate	Indica que se desconoce el valor total de los bytes de origen.
ProgressBar.label	Texto que acompaña a la barra de progreso.
ProgressBar.labelPlacement	Ubicación de la etiqueta con respecto a la barra de progreso.
ProgressBar.maximum	Valor máximo de la barra de progreso en modo manual.
ProgressBar.minimum	Valor mínimo de la barra de progreso en modo manual.
ProgressBar.mode	Modo en el que la barra de progreso carga el contenido.
ProgressBar.percentComplete	Número que indica el porcentaje cargado.
ProgressBar.source	Contenido para cargar cuyo progreso está controlado por la barra de progreso.
ProgressBar.value	Indica la cantidad del progreso realizado. Es una propiedad de sólo lectura.

Hereda todas las propiedades de [Clase UIObject](#).

Resumen de eventos de la clase ProgressBar

Evento	Descripción
ProgressBar.complete	Se activa cuando se completa la carga.
ProgressBar.progress	Se activa a medida que se carga el contenido en modo de eventos o de sondeo.

Hereda todos los eventos de [Clase UIObject](#).

ProgressBar.complete

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(complete){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.complete = function(eventObject){  
    ...  
}  
pBar.addEventListener("complete", listenerObject)
```

Objeto Event

Además de las propiedades estándar del objeto Event, existen dos propiedades adicionales definidas para el evento `ProgressBar.complete`: `current` (el valor cargado es igual al total) y `total` (el valor total).

Descripción

Evento; se difunde a todos los detectores cuando se ha completado el progreso de carga.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente `ProgressBar`. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado a la instancia `pBar`, envía “_level0.pBar” al panel Salida:

```
on(complete){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (`pBar`) distribuye un evento (en este caso `complete`) y una función asociada al objeto detector creado (`listenerObject`) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto Event (`eventObject`) al método del objeto detector. Cada objeto Event tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en [la página 236](#).

Ejemplo

En este ejemplo se crea un objeto detector `form` con una función callback `complete` que envía un mensaje al panel Salida con el valor de la instancia `pBar`, como en el ejemplo siguiente:

```
form.complete = function(eventObj){  
    // eventObj.target es el componente que ha generado el evento change,  
    // por ejemplo, Progress Bar.  
    trace("Valor cambiado a " + eventObj.target.value);  
}  
pBar.addEventListener("complete", form);
```


Véase también

`UIEventDispatcher.addEventListener()`

ProgressBar.conversion

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

pBarInstance.conversion

Descripción

Propiedad; número que define un valor de conversión para los valores entrantes. Divide los valores actual y total, los promedia y muestra el valor convertido en la propiedad `label`. El valor predeterminado es 1.

Ejemplo

El código siguiente muestra el valor del progreso de carga, expresado en kilobytes:

```
pBar.conversion = 1024;
```

ProgressBar.direction

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

pBarInstance.direction

Descripción

Propiedad; indica la dirección de relleno de la barra de progreso. El valor predeterminado es "right".

Ejemplo

El código siguiente hace que la barra de progreso se rellene de derecha a izquierda:

```
pBar.direction = "left";
```

ProgressBar.indeterminate

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

pBarInstance.indeterminate

Descripción

Propiedad; valor booleano que indica si la barra de progreso tiene un relleno discontinuo y un origen de carga de tamaño desconocido (*true*), o un relleno continuo y un origen de carga con un tamaño conocido (*false*).

Ejemplo

El código siguiente crea una barra de progreso determinada con un relleno continuo de izquierda a derecha:

```
pBar.direction = "right";  
pBar.indeterminate = false;
```

ProgressBar.label

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

pBarInstance.label

Descripción

Propiedad; texto que indica el progreso de carga. La propiedad es una cadena con el formato "%1 de %2 cargado (%3%%)"; %1 es un marcador de posición para los bytes actuales cargados, %2 es un marcador de posición para los bytes totales cargados y %3 es un marcador de posición para el porcentaje del contenido cargado. Los caracteres “%%” son un marcador de posición del carácter “%”. Si el valor de %2 es desconocido, se sustituye por “?”. Si el valor es undefined, la etiqueta no se muestra. El valor predeterminado es "LOADING %3%%"

Ejemplo

El código siguiente define el texto que aparece junto a la barra de progreso con el formato "4 archivos cargados":

```
pBar.label = "%1 archivos cargados";
```

Véase también

[ProgressBar.labelPlacement](#)

ProgressBar.labelPlacement

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

pBarInstance.labelPlacement

Descripción

Propiedad; define la ubicación de la etiqueta con respecto a la barra de progreso. Los valores posibles son "left", "right", "top", "bottom" y "center".

Ejemplo

El código siguiente establece que la etiqueta aparezca por encima de la barra de progreso:

```
pBar.label = "%1 de %2 cargado (%3%)";  
pBar.labelPlacement = "top";
```

Véase también

[ProgressBar.label](#)

ProgressBar.maximum

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

pBarInstance.maximum

Descripción

Propiedad; valor más alto para la barra de progreso cuando la propiedad [ProgressBar.mode](#) está definida en "manual".

Ejemplo

El código siguiente define la propiedad máxima del total de fotogramas de una aplicación de Flash en proceso de carga:

```
pBar.maximum = _totalframes;
```

Véase también

[ProgressBar.minimum](#), [ProgressBar.mode](#)

ProgressBar.minimum

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

pBarInstance.minimum

Descripción

Propiedad; valor más bajo de la barra de progreso cuando la propiedad `ProgressBar.mode` está definida en "manual".

Ejemplo

El código siguiente define el valor mínimo de la barra de progreso:

```
pBar.minimum = 0;
```

Véase también

[ProgressBar.maximum](#), [ProgressBar.mode](#)

ProgressBar.mode

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

pBarInstance.mode

Descripción

Propiedad; modo en el que la barra de progreso carga el contenido. Este valor puede ser uno de los siguientes: "event", "polled" o "manual". Los modos utilizados con más frecuencia son "event" (eventos) y "polled" (sondeo). Estos modos utilizan el parámetro `source` para especificar un proceso de carga que emite eventos `progress` y `complete`, como un componente `Loader` (modo de eventos), o que revela los métodos `getBytesLoaded` y `getBytesTotal`, como un objeto `MovieClip` (modo de sondeo). También puede utilizar el componente `ProgressBar` en modo manual si define manualmente las propiedades `maximum`, `minimum` e `indeterminate` junto con las llamadas al método [ProgressBar.setProgress\(\)](#).

Un objeto `Loader` se debe utilizar como fuente en el modo de eventos. Cualquier objeto que expone los métodos `getBytesLoaded()` y `getBytesTotal()` puede utilizarse como fuente en el modo de sondeo. (Incluido un objeto `Custom` o el objeto `_root`)

Ejemplo

El código siguiente define la barra de progreso en modo de eventos:

```
pBar.mode = "event";
```

ProgressBar.percentComplete

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

pBarInstance.percentComplete

Descripción

Propiedad (sólo lectura); devuelve el porcentaje realizado en el proceso. Este valor se promedia. A continuación se indica la fórmula utilizada para calcular el porcentaje:

$$100 * (\text{valor} - \text{mínimo}) / (\text{máximo} - \text{mínimo})$$

Ejemplo

El código siguiente envía el valor de la propiedad `percentComplete` al panel Salida:

```
trace("porcentaje completo = " + pBar.percentComplete);
```

ProgressBar.progress

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(progress){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.progress = function(eventObject){  
    ...  
}  
pBarInstance.addEventListener("progress", listenerObject)
```

Objeto Event

Además de las propiedades estándar del objeto `Event`, existen dos propiedades adicionales definidas para el evento `ProgressBar.progress`: `current` (el valor cargado es igual al total) y `total` (el valor total).

Descripción

Evento; se difunde a todos los detectores registrados siempre que cambia el valor de una barra de progreso. Este evento sólo se difunde cuando la propiedad `ProgressBar.mode` está definida en `"manual"` o `"polled"`.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente `ProgressBar`. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado a la instancia `myPBar`, envía “_level0.myPBar” al panel Salida:

```
on(progress){
    trace(this);
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*pBarInstance*) distribuye un evento (en este caso `progress`) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (*eventObject*) al método del objeto detector. Cada objeto `Event` tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en [la página 236](#).

Ejemplo

En este ejemplo se crea un objeto detector, `form`, y se define en él un controlador de eventos `progress`. El detector `form` se registra en la última línea del código de la instancia `pBar`. Cuando se activa el evento `progress`, `pBar` difunde el evento al detector `form` que llama a la función `callback progress`, como se indica a continuación:

```
var form:Object = new Object();
form.progress = function(eventObj){
    // eventObj.target es el componente que ha generado el evento change,
    // es decir, Progress Bar.
    trace("Valor cambiado a " + eventObj.target.value);
}
pBar.addEventListener("progress", form);
```

Véase también

[UIEventDispatcher.addEventListener\(\)](#)

ProgressBar.setProgress()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
pBarInstance.setProgress(completed, total)
```

Parámetros

completed número que indica la cantidad del progreso realizado. Se pueden utilizar las propiedades `ProgressBar.label` y `ProgressBar.conversion` para mostrar el número en valor porcentual o cualquier unidad que seleccione, en función de la fuente de la barra de progreso.

total número que indica el progreso que debe realizarse hasta alcanzar el 100%.

Valor devuelto

Número que indica la cantidad del progreso realizado.

Descripción

Método; define el estado de la barra de progreso para que refleje la cantidad de progreso realizado cuando la propiedad `ProgressBar.mode` está definida en "manual". Puede hacer una llamada a este método para que la barra refleje el estado de un proceso que no sea de carga. El argumento completo se asigna a la propiedad `value`; el argumento `total` se asigna a la propiedad máxima. La propiedad mínima no cambia.

Ejemplo

El código siguiente llama al método `setProgress()` en función del progreso realizado por la línea de tiempo de una aplicación de Flash:

```
pBar.setProgress(_currentFrame, _totalFrames);
```

ProgressBar.source

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
pBarInstance.source
```

Descripción

Propiedad; referencia a la instancia que se va a cargar y de la que mostrará el proceso de carga. El contenido que se carga debe emitir un evento `progress` del que se obtienen los valores `total` y `actual`. Este evento sólo se difunde cuando `ProgressBar.mode` está definida en "event" o "polled". El valor predeterminado es `undefined`.

`ProgressBar` se puede utilizar con los contenidos de una aplicación, incluido `_root`.

Ejemplo

En este ejemplo se define la instancia `pBar` para que muestre el progreso de carga de un componente `Loader` con el nombre de instancia `loader`:

```
pBar.source = loader;
```

Véase también

[ProgressBar.mode](#)

ProgressBar.value

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

pBarInstance.value

Descripción

Propiedad (sólo lectura); indica la cantidad del progreso realizado. La propiedad es un número entre el valor de `ProgressBar.minimum` y `ProgressBar.maximum`. El valor predeterminado es 0.

Componente RadioButton

El componente `RadioButton` permite hacer que el usuario seleccione una opción de un conjunto de opciones. El componente `RadioButton` debe utilizarse en un grupo formado al menos por dos instancias de `RadioButton`. Sólo es posible seleccionar un miembro del grupo cada vez. Al seleccionar un botón de opción de un grupo, se anula la selección del botón de opción seleccionado en dicho grupo. Puede definir el parámetro `groupName` para indicar el grupo al que pertenece el botón de opción.

Los botones de opción pueden estar activados o desactivados. Cuando el usuario desplaza el tabulador hasta un grupo de botones de opción, la selección recae sólo en el botón elegido. El usuario puede presionar las teclas de flecha para cambiar la selección dentro del grupo. Un botón de opción en estado Desactivado no recibe la entrada del ratón ni del teclado.

El grupo de componentes `RadioButton` se selecciona cuando el usuario hace clic sobre él o presiona el tabulador hasta su posición. Cuando la selección esté sobre el grupo `RadioButton` podrá controlarla con las teclas siguientes:

Tecla	Descripción
Arriba/Derecha	La selección se desplaza al botón anterior del grupo de botones de opción.
Abajo/Izquierda	La selección se desplaza al botón siguiente del grupo de botones de opción.
Tabulador	Desplaza la selección desde el grupo de botones de opción hasta el componente siguiente.

Para más información sobre el control de la selección, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 24](#) o [“Clase FocusManager” en la página 101](#).

La vista previa dinámica de cada instancia de `RadioButton` del escenario refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o el panel Inspector de componentes. Sin embargo, la vista previa no muestra la exclusión mutua de la selección. Si define el parámetro seleccionado en true para dos botones de opción del mismo grupo, ambos botones aparecerán seleccionados, si bien sólo la última instancia creada estará seleccionada en tiempo de ejecución. Para más información, consulte [“Parámetros de RadioButton” en la página 177](#).

Cuando se añade el componente `RadioButton` a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de la pantalla puedan acceder a él. En primer lugar, debe añadir la línea de código siguiente para activar la accesibilidad:

```
mx.accessibility.RadioButtonAccImpl.enableAccessibility();
```

La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias. Para más información, consulte “Creación de contenido accesible” en el apartado Utilización de Flash de la Ayuda. Es posible que tenga que actualizar el sistema de Ayuda para consultar esta información.

Utilización del componente `RadioButton`

Los botones de opción son una parte fundamental de cualquier formulario o aplicación Web. Puede utilizar los botones de opción siempre que necesite que un usuario seleccione un elemento de un grupo de opciones. Por ejemplo, podría emplear botones de opción en un formulario para preguntar qué tarjeta de crédito va a utilizar el cliente para pagar.

Parámetros de `RadioButton`

Los siguientes son parámetros de edición que se pueden definir para cada instancia del componente `RadioButton` en el inspector de propiedades o el panel Inspector de componentes:

label define el valor del texto de la etiqueta que aparece en el botón; el valor predeterminado es `RadioButton`.

data es el valor de los datos asociados al botón de opción. No hay un valor predeterminado.

groupName es el nombre del grupo del botón de opción. El valor predeterminado es `radioGroup`.

selected define el valor inicial del botón de opción en seleccionado (`true`) o no (`false`). Un botón de opción seleccionado muestra un punto en su interior. Sólo un botón de opción de un grupo puede tener un valor `selected` definido en `true`. Si el grupo contiene más de un botón de opción definido en `true`, sólo se seleccionará el botón cuya instancia se haya hecho en último lugar. El valor predeterminado es `false`.

labelPlacement orienta al texto de la etiqueta que muestra el botón. Este parámetro puede tener uno de estos cuatro valores: `left`, `right`, `top` o `bottom`; el valor predeterminado es `right`. Para más información, consulte [RadioButton.labelPlacement](#).

Puede escribir código `ActionScript` para definir otras opciones para las instancias de `RadioButton` con los métodos, propiedades y eventos de la clase `RadioButton`. Para más información, consulte [Clase RadioButton](#).

Creación de aplicaciones con el componente `RadioButton`

El procedimiento siguiente explica cómo añadir componentes `RadioButton` a una aplicación durante la edición. En este ejemplo, los botones de opción se utilizan para presentar una pregunta “¿Es aficionado a Flash?”, que requiere una respuesta afirmativa o negativa. Los datos del grupo de botones de opción aparecen en un componente `TextArea` con el nombre de instancia `theVerdict`.

Para crear una aplicación con el componente `RadioButton`, siga este procedimiento:

- 1 Arrastre el componente `RadioButton` desde el panel Componentes al escenario.

- 2 Seleccione uno de los botones de opción, vaya al panel Inspector de componentes y siga este procedimiento:
 - Introduzca **Yes** para el parámetro de la etiqueta.
 - Introduzca **Aficionado a Flash** para el parámetro de los datos.
- 3 Seleccione el otro botón de opción, vaya al panel Inspector de componentes y siga este procedimiento:
 - Introduzca **No** para el parámetro de la etiqueta.
 - Introduzca **Anti-Flash** para el parámetro de los datos.
- 4 Seleccione el fotograma 1 de la línea de tiempo, abra el panel Acciones e introduzca el código siguiente:

```
flashistListener = new Object();
flashistListener.click = function (evt){
    theVerdict.text = evt.target.selection.data
}
radioGroup.addEventListener("click", flashistListener);
```

La última línea del código añade un controlador de eventos `click` al grupo de botones de opción `radioGroup`. El controlador define la propiedad `text` de la instancia `theVerdict` del componente `TextArea` con el valor de la propiedad `data` del botón de opción seleccionado en el grupo de botones de opción `radioGroup`. Para más información, consulte [RadioButton.click](#).

Personalización del componente RadioButton

El componente `RadioButton` puede transformarse horizontal o verticalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `setSize()` (véase “[UIObject.setSize\(\)](#)” en la página 249).

El recuadro de delimitación de un componente `RadioButton` es invisible y designa el área activa del componente. Si aumenta el tamaño del componente, aumenta también el tamaño del área activa.

Si el tamaño de la etiqueta supera el del recuadro de delimitación del componente, la etiqueta se recorta para ajustarse al espacio disponible.

Utilización de estilos con el componente RadioButton

Puede definir propiedades de estilo para cambiar la apariencia de una instancia de `RadioButton`. Si el nombre de una propiedad de estilo termina por “Color”, significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte “[Utilización de estilos para personalizar el texto y el color de un componente](#)” en la página 27.

Un componente `RadioButton` utiliza los siguientes estilos de Halo:

Estilo	Descripción
<code>themeColor</code>	Fondo de un componente. Es el único estilo de color que no hereda su valor. Los valores posibles son “haloGreen”, “haloBlue” y “haloOrange”.
<code>color</code>	Texto de la etiqueta de un componente.
<code>disabledColor</code>	Color desactivado para el texto.

Estilo	Descripción
fontFamily	Nombre de la fuente del texto.
fontSize	Tamaño de la fuente en puntos.
fontStyle	Estilo de la fuente; puede ser “normal” o “italic”.
fontWeight	Grosor de la fuente; puede ser “normal” o “bold”.

Utilización de aspectos con el componente RadioButton

Se pueden aplicar aspectos al componente RadioButton durante la edición si se modifican los símbolos del componente en la biblioteca. Los aspectos del componente RadioButton se encuentran en la carpeta siguiente de la biblioteca de HaloTheme.fla o SampleTheme.fla: Flash UI Components 2/Themes/MMDefault/RadioButton Assets/States. Véase [“Aplicación de aspectos a los componentes” en la página 37](#).

Si hay un botón activado y no seleccionado, al mover el puntero sobre él muestra su estado de roll over. Cuando se hace clic sobre un botón de opción no seleccionado, se selecciona y muestra el estado false, correspondiente a Presionado. Si el usuario suelta el ratón, el botón de opción muestra el estado true y el botón de opción seleccionado en el grupo recupera su estado false. Si se mueve el puntero fuera del botón de opción al tiempo que se presiona el ratón, el botón de opción recupera el estado false y retiene la selección de entrada.

Si se desactiva un botón de opción o un grupo de botones de opción, muestra su estado Desactivado sea cual sea la acción del usuario.

Si utiliza el método `UIObject.createClassObject()` para crear dinámicamente una instancia del componente RadioButton, también podrá asignarle aspectos dinámicamente. Para asignar aspectos a un componente RadioButton de forma dinámica, deberá pasar las propiedades del aspecto al método `UIObject.createClassObject()`. Para más información, consulte [“Aplicación de aspectos a los componentes” en la página 37](#). Las propiedades de aspecto indican qué símbolo se debe utilizar para mostrar un componente.

Un componente RadioButton utiliza las propiedades de aspecto siguientes:

Nombre	Descripción
falseUpIcon	Estado Sin marcar. El valor predeterminado es <code>radioButtonFalseUp</code> .
falseDownIcon	Estado Presionado sin marcar. El valor predeterminado es <code>radioButtonFalseDown</code> .
falseOverIcon	Estado Sobre sin marcar. El valor predeterminado es <code>radioButtonFalseOver</code> .
falseDisabledIcon	Estado Desactivado sin marcar. El valor predeterminado es <code>radioButtonFalseDisabled</code> .
trueUpIcon	Estado Marcado. El valor predeterminado es <code>radioButtonTrueUp</code> .
trueDisabledIcon	Estado Desactivado marcado. El valor predeterminado es <code>radioButtonTrueDisabled</code> .

Clase RadioButton

Herencia UIObject > UIComponent > SimpleButton > Button > RadioButton

Nombre de paquete de ActionScript mx.controls.RadioButton

Las propiedades de la clase RadioButton permiten crear en tiempo de ejecución una etiqueta de texto y definir su posición con respecto al botón de opción. También puede asignar valores de datos a los botones de opción, asignarlos a grupos y seleccionarlos en función de los valores de datos o el nombre de instancia.

Si una propiedad de la clase RadioButton se define con ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o el panel Inspector de componentes.

El componente RadioButton utiliza FocusManager para anular el rectángulo de selección predeterminado de Flash Player y dibuja uno personalizado con esquinas redondeadas. Para más información sobre la creación de desplazamientos de la selección, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 24](#).

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.controls.RadioButton.version);
```

Nota: el código siguiente devuelve undefined: `trace(myRadioButtonInstance.version);`.

Resumen de métodos de la clase RadioButton

Hereda todos los métodos de [Clase UIObject](#), [Clase UIComponent](#), [SimpleButton](#) y [Clase Button](#).

Resumen de propiedades de la clase RadioButton

Propiedad	Descripción
RadioButton.data	Valor asociado a una instancia de botón de opción.
RadioButton.groupName	Nombre de grupo de un grupo de botones de opción o una instancia de botón de opción.
RadioButton.label	Texto que aparece junto al botón de opción.
RadioButton.labelPlacement	Orientación del texto de la etiqueta con respecto al botón de opción.
RadioButton.selected	Define el estado de la instancia de botón de opción que se va a seleccionar y anula la selección del elegido anteriormente.
RadioButton.selectedData	Selecciona el botón de opción en un grupo de botones de opción con el valor de datos especificado.
RadioButton.selection	Referencia al botón de opción seleccionado actualmente en un grupo de botones de opción.

Hereda todas las propiedades de [Clase UIObject](#), [Clase UIComponent](#), [SimpleButton](#) y [Clase Button](#).

Resumen de eventos de la clase RadioButton

Evento	Descripción
RadioButton.click	Se activa cuando se presiona el ratón sobre una instancia de botón.

Hereda todos los eventos de [Clase UIObject](#), [Clase UIComponent](#), [SimpleButton](#) y [Clase Button](#)

RadioButton.click

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(click){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.click = function(eventObject){  
    ...  
}  
radioButtonGroup.addEventListener("click", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se hace clic con el ratón (se presiona y se suelta) sobre el botón de opción o al seleccionar el botón con las teclas de flecha. El evento también se difunde si se presiona la barra espaciadora o las teclas de flecha cuando la selección esté sobre un grupo de botones de opción pero no hay ninguno de ellos seleccionado.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente `RadioButton`. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado al botón de opción `myRadioButton`, envía “_level0.myRadioButton” al panel Salida:

```
on(click){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*radioButtonInstance*) distribuye un evento (en este caso `click`) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto Event (*eventObject*) al método del objeto detector. El objeto Event tiene un conjunto de propiedades que contiene información acerca del mismo. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en la página 236.

Ejemplo

Este ejemplo, escrito sobre un fotograma de la línea de tiempo, envía un mensaje al panel Salida cuando se hace clic en un botón de opción de `radioGroup`. La primera línea de código crea un objeto detector denominado `form`. La segunda línea define una función para el evento `click` en el objeto detector. Dentro de la función hay una acción `trace` que utiliza el objeto Event pasado automáticamente a la función, en este ejemplo `eventObj`, para generar un mensaje. La propiedad `target` de un objeto Event es el componente que ha generado el evento. Es posible acceder a las propiedades de la instancia desde la propiedad `target` (en este ejemplo, se accede a la propiedad `RadioButton.selection`). La última línea llama al método `UIEventDispatcher.addEventListener()` desde `radioGroup` y lo pasa el evento `click` y el objeto detector `form` como parámetro, como en el ejemplo siguiente:

```
form = new Object();
form.click = function(eventObj){
    trace("La instancia de opción seleccionada es " +
        eventObj.target.selection);
}
radioGroup.addEventListener("click", form);
```

El código siguiente envía también un mensaje al panel Salida cuando se hace clic en `radioButtonInstance`. El controlador `on()` debe asociarse directamente a `radioButtonInstance`, como en el ejemplo siguiente:

```
on(click){
    trace("clic en componente de botón de opción");
}
```

RadioButton.data

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

radioButtonInstance.data

Descripción

Propiedad; especifica los datos que se asociarán a una instancia de botón de opción. La definición de esta propiedad sustituye el valor del parámetro de datos definido durante la edición en el inspector de propiedades o el panel Inspector de componentes. La propiedad `data` puede ser cualquier tipo de datos.

Ejemplo

En el ejemplo siguiente se asigna el valor de datos `"#FF00FF"` a la instancia de botón de opción `radioOne`:

```
radioOne.data = "#FF00FF";
```

RadioButton.groupName

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
radioButtonInstance.groupName  
radioButtonGroup.groupName
```

Descripción

Propiedad; define el nombre de grupo para una instancia o un grupo de botones de opción. Puede utilizar esta propiedad para obtener o definir un nombre de grupo para una instancia de botón de opción o un nombre de grupo para un grupo de botones de opción. Si se llama a este método, se sustituye el valor del parámetro `groupName` definido durante la edición. El valor predeterminado es `"radioGroup"`.

Ejemplo

En el ejemplo siguiente se define el nombre de grupo de una instancia de botón de opción en `"colorChoice"` y, a continuación, se cambia el nombre de grupo por `"sizeChoice"`. Para probar este ejemplo, coloque en el escenario un botón de opción con el nombre de instancia `myRadioButton` e introduzca el código siguiente en el fotograma 1:

```
myRadioButton.groupName = "colorChoice";  
trace(myRadioButton.groupName);  
colorChoice.groupName = "sizeChoice";  
trace(colorChoice.groupName);
```

RadioButton.label

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

radioButtonInstance.label

Descripción

Propiedad; especifica la etiqueta de texto para el botón de opción. De forma predeterminada, la etiqueta aparece a la derecha del botón de opción. Si se llama a este método, se sustituye el parámetro de etiqueta especificado durante la edición. Si el texto de la etiqueta es demasiado extenso para caber en el recuadro de delimitación del componente, se recorta el texto.

Ejemplo

En el ejemplo siguiente se define la propiedad de etiqueta de la instancia `radioButton`:

```
radioButton.label = "Eliminar de la lista";
```

RadioButton.labelPlacement

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

radioButtonInstance.labelPlacement
radioButtonGroup.labelPlacement

Descripción

Propiedad; cadena que indica la posición de la etiqueta con respecto a un botón de opción. Puede definir esta propiedad para una sola instancia o para un grupo de botones de opción. Si la propiedad se define para un grupo, la etiqueta se coloca en la posición apropiada a cada botón del grupo.

Éstos son los cuatro valores posibles:

- "right" El botón de opción se fija en la esquina superior izquierda del área de delimitación. La etiqueta aparece a la derecha del botón de opción.
- "left" El botón de opción se fija en la esquina superior derecha del área de delimitación. La etiqueta aparece a la izquierda del botón de opción.
- "bottom" La etiqueta aparece por debajo del botón de opción. El conjunto de botón de opción y etiqueta se centra horizontal y verticalmente. Si el recuadro de delimitación del botón de opción no es lo suficientemente grande, la etiqueta se recortará.
- "top" La etiqueta se coloca por encima del botón de opción. El conjunto de botón de opción y etiqueta se centra horizontal y verticalmente. Si el recuadro de delimitación del botón de opción no es lo suficientemente grande, la etiqueta se recortará.

Ejemplo

El código siguiente coloca la etiqueta a la izquierda de cada botón de opción de `radioGroup`:

```
radioGroup.labelPlacement = "left";
```


RadioButton.selected

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
radioButtonInstance.selected  
radioButtonGroup.selected
```

Descripción

Propiedad; valor booleano que define el estado de un botón de opción en seleccionado (*true*) y anula la selección de cualquier otro botón seleccionado anteriormente, o bien define el botón de opción en no seleccionado (*false*).

Ejemplo

La primera línea del código define la instancia de *mcButton* en *true*. La segunda línea del código devuelve el valor de la propiedad seleccionada, como en el ejemplo siguiente:

```
mcButton.selected = true;  
trace(mcButton.selected);
```

RadioButton.selectedData

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
radioButtonGroup.selectedData
```

Descripción

Propiedad; selecciona el botón de opción con el valor de datos especificado y anula la selección del botón seleccionado anteriormente. Si no se ha especificado la propiedad *data* de una instancia seleccionada, se selecciona y devuelve el valor de etiqueta de la instancia seleccionada. La propiedad *selectedData* puede ser cualquier tipo de datos.

Ejemplo

En el ejemplo siguiente se selecciona el botón de opción con el valor "#FF00FF" del grupo de opciones *colorGroup* y envía el valor al panel Salida:

```
colorGroup.selectedData = "#FF00FF";  
trace(colorGroup.selectedData);
```

RadioButton.selection

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
radioButtonInstance.selection  
radioButtonGroup.selection
```

Descripción

Propiedad; se comporta de forma distinta si la propiedad se obtiene o se define. Si la propiedad se obtiene, devuelve la referencia a un objeto del botón seleccionado en un grupo de botones de opción. Si la propiedad se define, selecciona el botón de opción especificado (que se pasa como una referencia a un objeto) de un grupo de botones de opción y anula la selección del botón seleccionado anteriormente.

Ejemplo

En el ejemplo siguiente se selecciona el botón de opción con el nombre de instancia `color1` y envía su nombre de instancia al panel Salida:

```
colorGroup.selection = color1;  
trace(colorGroup.selection._name)
```

Componente RDBMSResolver

Para obtener información reciente sobre esta función, haga clic en el botón Actualizar en la parte superior de la ficha Ayuda.

Interfaz de RemoteProcedureCall

Para obtener información reciente sobre esta función, haga clic en el botón Actualizar en la parte superior de la ficha Ayuda.

Clase Screen

Para obtener información reciente sobre esta función, haga clic en el botón Actualizar en la parte superior de la ficha Ayuda.

Componente ScrollPane

El componente ScrollPane muestra clips de película, archivos JPEG y archivos SWF en un área desplazable. Se pueden activar barras de desplazamiento para que muestren imágenes en un área limitada. También es posible mostrar el contenido que se carga desde una ubicación local o a través de Internet. El uso de ActionScript permite definir el contenido del panel de desplazamiento durante la edición y en tiempo de ejecución.

Cuando el panel de desplazamiento está seleccionado, se seleccionarán los marcadores si el contenido del panel de desplazamiento tiene tabulaciones válidas. Después de la última tabulación del contenido, la selección se desplazará al componente siguiente. La selección no se dirige nunca a las barras de desplazamiento vertical y horizontal del panel de desplazamiento.

La instancia de `ScrollPane` se selecciona cuando el usuario hace clic sobre ella o presiona el tabulador hasta su posición. Cuando la selección esté sobre la instancia de `ScrollPane`, podrá controlarla con las teclas siguientes:

Tecla	Descripción
Flecha abajo	Mueve el contenido una línea de desplazamiento vertical hacia arriba.
Fin	Mueve el contenido hasta la parte inferior del panel de desplazamiento.
Izquierda	Mueve el contenido una línea de desplazamiento horizontal a la derecha.
Inicio	Mueve el contenido hasta la parte superior del panel de desplazamiento.
Av Pág	Mueve el contenido una página vertical hacia arriba.
Re Pág	Mueve el contenido una página vertical hacia abajo.
Derecha	Mueve el contenido una línea de desplazamiento horizontal a la izquierda.
Flecha arriba	Mueve el contenido una línea de desplazamiento vertical hacia abajo.

Para más información sobre el control de la selección, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 24](#) o [“Clase FocusManager” en la página 101](#).

La vista previa dinámica de cada instancia de `ScrollPane` refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o el panel Inspector de componentes.

Utilización del componente `ScrollPane`

Puede utilizar un panel de desplazamiento para mostrar cualquier contenido que sea más grande que el área en la que se ha cargado. Por ejemplo, si tiene una imagen grande que debe mostrarse en un espacio reducido de una aplicación, puede cargarla en un panel de desplazamiento.

Para definir un panel de desplazamiento que permita a los usuarios arrastrar el contenido en el panel, defina el parámetro `scrollDrag` en `true`; en el contenido aparecerá un cursor en forma de mano. A diferencia de lo que ocurre en la mayoría de componentes, los eventos se difunden al presionar el botón del ratón y siguen difundiéndose hasta que se suelta el botón. Si el contenido de un panel de desplazamiento tiene tabulaciones válidas, deberá definir `scrollDrag` en `false` ya que, de lo contrario, cada interacción del ratón con el contenido invocará un arrastre del desplazamiento.

Parámetros de `ScrollPane`

Los siguientes son parámetros de edición que se pueden definir para cada instancia del componente `ScrollPane` en el inspector de propiedades o el panel Inspector de componentes:

contentPath indica el contenido que se va a cargar en el panel de desplazamiento. Este valor puede ser una ruta relativa a un archivo SWF o JPEG local, o bien una ruta relativa o absoluta a un archivo en Internet. También puede ser el identificador de vinculación de un símbolo de clip de película de la biblioteca definido en Exportar para ActionScript.

hLineScrollSize indica el número de unidades que se mueve la barra de desplazamiento horizontal cada vez que se presiona un botón de flecha. El valor predeterminado es 5.

hPageScrollSize indica el número de unidades que se mueve la barra de desplazamiento horizontal cada vez que se presiona la guía de deslizamiento. El valor predeterminado es 20.

hScrollPolicy muestra las barras de desplazamiento horizontal. El valor puede ser "on", "off" o "auto". El valor predeterminado es "auto".

scrollDrag es un valor booleano que permite al usuario desplazar el contenido en el panel de desplazamiento (true) o no (false). El valor predeterminado es false.

vLineScrollSize indica el número de unidades que se mueve la barra de desplazamiento vertical cada vez que se presiona un botón de flecha. El valor predeterminado es 5.

vPageScrollSize indica el número de unidades que se mueve la barra de desplazamiento vertical cada vez que se presiona la guía de deslizamiento. El valor predeterminado es 20.

vScrollPolicy muestra las barras de desplazamiento vertical. El valor puede ser "on", "off" o "auto". El valor predeterminado es "auto".

Puede escribir código ActionScript para controlar éstas y otras opciones adicionales para los componentes de ScrollPane utilizando sus propiedades, métodos y eventos. Para más información, consulte [Clase ScrollPane](#).

Creación de aplicaciones con el componente ScrollPane

En el procedimiento siguiente se explica cómo añadir un componente ScrollPane a una aplicación durante la edición. En este ejemplo, el panel de desplazamiento carga un archivo SWF que contiene un logotipo.

Para crear una aplicación con el componente ScrollPane, siga este procedimiento:

- 1 Arrastre un componente ScrollPane desde el panel Componentes al escenario.
- 2 En el inspector de propiedades, introduzca el nombre de la instancia **myScrollPane**.
- 3 En el inspector de propiedades, introduzca **logo.swf** para el parámetro **contentPath**.
- 4 Seleccione el fotograma 1 de la línea de tiempo, abra el panel Acciones e introduzca el código siguiente:

```
scrollListener = new Object();
scrollListener.scroll = function (evt){
    txtPosition.text = myScrollPane.vPosition;
}
myScrollPane.addEventListener("scroll", scrollListener);

completeListener = new Object;
completeListener.complete = function() {
    trace("completada la carga de logo.swf.");
}
myScrollPane.addEventListener("complete", completeListener);
```

El primer bloque del código es un controlador de eventos **scroll** en la instancia **myScrollPane** que muestra el valor de la propiedad **vPosition** en una instancia de TextField denominada **txtPosition** que se encuentra ya en el escenario. El segundo bloque del código crea un controlador de eventos para el evento **complete** que envía un mensaje al panel Salida.

Personalización del componente ScrollPane

El componente ScrollPane puede transformarse horizontal y verticalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `setSize()` (véase [UIObject.setSize\(\)](#)) o cualquier método o propiedad aplicable de la clase ScrollPane. Véase [Clase ScrollPane](#). Si ScrollPane es demasiado pequeño, es posible que el contenido no se muestre correctamente.

ScrollPane coloca el punto de registro de su contenido en la esquina superior izquierda del panel.

Cuando se desactiva la barra de desplazamiento horizontal, la barra vertical se muestra de arriba abajo sobre el lado derecho del panel de desplazamiento. Si se desactiva la barra de desplazamiento vertical, la barra horizontal se muestra de izquierda a derecha sobre el borde inferior del panel de desplazamiento. También se pueden desactivar las dos barras de desplazamiento.

Cuando se cambia el tamaño del panel de desplazamiento, los botones no se modifican, la guía de deslizamiento y el deslizador se expanden o se contraen y cambia el tamaño de sus áreas activas.

Utilización de estilos con el componente ScrollPane

ScrollPane no admite estilos, pero las barras de desplazamiento que utiliza sí. Para más información, consulte [“Utilización de estilos con el componente ScrollPane” en la página 189](#).

Utilización de aspectos con el componente ScrollPane

A diferencia de las barras de desplazamiento, el componente ScrollPane no tiene aspectos propios. Para más información, consulte [“Utilización de aspectos con el componente ScrollPane” en la página 189](#).

Clase ScrollPane

Herencia UIObject > UICComponent > View > ScrollView > ScrollPane

Namespace de clase de ActionScript mx.containers.ScrollPane

Las propiedades de la clase ScrollPane permiten definir el contenido, controlar el progreso de carga y ajustar la cantidad de desplazamiento en tiempo de ejecución.

Si una propiedad de la clase ScrollPane se define con ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o el panel Inspector de componentes.

Para definir un panel de desplazamiento que permita a los usuarios arrastrar el contenido del panel, establezca la propiedad `scrollDrag` en `true`; en el contenido aparecerá un cursor en forma de mano. A diferencia de lo que ocurre en la mayoría de componentes, los eventos se difunden al presionar el botón del ratón y siguen difundiéndose hasta que se suelta el botón. Si el contenido de un panel de desplazamiento tiene tabulaciones válidas, deberá definir `scrollDrag` en `false` ya que, de lo contrario, cada interacción del ratón con el contenido invocará un arrastre del desplazamiento.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.containers.ScrollPane.version);
```

Nota: el código siguiente devuelve `undefined`: `trace(myScrollPaneInstance.version);`.

Resumen de métodos de la clase ScrollPane

Método	Descripción
<code>ScrollPane.getBytesLoaded()</code>	Devuelve el número de bytes del contenido cargado.
<code>ScrollPane.getBytesTotal()</code>	Devuelve el número total de bytes del contenido que se va a cargar.
<code>ScrollPane.refreshPane()</code>	Vuelve a cargar el contenido del panel de desplazamiento.

Hereda todos los métodos de [Clase UIObject](#) y [Clase UIComponent](#).

Resumen de propiedades de la clase ScrollPane

Método	Descripción
<code>ScrollPane.content</code>	Referencia al contenido cargado en el panel de desplazamiento.
<code>ScrollPane.contentPath</code>	URL absoluta o relativa del archivo SWF o JPEG que se carga en el panel de desplazamiento.
<code>ScrollPane.hLineScrollSize</code>	Cantidad de contenido que se va a desplazar horizontalmente cuando se presione un botón de flecha.
<code>ScrollPane.hPageScrollSize</code>	Cantidad de contenido que se va a desplazar horizontalmente cuando se presione la guía de deslizamiento.
<code>ScrollPane.hPosition</code>	Posición horizontal en píxeles del panel de desplazamiento.
<code>ScrollPane.hScrollPolicy</code>	Estado de la barra de desplazamiento horizontal. Puede estar siempre activada ("on"), siempre desactivada ("off"), o bien activarse cuando sea necesario ("auto"). El valor predeterminado es "auto".
<code>ScrollPane.scrollDrag</code>	Indica si va a haber desplazamiento cuando el usuario presione y arrastre en el panel ScrollPane (true) o no (false). El valor predeterminado es false.
<code>ScrollPane.vLineScrollSize</code>	Cantidad de contenido que se va a desplazar verticalmente cuando se presione un botón de flecha.
<code>ScrollPane.vPageScrollSize</code>	Cantidad de contenido que se va a desplazar verticalmente cuando se presione la guía de deslizamiento.
<code>ScrollPane.vPosition</code>	Posición vertical en píxeles del panel de desplazamiento.
<code>ScrollPane.vScrollPolicy</code>	Estado de la barra de desplazamiento vertical. Puede estar siempre activada ("on"), siempre desactivada ("off"), o bien activarse cuando sea necesario ("auto"). El valor predeterminado es "auto".

Hereda todas las propiedades de [Clase UIObject](#) y [Clase UIComponent](#).

Resumen de eventos de la clase ScrollPane

Método	Descripción
ScrollPane.complete	Se difunde cuando se carga el contenido del panel de desplazamiento.
ScrollPane.progress	Se difunde mientras se carga el contenido de la barra de desplazamiento.
ScrollPane.scroll	Se difunde al presionar la barra de desplazamiento.

Hereda todos los eventos de [Clase UIObject](#) y [Clase UIComponent](#).

ScrollPane.complete

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(complete){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.complete = function(eventObject){  
    ...  
}  
scrollPaneInstance.addEventListener("complete", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando el contenido ha terminado de cargarse.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente `ScrollPane`. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado a la instancia `myScrollPaneComponent` del componente `ScrollPane`, envía “_level0.myScrollPaneComponent” al panel Salida:

```
on(complete){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*ScrollPaneInstance*) distribuye un evento (en este caso *complete*) y una función asociada al objeto detector creado (*ListenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto *Event* (*eventObject*) al método del objeto detector. Cada objeto *Event* tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en [la página 236](#).

Ejemplo

En el ejemplo siguiente se crea un objeto detector con un controlador de eventos *complete* para la instancia *ScrollPane*:

```
form.complete = function(eventObj){  
    // insertar código para controlar el evento  
}  
scrollPane.addEventListener("complete",form);
```

ScrollPane.content

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

ScrollPaneInstance.content

Descripción

Propiedad (sólo lectura); referencia al contenido del panel de desplazamiento. El valor es undefined hasta que comienza la carga.

Ejemplo

En este ejemplo se establece la variable *mcLoaded* en el valor de la propiedad *content*:

```
var mcLoaded = scrollPane.content;
```

Véase también

[ScrollPane.contentPath](#)

ScrollPane.contentPath

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
scrollPaneInstance.contentPath
```

Descripción

Propiedad; cadena que indica una URL absoluta o relativa del archivo SWF o JPEG que se carga en el panel de desplazamiento. La ruta relativa debe hacer referencia al archivo SWF que carga el contenido.

Si carga el contenido utilizando una URL relativa, el contenido cargado deberá ser relativo a la ubicación del archivo SWF que contiene el panel de desplazamiento. Por ejemplo, una aplicación que utilice un componente ScrollPane situado en el directorio /scrollpane/nav/example.swf podría cargar el contenido del directorio /scrollpane/content/flash/logo.swf con la siguiente propiedad contentPath: ". ./content/flash/logo.swf"

Ejemplo

En el ejemplo siguiente se indica al panel de desplazamiento que muestre el contenido de una imagen procedente de Internet:

```
scrollPane.contentPath ="http://imagecache2.allposters.com/images/43/  
033_302.jpg";
```

En el ejemplo siguiente se indica al panel de desplazamiento que muestre el contenido de un símbolo de la biblioteca:

```
scrollPane.contentPath ="movieClip_Name";
```

En el ejemplo siguiente se indica al panel de desplazamiento que muestre el contenido del archivo local "logo.swf":

```
scrollPane.contentPath ="logo.swf";
```

Véase también

[ScrollPane.content](#)

ScrollPane.getBytesLoaded()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
scrollPaneInstance.getBytesLoaded()
```

Parámetros

Ninguno.

Valor devuelto

Número de bytes cargados en el panel de desplazamiento.

Descripción

Método; devuelve el número de bytes cargados en la instancia ScrollPane. Mientras se carga el contenido, invoque este método en intervalos regulares para comprobar el progreso de la operación.

Ejemplo

En este ejemplo se crea una instancia de la clase ScrollPane denominada `scrollPane`. A continuación, se define un objeto detector denominado `loadListener` con un controlador de eventos `progress` que llama al método `getBytesLoaded()` para que le ayude a determinar el progreso de carga:

```
createClassObject(mx.containers.ScrollPane, "scrollPane", 0);
loadListener = new Object();
loadListener.progress = function(eventObj){
    // eventObj.target es el componente que genera el evento change
    var bytesLoaded = scrollPane.getBytesLoaded();
    var bytesTotal = scrollPane.getBytesTotal();
    var percentComplete = Math.floor(bytesLoaded/bytesTotal);

    if (percentComplete < 5 ) // la carga acaba de comenzar
    {
        trace(" Comenzada la carga de contenidos de Internet");
    }
    else if(percentComplete = 50) //50% completado
    {
        trace(" 50% del contenido descargado ");
    }
}
scrollPane.addEventListener("progress", loadListener);
scrollPane.contentPath = "http://www.geocities.com/hcls_matrix/Images/homeview5.jpg";
```

ScrollPane.getBytesTotal()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
scrollPaneInstance.getBytesTotal()
```

Parámetros

Ninguno.

Valor devuelto

Un número.

Descripción

Método; devuelve el número total de bytes que se van a cargar en la instancia ScrollPane.

Véase también

[ScrollPane.getBytesLoaded\(\)](#)

ScrollPane.hLineScrollSize

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

scrollPaneInstance.hLineScrollSize

Descripción

Propiedad; número de píxeles que se va a mover el contenido al presionar la flecha izquierda o derecha en la barra de desplazamiento horizontal. El valor predeterminado es 5.

Ejemplo

En este ejemplo se incrementa en 10 unidades el desplazamiento horizontal:

```
scrollPane.hLineScrollSize = 10;
```

ScrollPane.hPageScrollSize

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

scrollPaneInstance.hPageScrollSize

Descripción

Propiedad; número de píxeles que se va a mover el contenido al presionar la guía en la barra de desplazamiento horizontal. El valor predeterminado es 20.

Ejemplo

En este ejemplo se incrementa en 30 unidades el desplazamiento horizontal de la página:

```
scrollPane.hPageScrollSize = 30;
```

ScrollPane.hPosition

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

scrollPaneInstance.hPosition

Descripción

Propiedad; posición en píxeles de la barra de desplazamiento horizontal. La posición 0 se encuentra a la izquierda de la barra.

Ejemplo

En este ejemplo se establece la barra de desplazamiento en 20:

```
scrollPane.hPosition = 20;
```

ScrollPane.hScrollPolicy

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

scrollPaneInstance.hScrollPolicy

Descripción

Propiedad; determina si la barra de desplazamiento horizontal está presente siempre ("on"), nunca ("off") o aparece en función del tamaño de la imagen ("auto"). El valor predeterminado es "auto".

Ejemplo

El código siguiente mantiene siempre activas las barras desplazamiento:

```
scrollPane.hScrollPolicy = "on";
```

ScrollPane.progress

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(progress){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();
listenerObject.progress = function(eventObject){
    ...
}
scrollPaneInstance.addEventListener("progress", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados mientras se carga el contenido. El evento `progress` no siempre se difunde; el evento `complete` puede difundirse sin que se distribuyan eventos `progress`. Esto sucede especialmente cuando el contenido cargado es un archivo local. Este evento se activa cuando el contenido empieza a cargarse al establecer el valor de la propiedad `contentPath`.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente `ScrollPane`. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado a la instancia `mySPComponent` del componente `ScrollPane`, envía “_level0.mySPComponent” al panel Salida:

```
on(progress){
    trace(this);
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (`scrollPaneInstance`) distribuye un evento (en este caso `progress`) y una función asociada al objeto detector creado (`listenerObject`) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (`eventObject`) al método del objeto detector. Cada objeto `Event` tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 236](#).

Ejemplo

El código siguiente crea una instancia de `ScrollPane`, llamada `scrollPane` y un objeto detector con un controlador de eventos para el evento `progress` que envía un mensaje al panel Salida con el número de bytes del contenido que se ha cargado:

```
createClassObject(mx.containers.ScrollPane, "scrollPane", 0);
loadListener = new Object();
loadListener.progress = function(eventObj){
    // eventObj.target es el componente que genera el evento progress
    // en este caso, scrollPane
    trace("logo.swf cargado " + scrollPane.getBytesLoaded() + " Bytes.");
    // llevar seguimiento de progreso de carga
}
scrollPane.addEventListener("complete", loadListener);
scrollPane.contentPath = "logo.swf";
```

ScrollPane.refreshPane()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
ScrollPaneInstance.refreshPane()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; actualiza el panel de desplazamiento después de cargar el contenido. Este método vuelve a cargar el contenido. Utilice este método si, por ejemplo, ha cargado un formulario en ScrollPane y se ha modificado una propiedad de entrada (por ejemplo, en un campo de texto) con ActionScript. En tal caso, llame a `refreshPane()` para volver a cargar el mismo formulario con los nuevos valores para las propiedades de entrada.

Ejemplo

En el ejemplo siguiente se actualiza la instancia `sp` del panel de desplazamiento:

```
sp.refreshPane();
```

ScrollPane.scroll

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(scroll){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.scroll = function(eventObject){  
    ...  
}  
ScrollPaneInstance.addEventListener("scroll", listenerObject)
```

Objeto Event

Además de las propiedades estándar del objeto Event, hay una propiedad `type` definida para el evento `scroll`; el valor es `"scroll"`. También hay una propiedad `direction` con los valores `"vertical"` y `"horizontal"` posibles.

Descripción

Evento; se difunde a todos los detectores registrados cuando un usuario presiona los botones de la barra de desplazamiento, el deslizador o la guía de deslizamiento. A diferencia de otros eventos, `scroll` se difunde cuando el usuario presiona la barra de desplazamiento y continúa difundiéndose hasta que la suelta.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente `ScrollPane`. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado a la instancia `sp`, envía `"_level0.sp"` al panel Salida:

```
on(scroll){
  trace(this);
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*ScrollPaneInstance*) distribuye un evento (en este caso `scroll`) y una función asociada al objeto detector creado (*ListenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto Event (*EventObject*) al método del objeto detector. Cada objeto Event tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 236](#).

Ejemplo

En este ejemplo se crea un objeto detector `form` con una función callback `scroll` que se registra en la instancia `spInstance`. Debe rellenar `spInstance` de contenido, como en el ejemplo siguiente:

```
spInstance.contentPath = "mouse3.jpg";
form = new Object();
form.scroll = function(eventObj){
  trace("ScrollPane recorrido");
}
spInstance.addEventListener("scroll", form);
```

Véase también

[UIEventDispatcher.addEventListener\(\)](#)

ScrollPane.scrollDrag

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
ScrollPaneInstance.scrollDrag
```

Descripción

Propiedad; valor booleano que indica si se va a producir un desplazamiento cuando el usuario presione y arrastre en ScrollPane (`true`) o no (`false`). El valor predeterminado es `false`.

Ejemplo

En este ejemplo se activa el desplazamiento con ratón en el interior del panel de desplazamiento:

```
ScrollPane.scrollDrag = true;
```

ScrollPane.vLineScrollSize

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
ScrollPaneInstance.vLineScrollSize
```

Descripción

Propiedad; número de píxeles que se moverá el área de visualización al presionar el botón de flecha arriba o abajo en una barra de desplazamiento vertical. El valor predeterminado es 5.

Ejemplo

Este código incrementa en 10 la cantidad del área de visualización que se mueve al presionar los botones de flecha de la barra de desplazamiento:

```
ScrollPane.vLineScrollSize = 10;
```

ScrollPane.vPageScrollSize

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

```
ScrollPaneInstance.vPageScrollSize
```


Descripción

Propiedad; número de píxeles que se moverá el área de visualización al presionar la guía en una barra de desplazamiento vertical. El valor predeterminado es 20.

Ejemplo

Este código incrementa en 30 la cantidad del área de visualización que se mueve al presionar los botones de flecha de la barra de desplazamiento vertical:

```
scrollPane.vPageScrollSize = 30;
```

ScrollPane.vPosition

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

scrollPaneInstance.vPosition

Descripción

Propiedad; posición en píxeles de la barra de desplazamiento vertical. El valor predeterminado es 0.

ScrollPane.vScrollPolicy

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

scrollPaneInstance.vScrollPolicy

Descripción

Propiedad; determina si la barra de desplazamiento vertical está presente siempre ("on"), nunca ("off") o si aparece en función del tamaño de la imagen ("auto"). El valor predeterminado es "auto".

Ejemplo

El código siguiente mantiene siempre activas las barras de desplazamiento vertical:

```
scrollPane.vScrollPolicy = "on";
```

Clase StyleManager

Namespace de clase de ActionScript mx.styles.StyleManager

La clase StyleManager lleva el seguimiento de los colores y estilos que se heredan. Sólo necesitará utilizar esta clase si ha creado componentes y desea añadir un color o un estilo heredados.

Para determinar qué estilos se heredan, consulte el [sitio Web W3C](#).

Resumen de métodos de la clase StyleManager

Método	Descripción
StyleManager.registerColorName()	Registra un nombre de color nuevo con StyleManager.
StyleManager.registerColorStyle()	Registra un estilo de color nuevo con StyleManager.
StyleManager.registerInheritingStyle()	Registra un estilo heredado nuevo con StyleManager.

StyleManager.registerColorName()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
StyleManager.registerColorName(colorName, value)
```

Parámetros

colorName Cadena que indica el nombre del color (por ejemplo, "gray", "darkGrey", y así sucesivamente).

value Número hexadecimal que indica el color (por ejemplo, 0x808080, 0x404040, y así sucesivamente).

Valor devuelto

Ninguno.

Descripción

Método; asocia un nombre de color con un valor hexadecimal y lo registra con StyleManager.

Ejemplo

En el ejemplo siguiente se registra "gray" como el nombre de color representado por el valor hexadecimal 0x808080:

```
StyleManager.registerColorName("gray", 0x808080 );
```

StyleManager.registerColorStyle()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
StyleManager.registerColorStyle(colorStyle)
```

Parámetros

colorStyle Cadena que indica el nombre del color (por ejemplo, "highlightColor", "shadowColor", "disabledColor", y así sucesivamente).

Valor devuelto

Ninguno.

Descripción

Método; añade un estilo de color nuevo a StyleManager.

Ejemplo

En el ejemplo siguiente se registra "highlightColor" como un estilo de color:

```
StyleManager.registerColorStyle("highlightColor");
```

StyleManager.registerInheritingStyle()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
StyleManager.registerInheritingStyle(propertyName)
```

Parámetros

propertyName Cadena que indica el nombre de la propiedad de estilo (por ejemplo, "newProp1", "newProp2", y así sucesivamente).

Valor devuelto

Ninguno.

Descripción

Método; marca la propiedad de estilo como heredada. Utilice este método para registrar las propiedades de estilo que no aparecen en la lista de la especificación CSS. No utilice este método para cambiar propiedades de estilo no heredadas a heredadas.

Ejemplo

En el ejemplo siguiente se registra `newProp1` como un estilo heredado:

```
StyleManager.registerInheritingStyle("newProp1");
```

Clase Slide

Para obtener información reciente sobre esta función, haga clic en el botón Actualizar en la parte superior de la ficha Ayuda.

Componente TextArea

El componente `TextArea` ajusta el objeto `TextField` nativo de `ActionScript`. Puede utilizar estilos para personalizar el componente `TextArea`; cuando se desactiva una instancia, su contenido se muestra en un color representado por el estilo `disabledColor`. El componente `TextArea` también se puede formatear con HTML o como un campo de contraseña que disfraza el texto.

Es posible activar o desactivar el componente `TextArea` en una aplicación. Si está desactivado, no recibe la entrada del ratón ni del teclado. Cuando está activado, sigue las mismas reglas de selección y navegación que un objeto `TextField` de `ActionScript`. Cuando la instancia de `TextArea` esté seleccionada, podrá controlarla con las teclas siguientes:

Tecla	Descripción
Teclas de flecha	Desplaza el punto de inserción una línea hacia arriba, abajo, izquierda o derecha.
Av Pág	Avanza el punto de inserción una página.
Re Pág	Retrocede el punto de inserción una página.
Mayúsculas + Tabulador	Desplaza la selección al objeto anterior.
Tabulador	Desplaza la selección al objeto siguiente.

Para más información sobre el control de la selección, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 24](#) o [“Clase FocusManager” en la página 101](#).

La vista previa dinámica de cada instancia de `TextArea` refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o el panel Inspector de componentes. Si se necesita una barra de desplazamiento, ésta aparece en la vista previa dinámica, aunque no funciona. No es posible seleccionar texto en la vista previa, de igual modo que tampoco se puede introducir texto en la instancia del componente en el escenario.

Cuando se añade el componente `TextArea` a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de pantalla puedan acceder al mismo.

Puede utilizar un componente `TextArea` siempre que necesite un campo de texto de varias líneas. Si necesita un campo de texto con una sola línea, utilice [“Componente TextInput” en la página 216](#). Por ejemplo, suponga que precisa un componente `TextArea` como campo de comentarios de un formulario. En tal caso, puede definir un detector que compruebe si el campo está vacío cuando un usuario emplea el tabulador fuera del campo. El detector podría mostrar un mensaje de error indicando que debe introducirse un comentario en el campo.

Parámetros del componente TextArea

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente TextArea en el inspector de propiedades o el panel Inspector de componentes:

text indica el contenido de TextArea. No es posible introducir retornos de carro en el inspector de propiedades ni en el panel Inspector de componentes. El valor predeterminado es "" (cadena vacía).

html indica si el texto está formateado con HTML (true) o no (false). El valor predeterminado es false.

editable indica si el componente TextArea es editable (true) o no (false). El valor predeterminado es true.

wordWrap indica si el texto se ajusta (true) o no (false). El valor predeterminado es true.

Puede escribir código ActionScript para controlar éstas y otras opciones adicionales para los componentes de TextArea utilizando sus propiedades, métodos y eventos. Para más información, consulte [Clase TextArea](#).

Creación de aplicaciones con el componente TextArea

El procedimiento siguiente explica cómo añadir un componente TextArea a una aplicación durante la edición. En este ejemplo, el componente es un campo Comment con un detector de eventos que determina si un usuario ha introducido texto.

Para crear una aplicación con el componente TextArea, siga este procedimiento:

- 1 Arrastre un componente TextArea desde el panel Componentes al escenario.
- 2 En el inspector de propiedades, introduzca el nombre de la instancia **comment**.
- 3 En el inspector de propiedades, defina los parámetros deseados. No obstante, deberá dejar el parámetro text vacío, el parámetro editable establecido en true y el parámetro password en false.
- 4 Seleccione el fotograma 1 de la línea de tiempo, abra el panel Acciones e introduzca el código siguiente:

```
textListener = new Object();
textListener.handleEvent = function (evt){
    if (comment.length < 1) {
        Alert(_root, "Error", "Debe introducir al menos un comentario en este
        campo", mxModal | mxOK);
    }
}
comment.addEventListener("focusOut", textListener);
```

Este código establece un controlador de eventos focusOut en la instancia comment de TextArea para comprobar que el usuario ha introducido algo en el campo de texto.

- 5 Una vez introducido el texto en la instancia, puede obtener su valor como se muestra a continuación:

```
var login = comment.text;
```

Personalización del componente TextArea

El componente TextArea puede transformarse horizontal o verticalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice `UIObject.setSize()` o cualquiera de las propiedades y métodos aplicables de [Clase TextArea](#).

Cuando se cambia el tamaño de un componente TextArea, el borde se ajusta a los límites del cuadro nuevo. Si es necesario, las barras de desplazamiento se sitúan en los bordes inferior y derecho. A continuación, se ajusta el tamaño del campo de texto al interior del área restante; el componente TextArea carece de elementos con tamaño fijo. Si el componente TextArea es demasiado pequeño para mostrar el texto, éste se recorta.

Utilización de estilos con el componente TextArea

Si bien el componente TextArea admite un conjunto de estilos de componente para todo el texto del campo, puede mostrar también HTML compatible con la representación HTML de Flash Player. Para mostrar texto HTML, defina `TextArea.html` en `true`.

El componente TextArea tiene sus propiedades de estilo `backgroundColor` y `borderStyle` definidas en una declaración de estilo de clase. Los estilos de clase anulan los estilos `_global`; por tanto, si desea definir las propiedades de estilo `backgroundColor` y `borderStyle`, debe crear una declaración distinta de estilo predeterminado en la instancia.

Si el nombre de una propiedad de estilo termina por “Color”, significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 27](#).

Un componente TextArea admite los siguientes estilos:

Estilo	Descripción
<code>color</code>	Color predeterminado del texto.
<code>embedFonts</code>	Fuentes que incorporar al documento.
<code>fontFamily</code>	Nombre de la fuente del texto.
<code>fontSize</code>	Tamaño de la fuente en puntos.
<code>fontStyle</code>	Estilo de la fuente; puede ser “normal” o “italic”.
<code>fontWeight</code>	Grosor de la fuente; puede ser “normal” o “bold”.
<code>textAlign</code>	Alineación del texto; puede ser “left”, “right” o “center”.
<code>textDecoration</code>	Decoración del texto; puede ser “none” o “underline”.

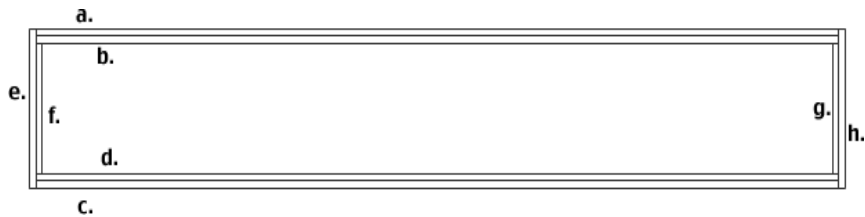
Utilización de aspectos con el componente TextArea

El componente TextArea utiliza la clase RectBorder para dibujar su borde. Puede utilizar el método `setStyle()` (véase [UIObject.setStyle\(\)](#)) para cambiar las propiedades de estilo RectBorder siguientes:

estilos RectBorder

`borderColor`
`highlightColor`
`borderColor`
`shadowColor`
`borderCapColor`
`shadowCapColor`
`shadowCapColor`
`borderCapColor`

Las propiedades de estilo definen las siguientes posiciones en el borde:



Clase TextArea

Herencia UIObject > UIComponent > View > ScrollView > TextArea

Namespace de clase de ActionScript mx.controls.TextArea

Las propiedades de la clase TextArea permiten definir el contenido y el formato del texto, así como las posiciones vertical y horizontal en tiempo de ejecución. También puede indicar si el campo es editable y si se trata de un campo “contraseña”, o restringir los caracteres que el usuario puede introducir.

Si una propiedad de la clase TextArea se define con ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o el panel Inspector de componentes.

El componente TextArea anula el rectángulo de selección predeterminado de Flash Player y dibuja uno personalizado con esquinas redondeadas.

El componente TextArea admite estilos CSS y cualquier otro estilo adicional de HTML admitido por Flash Player.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.controls.TextArea.version);
```

Nota: el código siguiente devuelve undefined: `trace(myTextAreaInstance.version);`.

Resumen de propiedades de la clase `TextArea`

Propiedad	Descripción
<code>TextArea.editable</code>	Valor booleano que indica si el campo es editable (<code>true</code>) o no (<code>false</code>).
<code>TextArea.hPosition</code>	Define la posición horizontal del texto incluido en el panel de desplazamiento.
<code>TextArea.hScrollPolicy</code>	Indica si la barra de desplazamiento horizontal está activada siempre (<code>"on"</code>), nunca (<code>"off"</code>) o se activa cuando es necesario (<code>"auto"</code>).
<code>TextArea.html</code>	Etiqueta que indica si el campo de texto admite formato HTML.
<code>TextArea.length</code>	Número de caracteres de un campo de texto. Es una propiedad de sólo lectura.
<code>TextArea.maxChars</code>	Número máximo de caracteres que puede contener el campo de texto.
<code>TextArea.maxHPosition</code>	Valor máximo de <code>TextArea.hPosition</code> .
<code>TextArea.maxVPosition</code>	Valor máximo de <code>TextArea.vPosition</code> .
<code>TextArea.password</code>	Valor booleano que indica si el campo es un campo de contraseña (<code>true</code>) o no (<code>false</code>).
<code>TextArea.restrict</code>	Conjunto de caracteres que puede introducir un usuario en el campo de texto.
<code>TextArea.text</code>	Contenido de texto de un componente <code>TextArea</code> .
<code>TextArea.vPosition</code>	Número que indica la posición de desplazamiento vertical.
<code>TextArea.vScrollPolicy</code>	Indica si la barra de desplazamiento vertical está activada siempre (<code>"on"</code>), nunca (<code>"off"</code>) o se activa cuando es necesario (<code>"auto"</code>).
<code>TextArea.wordWrap</code>	Valor booleano que indica si el texto se ajusta (<code>true</code>) o no (<code>false</code>).

Resumen de eventos de la clase `TextArea`

Evento	Descripción
<code>TextArea.change</code>	Notifica a los detectores que el texto ha cambiado.

`TextArea.change`

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(change){  
    ...  
}
```


Sintaxis 2:

```
listenerObject = new Object();
listenerObject.change = function(eventObject){
    ...
}
textAreaInstance.addEventListener("change", listenerObject)
```

Descripción

Evento; notifica a los detectores que el texto ha cambiado. Este evento se difunde una vez modificado el texto. No se puede utilizar para impedir que ciertos caracteres se añadan al campo de texto del componente; para ello, deberá utilizar [TextArea.restrict](#).

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente `TextArea`. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado a la instancia `myTextArea`, envía “_level0.myTextArea” al panel Salida.

```
on(change){
    trace(this);
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*textAreaInstance*) distribuye un evento (en este caso `change`) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (*eventObject*) al método del objeto detector. Cada objeto `Event` tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 236](#).

Ejemplo

En este ejemplo se señala el número total de veces que ha cambiado el campo de texto:

```
myTextArea.changeHandler = function(obj) {
    this.changeCount++;
    trace(obj.target);
    trace("el texto ha cambiado" + this.changeCount + " veces! ahora contiene "
    +
    this.text);
}
```

Véase también

[UIEventDispatcher.addEventListener\(\)](#)

TextArea.editable

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

textAreaInstance.editable

Descripción

Propiedad; valor booleano que indica si el componente es editable (*true*) o no (*false*). El valor predeterminado es *true*.

TextArea.hPosition

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

textAreaInstance.hPosition

Descripción

Propiedad; define la posición horizontal del texto dentro del campo. El valor predeterminado es 0.

Ejemplo

El código siguiente muestra los caracteres situados en el extremo izquierdo del campo:

```
myTextArea.hPosition = 0;
```

TextArea.hScrollPolicy

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

textAreaInstance.hScrollPolicy

Descripción

Propiedad; determina si la barra de desplazamiento horizontal está presente siempre (*"on"*), nunca (*"off"*) o aparece automáticamente en función del tamaño del campo (*"auto"*). El valor predeterminado es *"auto"*.

Ejemplo

El código siguiente mantiene siempre activas las barras de desplazamiento horizontal:

```
text.hScrollPolicy = "on";
```

TextArea.html

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

textAreaInstance.html

Descripción

Propiedad; valor booleano que indica si el campo de texto tiene formato HTML (*true*) o no (*false*). Si el valor de la propiedad *html* es *true*, el campo de texto es un campo de texto HTML. Si el valor de *html* es *false*, el campo de texto no es un campo de texto HTML. El valor predeterminado es *false*.

Ejemplo

En el ejemplo siguiente se convierte el campo *myTextArea* en un campo de texto HTML y luego se formatea el texto con etiquetas HTML:

```
myTextArea.html = true;  
myTextArea.text = "The <b>Royal</b> Nonesuch"; // mostrar "The Royal Nonesuch"
```

TextArea.length

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

textAreaInstance.length

Descripción

Propiedad (sólo lectura); indica el número de caracteres de un campo de texto. Esta propiedad devuelve el mismo valor que *text.length* de *ActionScript*, pero es más rápida. El carácter de tabulador ("*\t*") cuenta como un carácter. El valor predeterminado es 0.

Ejemplo

En el ejemplo siguiente se obtiene la longitud del campo de texto y se copia en la variable *length*:

```
var length = myTextArea.length; // hallar la longitud de la cadena de texto
```

TextArea.maxChars

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

textAreaInstance.maxChars

Descripción

Propiedad; indica el número máximo de caracteres que puede contener el campo de texto. Un script puede insertar más texto del que permite `maxChars`; la propiedad `maxChars` sólo indica cuánto texto puede introducir el usuario. Si el valor de esta propiedad es `null`, no hay límite en cuanto a la cantidad de texto que puede introducirse. El valor predeterminado es `null`.

Ejemplo

En el ejemplo siguiente, se limita a 255 el número de caracteres que puede introducir el usuario:

```
myTextArea.maxChars = 255;
```

TextArea.maxHPosition

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

textAreaInstance.maxHPosition

Descripción

Propiedad (sólo lectura); valor máximo de `TextArea.hPosition`. El valor predeterminado es 0.

Ejemplo

El código siguiente desplaza el texto hasta el extremo derecho:

```
myTextArea.hPosition = myTextArea.maxHPosition;
```

Véase también

`TextArea.vPosition`

TextArea.maxVPosition

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

textAreaInstance.maxVPosition

Descripción

Propiedad (sólo lectura); indica el valor máximo de `TextArea.vPosition`. El valor predeterminado es 0.

Ejemplo

El código siguiente desplaza el texto hasta la parte inferior del componente:

```
myTextArea.vPosition = myTextArea.maxVPosition;
```

Véase también

[TextArea.hPosition](#)

TextArea.password

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
textAreaInstance.password
```

Descripción

Propiedad; valor booleano que indica si el campo de texto es un campo de contraseña (`true`) o no (`false`). Si el valor de `password` es `true`, el campo de texto es un campo de contraseña y oculta los caracteres de entrada. Si su valor es `false`, el campo de texto no es de contraseña. El valor predeterminado es `false`.

Ejemplo

El código siguiente convierte el campo de texto en un campo de contraseña y muestra todos los caracteres como asteriscos (*):

```
myTextArea.password = true;
```

TextArea.restrict

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
textAreaInstance.restrict
```

Descripción

Propiedad; indica el conjunto de caracteres que los usuarios pueden introducir en el campo de texto. El valor predeterminado es `undefined`. Si el valor de la propiedad `restrict` es `null`, el usuario puede introducir cualquier carácter. Si el valor de la propiedad `restrict` es una cadena vacía, no se puede introducir ningún carácter. Si el valor de la propiedad `restrict` es una cadena de caracteres, sólo podrán introducirse en el campo de texto; el texto se explora de izquierda a derecha. Puede especificarse un intervalo utilizando un guión (-).

La propiedad `restrict` sólo limita la interacción del usuario; un script puede introducir cualquier texto en el campo de texto. Esta propiedad no se sincroniza con las casillas de verificación de Incorporar contornos de fuentes del inspector de propiedades.

Si la cadena empieza por "^", inicialmente se aceptan todos los caracteres; los caracteres posteriores de la cadena se excluyen del conjunto de caracteres aceptados. Si la cadena no empieza por "^", inicialmente no se acepta ningún carácter; los caracteres posteriores de la cadena se incluyen en el conjunto de caracteres aceptados.

Ejemplo

En el ejemplo siguiente, la primera línea del código limita el campo de texto a letras en mayúsculas, números y espacios. La segunda línea del código admite todos los caracteres excepto letras en minúsculas.

```
my_txt.restrict = "A-Z 0-9"; // limitar control a letras en mayúsculas, números  
y espacios  
my_txt.restrict = "^a-z"; // permitir todos los caracteres, excepto letras en  
minúsculas
```

TextArea.text

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

textAreaInstance.text

Descripción

Propiedad; contenido de texto de un componente `TextArea`. El valor predeterminado es "" (cadena vacía).

Ejemplo

El código siguiente coloca una cadena en la instancia `myTextArea` y luego traza dicha cadena en el panel Salida:

```
myTextArea.text = "The Royal Nonesuch";  
trace(myTextArea.text); // traza "The Royal Nonesuch"
```

TextArea.vPosition

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

textAreaInstance.vPosition

Descripción

Propiedad; define la posición vertical del texto de un campo de texto. La propiedad scroll es útil para dirigir a los usuarios a un párrafo específico en un pasaje largo, o para crear campos de texto con desplazamiento. Es posible obtener y definir esta propiedad. El valor predeterminado es 0.

Ejemplo

El código siguiente muestra los caracteres situados en el extremo superior del campo:

```
myTextArea.vPosition = 0;
```

TextArea.vScrollPolicy

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

textAreaInstance.vScrollPolicy

Descripción

Propiedad; determina si la barra de desplazamiento vertical está presente siempre ("on"), nunca ("off") o aparece automáticamente en función del tamaño del campo ("auto"). El valor predeterminado es "auto".

Ejemplo

El código siguiente mantiene siempre activas las barras de desplazamiento vertical:

```
text.vScrollPolicy = "off";
```

TextArea.wordWrap

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

`textAreaInstance.wordWrap`

Descripción

Propiedad; valor booleano que indica si el texto se ajusta (`true`) o no (`false`). El valor predeterminado es `true`.

Componente TextInput

TextInput es un componente de una sola línea que ajusta el objeto TextField nativo de ActionScript. Puede utilizar estilos para personalizar el componente TextInput; cuando se desactiva una instancia, su contenido se muestra en un color representado por el estilo “disabledColor”. El componente TextInput se puede formatear también con HTML o como un campo de contraseña que disfraza el texto.

Es posible activar o desactivar el componente TextInput en una aplicación. Si está desactivado, no recibe la entrada del ratón ni del teclado. Cuando está activado, sigue las mismas reglas de selección y navegación que un objeto TextField de ActionScript. Cuando la instancia de TextInput está seleccionada, puede controlarla con las teclas siguientes:

Tecla	Descripción
Teclas de flecha	Desplaza el carácter un carácter hacia la izquierda o la derecha.
Mayúsculas + Tabulador	Desplaza la selección al objeto anterior.
Tabulador	Desplaza la selección al objeto siguiente.

Para más información sobre el control de la selección, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 24](#) o [“Clase FocusManager” en la página 101](#).

La vista previa dinámica de cada instancia de TextInput refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o el panel Inspector de componentes. No es posible seleccionar texto en la vista previa, de igual modo que tampoco se puede introducir texto en la instancia del componente en el escenario.

Cuando se añade el componente TextInput a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de pantalla puedan acceder al mismo.

Utilización del componente TextInput

Puede utilizar un componente TextInput siempre que necesite un campo de texto de una sola línea. Si necesita un campo de texto con varias líneas, utilice [“Componente TextArea” en la página 204](#). Por ejemplo, si desea utilizar un componente TextInput como campo de contraseña en un formulario, puede definir un detector que compruebe si el campo tiene caracteres suficientes cuando el usuario utiliza el tabulador para salir del campo. El detector podría mostrar un mensaje de error indicando que se debe introducir el número de caracteres correcto.

Parámetros del componente TextInput

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente TextInput en el inspector de propiedades o el panel Inspector de componentes:

text especifica el contenido de TextInput. No es posible introducir retornos de carro en el inspector de propiedades ni en el panel Inspector de componentes. El valor predeterminado es "" (cadena vacía).

editable indica si el componente TextInput es editable (true) o no (false). El valor predeterminado es true.

password indica si el campo es un campo de contraseña (true) o no (false). El valor predeterminado es false.

Puede escribir código ActionScript para controlar éstas y otras opciones adicionales para los componentes de TextInput utilizando sus propiedades, métodos y eventos. Para más información, consulte [Clase TextInput](#).

Creación de aplicaciones con el componente TextInput

El procedimiento siguiente explica cómo añadir un componente TextInput a una aplicación durante la edición. En este ejemplo, el componente es un campo de contraseña con un detector de eventos que determina si se ha introducido el número de caracteres correcto.

Para crear una aplicación con el componente TextInput, siga este procedimiento:

- 1 Arrastre un componente TextInput desde el panel Componentes al escenario.
- 2 En el inspector de propiedades, introduzca el nombre de instancia **passwordField**.
- 3 En el inspector de propiedades, siga este procedimiento:

- Deje el parámetro text vacío.
- Defina el parámetro editable en true.
- Defina el parámetro password en true.

- 4 Seleccione el fotograma 1 de la línea de tiempo, abra el panel Acciones e introduzca el código siguiente:

```
textListener = new Object();
textListener.handleEvent = function (evt){
    if (evt.type == "enter"){
        trace("Debe introducir al menos 8 caracteres");
    }
}
passwordField.addEventListener("enter", textListener);
```

Este código define un controlador de eventos enter en la instancia passwordField de TextInput para comprobar que el usuario ha introducido el número de caracteres correcto.

- 5 Una vez introducido el texto en la instancia passwordField, puede obtener su valor como se muestra a continuación:

```
var login = passwordField.text;
```

Personalización del componente TextInput

El componente `TextInput` puede transformarse horizontalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice `UIObject.setSize()` o cualquiera de las propiedades y métodos aplicables de Clase `TextInput`.

Cuando se cambia el tamaño de un componente `TextInput`, el borde se ajusta al nuevo recuadro de delimitación. El componente `TextInput` no utiliza barras de desplazamiento, si bien el punto de inserción se desplaza automáticamente a medida que el usuario interactúa con el texto. Así, el tamaño del campo de texto se ajusta al interior del área restante; el componente `TextInput` carece de elementos con tamaño fijo. Si el componente `TextInput` es demasiado pequeño para mostrar el texto, éste se recorta.

Utilización de estilos con el componente TextInput

El componente `TextInput` tiene sus propiedades de estilo `backgroundColor` y `borderStyle` definidas en una declaración de estilo de clase. Los estilos de clase anulan los estilos `_global`, por tanto, si desea definir las propiedades de estilo `backgroundColor` y `borderStyle`, debe crear una declaración distinta de estilo predeterminado en la instancia.

Un componente `TextInput` admite los siguientes estilos:

Estilo	Descripción
<code>color</code>	Color predeterminado del texto.
<code>embedFonts</code>	Fuentes que incorporar al documento.
<code>fontFamily</code>	Nombre de la fuente del texto.
<code>fontSize</code>	Tamaño de la fuente en puntos.
<code>fontStyle</code>	Estilo de la fuente; puede ser “normal” o “italic”.
<code>fontWeight</code>	Grosor de la fuente; puede ser “normal” o “bold”.
<code>textAlign</code>	Alineación del texto; puede ser “left”, “right” o “center”.
<code>textDecoration</code>	Decoración del texto; puede ser “none” o “underline”.

Utilización de aspectos con el componente TextInput

El componente `TextArea` utiliza la clase `RectBorder` para dibujar su borde. Puede utilizar el método `setStyle()` (véase `UIObject.setStyle()`) para cambiar las propiedades de estilo `RectBorder` siguientes:

Estilos <code>RectBorder</code>
<code>borderColor</code>
<code>highlightColor</code>
<code>borderColor</code>
<code>shadowColor</code>
<code>borderCapColor</code>

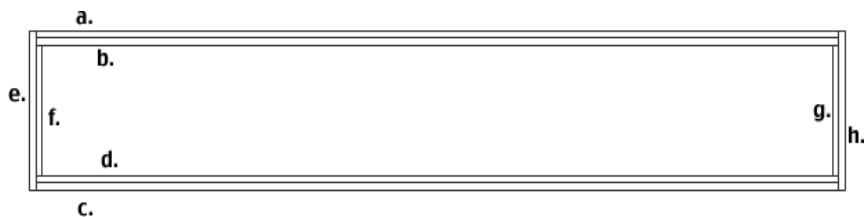
Estilos RectBorder

shadowCapColor

shadowCapColor

borderCapColor

Las propiedades de estilo definen las siguientes posiciones en el borde:



Clase TextInput

Herencia UIObject > UIComponent > TextInput

Namespace de clase de ActionScript mx.controls.TextInput

Las propiedades de la clase TextInput permiten definir el contenido, el formato y la posición horizontal del texto en tiempo de ejecución. También puede indicar si el campo es editable y si se trata de un campo “contraseña”, o restringir los caracteres que el usuario puede introducir.

Si una propiedad de la clase TextInput se define con ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o el panel Inspector de componentes.

El componente TextInput utiliza FocusManager para anular el rectángulo de selección predeterminado de Flash Player y dibuja uno personalizado con esquinas redondeadas. Para más información, consulte [“Clase FocusManager” en la página 101](#).

El componente TextInput admite estilos CSS y cualquier otro estilo adicional de HTML admitido por Flash Player. Para información sobre la compatibilidad con CSS, consulte la [especificación W3C](#).

Puede manipular la cadena de texto utilizando la cadena devuelta por el objeto de texto.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.controls.TextInput.version);
```

Nota: el código siguiente devuelve undefined: `trace(myTextInputInstance.version);`.

Resumen de métodos de la clase TextInput

Hereda todos los métodos de [Clase UIObject](#) y [Clase UIComponent](#).

Resumen de propiedades de la clase TextInput

Propiedad	Descripción
<code>TextInput.editable</code>	Valor booleano que indica si el campo es editable (<code>true</code>) o no (<code>false</code>).
<code>TextInput.hPosition</code>	Posición del desplazamiento horizontal del campo de texto.
<code>TextInput.length</code>	Número de caracteres de un campo de texto de <code>TextInput</code> . Es una propiedad de sólo lectura.
<code>TextInput.maxChars</code>	Número máximo de caracteres que el usuario puede introducir en un campo de texto de <code>TextInput</code> .
<code>TextInput.maxHPosition</code>	Valor máximo posible para <code>TextField.hPosition</code> . Esta propiedad es de sólo lectura.
<code>TextInput.password</code>	Valor booleano que indica si el campo de texto de entrada es un campo de contraseña que oculta los caracteres introducidos.
<code>TextInput.restrict</code>	Indica los caracteres que el usuario puede introducir en un campo de texto.
<code>TextInput.text</code>	Define el contenido de texto de un campo de texto de <code>TextInput</code> .

Hereda todos los métodos de [Clase UIObject](#) y [Clase UIComponent](#).

Resumen de eventos de la clase TextInput

Evento	Descripción
<code>TextInput.change</code>	Se activa cuando cambia el campo Input.
<code>TextInput.enter</code>	Se activa cuando se presiona la tecla Intro.

Hereda todos los métodos de [Clase UIObject](#) y [Clase UIComponent](#).

TextInput.change

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(change){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.change = function(eventObject){  
    ...  
}  
textInputInstance.addEventListener("change", listenerObject)
```

Descripción

Evento; notifica a los detectores que el texto ha cambiado. Este evento se difunde una vez modificado el texto. No se puede utilizar para impedir que ciertos caracteres se añadan al campo de texto del componente; para ello, deberá utilizar `TextInput.restrict`. Este evento se activa sólo por una entrada del usuario, no por un cambio provocado por `ActionScript`.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente `TextInput`. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado a la instancia `myTextInput`, envía “_level0.myTextInput” al panel Salida:

```
on(change){
    trace(this);
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*textInputInstance*) distribuye un evento (en este caso `change`) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (*eventObject*) al método del objeto detector. Cada objeto `Event` tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en la [página 236](#).

Ejemplo

En este ejemplo se define una etiqueta en la aplicación que indica si el contenido del campo `TextInput` ha sufrido cambios:

```
form.change = function(eventObj){
    // eventObj.target es el componente que ha generado el evento change,
    // es decir, el componente Input.
    myFormChanged.visible = true; // definir un indicador de cambios si se
    cambia el contenido;
}
myInput.addEventListener("change", form);
```

Véase también

[UIEventDispatcher.addEventListener\(\)](#)

TextInput.editable

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

textInputInstance.editable

Descripción

Propiedad; valor booleano que indica si el componente es editable (`true`) o no (`false`). El valor predeterminado es `true`.

TextInput.enter

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(enter){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.enter = function(eventObject){  
    ...  
}  
textInputInstance.addEventListener("enter", listenerObject)
```

Descripción

Evento; notifica al detector que se ha presionado la tecla Intro.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente `TextInput`. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado a la instancia `myTextInput`, envía “_level0.myTextInput” al panel Salida:

```
on(enter){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (`textInputInstance`) distribuye un evento (en este caso `enter`) y una función asociada al objeto detector creado (`listenerObject`) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (`eventObject`) al método del objeto detector. Cada objeto `Event` tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en [la página 236](#).

Ejemplo

En este ejemplo se define una etiqueta en la aplicación que indica si el contenido del campo `TextInput` ha sufrido cambios:

```
form.enter = function(eventObj){
    // eventObj.target es el componente que genera el evento enter,
    // es decir, el componente Input.
    myFormChanged.visible = true;
    // definir un indicador de cambios si el usuario presiona la tecla Intro;
}
myInput.addEventListener("enter", form);
```

Véase también

`UIEventDispatcher.addEventListener()`

TextInput.hPosition

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

textInputInstance.hPosition

Descripción

Propiedad; define la posición horizontal del texto dentro del campo. El valor predeterminado es 0.

Ejemplo

El código siguiente muestra los caracteres situados en el extremo izquierdo del campo:

```
myTextInput.hPosition = 0;
```

TextInput.length

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

inputInstance.length

Descripción

Propiedad (sólo lectura); valor que indica el número de caracteres de un componente `TextInput`. El carácter de tabulador (“\t”) cuenta como un carácter. El valor predeterminado es 0.

Ejemplo

El código siguiente determina el número de caracteres de la cadena `myTextInput` y lo copia en la variable `length`:

```
var length = myTextInput.length;
```

TextInput.maxChars

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

textInputInstance.maxChars

Descripción

Propiedad; indica el número máximo de caracteres que puede contener el campo de texto. Un script puede insertar más texto del que permite `maxChars`; la propiedad `maxChars` sólo indica cuánto texto puede introducir el usuario. Si el valor de esta propiedad es `null`, no hay límite en cuanto a la cantidad de texto que puede introducirse. El valor predeterminado es `null`.

Ejemplo

En el ejemplo siguiente, se limita a 255 el número de caracteres que puede introducir el usuario:

```
myTextInput.maxChars = 255;
```

TextInput.maxHPosition

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

textInputInstance.maxHPosition

Descripción

Propiedad (sólo lectura); indica el valor máximo de `TextInput.hPosition`. El valor predeterminado es 0.

Ejemplo

El código siguiente se desplaza hasta el extremo derecho:

```
myTextInput.hPosition = myTextInput.maxHPosition;
```


TextInput.password

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

textInputInstance.password

Descripción

Propiedad; valor booleano que indica si el campo de texto es un campo de contraseña (`true`) o no (`false`). Si el valor de `password` es `true`, el campo de texto es un campo de contraseña y oculta los caracteres de entrada. Si su valor es `false`, el campo de texto no es de contraseña. El valor predeterminado es `false`.

Ejemplo

El código siguiente convierte el campo de texto en un campo de contraseña y muestra todos los caracteres como asteriscos (*):

```
myTextInput.password = true;
```

TextInput.restrict

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

textInputInstance.restrict

Descripción

Propiedad; indica el conjunto de caracteres que los usuarios pueden introducir en el campo de texto. El valor predeterminado es `undefined`. Si el valor de la propiedad `restrict` es `null` o una cadena vacía (""), el usuario puede introducir cualquier carácter. Si el valor de la propiedad `restrict` es una cadena de caracteres, sólo podrán introducirse en el campo de texto; el texto se explora de izquierda a derecha. Puede especificarse un intervalo utilizando un guión (-).

La propiedad `restrict` sólo limita la interacción del usuario; un script puede introducir cualquier texto en el campo de texto. Esta propiedad no se sincroniza con las casillas de verificación de Incorporar contornos de fuentes del inspector de propiedades.

Si la cadena empieza por "^", inicialmente se aceptan todos los caracteres; los caracteres posteriores de la cadena se excluyen del conjunto de caracteres aceptados. Si la cadena no empieza por "^", inicialmente no se acepta ningún carácter; los caracteres posteriores de la cadena se incluyen en el conjunto de caracteres aceptados.

Puede utilizar el carácter barra invertida para introducir los caracteres “-”, “^” y “\”, como en el ejemplo siguiente:

```
\^  
\-  
\\
```

Cuando se introduce el carácter \ en el panel Acciones entre " " (comillas dobles), tiene un significado especial para el intérprete de comillas dobles del panel Acciones. Significa que el carácter después del carácter \ debe tratarse como tal. Por ejemplo, el siguiente código se utiliza para introducir una comilla sencilla:

```
var leftQuote = "\"";
```

El intérprete .restrict del panel Acciones también utiliza \ como un carácter de escape. Por tanto, es posible que crea que la siguiente expresión debe funcionar:

```
myText.restrict = "0-9\-\^\\\";
```

Sin embargo, el hecho de que esta expresión se encuentre entre comillas dobles, hace que el valor siguiente se envíe al intérprete .restrict: 0-9-^ y el intérprete .restrict no comprende este valor.

Esta expresión se debe incluir entre comillas dobles, pero no por ello debe proporcionarla sólo al intérprete .restrict, sino que también debe anular el intérprete incorporado del panel Acciones para las comillas dobles. Para enviar el valor 0-9\-\^\\ al intérprete .restrict, debe introducir el siguiente código:

```
myText.restrict = "0-9\\-\^\\\\\";
```

Ejemplo

En el ejemplo siguiente, la primera línea del código limita el campo de texto a letras en mayúsculas, números y espacios. La segunda línea del código admite todos los caracteres excepto letras en minúsculas.

```
my_txt.restrict = "A-Z 0-9";  
my_txt.restrict = "^a-z";
```

El código siguiente permite al usuario introducir los caracteres “0 1 2 3 4 5 6 7 8 9 - ^ \” en la instancia myText. Debe utilizar dos barras invertidas para anular los caracteres “-, ^ y \”. La primera “\” anula el “-”, la segunda “\” le indica al intérprete que el carácter siguiente no debe ser tratado como un carácter especial, como en el ejemplo siguiente:

```
myText.restrict = "0-9\\-\^\\\\\";
```

TextInput.text

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

textInputInstance.text

Descripción

Propiedad; contenido de texto de un componente TextInput. El valor predeterminado es "" (cadena vacía).

Ejemplo

El código siguiente coloca una cadena en la instancia myTextInput y luego traza dicha cadena en el panel Salida:

```
myTextInput.text = "The Royal Nonesuch";  
trace(myTextInput.text); // trazar "The Royal Nonesuch"
```

Componente Tree

Para obtener información reciente sobre esta función, haga clic en el botón Actualizar en la parte superior de la ficha Ayuda.

Clase UIComponent

Herencia UIObject > UIComponent

Namespace de la clase de ActionScript mx.core.UIComponent

Todos los componentes de la v2 amplían UIComponent; no es un componente visual. La clase UIComponent contiene funciones y propiedades que permiten a los componentes de Macromedia compartir algunos comportamientos comunes. La clase UIComponent permite realizar lo siguiente:

- Recibir selecciones y entradas del teclado
- Activar y desactivar componentes
- Cambiar el tamaño mediante el diseño

Para utilizar los métodos y propiedades de UIComponent, puede llamarlos directamente desde cualquier componente en uso. Por ejemplo, para llamar al método `UIComponent.setFocus()` desde el componente RadioButton, deberá escribir el código siguiente:

```
myRadioButton.setFocus();
```

Sólo necesita crear una instancia de UIComponent si está utilizando la arquitectura de componentes de Macromedia V2 para crear un componente nuevo. Incluso en tal caso, UIComponent se crea de forma implícita por medio de otras subclases, por ejemplo Button. Si necesita crear una instancia de UIComponent, utilice el código siguiente:

```
class MyComponent extends UIComponent;
```

Resumen de métodos de la clase UIComponent

Método	Descripción
<code>UIComponent.setFocus()</code>	Devuelve una referencia al objeto seleccionado.
<code>UIComponent.setFocus()</code>	Define la selección en la instancia de componente.

Hereda todos los métodos de la [Clase UIObject](#).

Resumen de propiedades de la clase `UIComponent`

Propiedad	Descripción
<code>UIComponent.enabled</code>	Indica si el componente puede recibir selecciones y entradas.
<code>UIComponent.tabIndex</code>	Número que indica el orden de tabulación para un componente de un documento.

Hereda todas las propiedades de la [Clase `UIObject`](#).

Resumen de eventos de la clase `UIComponent`

Evento	Descripción
<code>UIComponent.focusIn</code>	Se difunde cuando se selecciona un objeto.
<code>UIComponent.focusOut</code>	Se difunde cuando un objeto deja de seleccionarse.
<code>UIComponent.keyDown</code>	Se difunde cuando se presiona una tecla.
<code>UIComponent.keyUp</code>	Se difunde cuando se suelta una tecla.

Hereda todos los eventos de la [Clase `UIObject`](#).

`UIComponent.focusIn`

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
on(focusIn){
    ...
}
listenerObject = new Object();
listenerObject.focusIn = function(eventObject){
    ...
}
componentInstance.addEventListener("focusIn", listenerObject)
```

Descripción

Evento; notifica al detector que el objeto se ha seleccionado con el teclado.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente.

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*componentInstance*) distribuye un evento (en este caso *focusIn*) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto *Event* (*eventObject*) al método del objeto detector. Cada objeto *Event* tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en [la página 236](#).

Ejemplo

El código siguiente desactiva un botón mientras el usuario escribe en el campo de texto `txt`:

```
txtListener.handleEvent = function(eventObj) {  
    form.button.enabled = false;  
}  
txt.addEventListener("focusIn", txtListener);
```

Véase también

`UIEventDispatcher.addEventListener()`

UIComponent.focusOut

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
on(focusOut){  
    ...  
}  
listenerObject = new Object();  
listenerObject.focusOut = function(eventObject){  
    ...  
}  
componentInstance.addEventListener("focusOut", listenerObject)
```

Descripción

Evento; notifica a los detectores que el objeto ya no tiene la selección del teclado.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente.

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*componentInstance*) distribuye un evento (en este caso `focusOut`) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (*eventObject*) al método del objeto detector. Cada objeto `Event` tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en la página 236.

Ejemplo

El código siguiente activa un botón cuando el usuario abandona el campo de texto `txt`:

```
txtListener.handleEvent = function(eventObj) {  
    if (eventObj.type == focusOut){  
        form.button.enabled = true;  
    }  
}  
txt.addEventListener("focusOut", txtListener);
```

Véase también

`UIEventDispatcher.addEventListener()`

UIComponent.enabled

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

componentInstance.enabled

Descripción

Propiedad; indica si el componente puede aceptar selecciones y entradas del ratón. Si el valor es `true`, puede recibir selecciones y entradas; si el valor es `false`, no puede. El valor predeterminado es `true`.

Ejemplo

En el ejemplo siguiente se establece la propiedad `enabled` de un componente `CheckBox` en `false`:

```
checkBoxInstance.enabled = false;
```

UIComponent.setFocus()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
componentInstance.setFocus();
```

Parámetros

Ninguno.

Valor devuelto

Referencia al objeto que recibe actualmente la selección.

Descripción

Método; devuelve una referencia al objeto que recibe la selección del teclado.

Ejemplo

El código siguiente devuelve una referencia al objeto que recibe la selección que asigna a la variable tmp:

```
var tmp = checkbox.setFocus();
```

UIComponent.keyDown

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
on(keyDown){  
    ...  
}  
listenerObject = new Object();  
listenerObject.keyDown = function(eventObject){  
    ...  
}  
componentInstance.addEventListener("keyDown", listenerObject)
```

Descripción

Evento; notifica a los detectores que se ha presionado una tecla. Éste es un evento de nivel muy bajo que no debería utilizarse a menos que sea necesario, ya que puede afectar al rendimiento del sistema.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente.

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*componentInstance*) distribuye un evento (en este caso `keyDown`) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto *Event* (*eventObject*) al método del objeto detector. Cada objeto *Event* tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en la página 236.

Ejemplo

El código siguiente hace que un icono parpadee cuando se presiona una tecla:

```
formListener.handleEvent = function(eventObj)
{
    form.icon.visible = !form.icon.visible;
}
form.addEventListener("keyDown", formListener);
```

UIComponent.keyUp

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
on(keyUp){
    ...
}
listenerObject = new Object();
listenerObject.keyUp = function(eventObject){
    ...
}
componentInstance.addEventListener("keyUp", listenerObject)
```

Descripción

Evento; notifica a los detectores cuando se suelta una tecla. Éste es un evento de nivel muy bajo que no debería utilizarse a menos que sea necesario, ya que puede afectar al rendimiento del sistema.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente.

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*componentInstance*) distribuye un evento (en este caso `keyUp`) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto *Event* (*eventObject*) al método del objeto detector. Cada objeto *Event* tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en [la página 236](#).

Ejemplo

El código siguiente hace que un icono parpadee cuando se suelta una tecla:

```
formListener.handleEvent = function(eventObj)
{
    form.icon.visible = !form.icon.visible;
}
form.addEventListener("keyUp", formListener);
```

UIComponent.setFocus()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
componentInstance.setFocus();
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; define la selección en esta instancia del componente. La instancia con la selección recibe todas las entradas del teclado.

Ejemplo

El código siguiente define la selección sobre la instancia `checkbox`:

```
checkbox.setFocus();
```

UIComponent.tabIndex

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

instance.tabIndex

Descripción

Propiedad; número que indica el orden de tabulación para un componente en un documento.

Ejemplo

El código siguiente define el valor de `tmp` en la propiedad `tabIndex` de la instancia `checkbox`:

```
var tmp = checkbox.tabIndex;
```

Clase UIEventDispatcher

Namespace de clase de ActionScript `mx.events.EventDispatcher`; `mx.events.UIEventDispatcher`

Los eventos permiten conocer el momento en el que el usuario interactúa con un componente, así como los cambios importantes producidos en la apariencia o el ciclo de vida de un componente, como la creación o eliminación de un componente o sus cambios de tamaño.

Cada componente difunde distintos eventos, que aparecen en la entrada de cada componente. Hay varias formas de utilizar eventos de componentes en un código ActionScript. Para más información, consulte [“Eventos de componentes” en la página 22](#).

Utilice `UIEventDispatcher.addEventListener()` para registrar un detector con una instancia de componente. El detector se invoca cuando se activa un evento del componente.

UIEventDispatcher.addEventListener()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004 y Flash MX Professional 2004.

Sintaxis

componentInstance.addEventListener(event, listener)

Parámetros

event Cadena que es el nombre del evento.

listener Referencia a un objeto detector o a una función.

Valor devuelto

Ninguno.

Descripción

Método; registra un objeto detector con una instancia de componente que difunde un evento. Cuando el evento se activa, se notifica al objeto detector o a la función. Se puede invocar este método desde cualquier instancia de componente. Por ejemplo, el código siguiente registra un detector para la instancia de componente `myButton`:

```
myButton.addEventListener("click", myListener);
```

Es necesario definir el detector (tanto si es un objeto como una función) antes de llamar a `addEventListener()` para registrar el detector con la instancia de componente. Si el detector es un objeto, debe tener definida una función callback que se invoque al activarse el evento. Por lo general, la función callback tiene el mismo nombre que el evento con el que se ha registrado el detector. Si el detector es una función, ésta se invoca cuando se activa el evento. Para más información, consulte [“Utilización de detectores de eventos de componentes” en la página 23](#).

Es posible registrar varios detectores para una sola instancia de componente, pero es preciso utilizar una llamada individual al `addEventListener()` de cada detector. Asimismo, se puede registrar un detector para varias instancias de componente, pero hay que utilizar una llamada individual al `addEventListener()` de cada instancia. Por ejemplo, el siguiente código define un objeto detector y lo asigna a dos instancias de componente `Button`:

```
lo = new Object();
lo.click = function(evt){
    if (evt.target == button1){
        trace("clic en botón 1");
    } else if (evt.target == button2){
        trace("clic en botón 2");
    }
}
button1.addEventListener("click", lo);
button2.addEventListener("click", lo);
```

Un objeto `Event` pasa al detector como un parámetro. El objeto `Event` tiene propiedades que contienen información sobre el evento producido. Es posible utilizar el objeto `Event` dentro de la función callback del detector para acceder a la información sobre el tipo de evento que tuvo lugar y qué instancia difundió el evento. En el ejemplo anterior, el objeto de evento es `evt` (se puede utilizar cualquier identificador como nombre del objeto `Event`) y se utiliza en las sentencias `if` para determinar sobre qué instancia de botón se ha hecho clic. Para más información, consulte [“Objetos Event” en la página 236](#).

Ejemplo

En el ejemplo siguiente se define un objeto detector, `myListener`, y el clic de la función callback. A continuación, llama a `addEventListener()` para registrar el objeto detector `myListener` con la instancia del componente `myButton`. Para probar este código, coloque un componente `Button` en el escenario con el nombre de instancia `myButton` e introduzca el código siguiente en el fotograma 1:

```
myListener = new Object();
myListener.click = function(evt){
    trace(evt.type + " activado");
}
myButton.addEventListener("click", myListener);
```

Objetos Event

Un objeto Event pasa al detector como un parámetro. El objeto Event es un objeto de ActionScript que tiene propiedades donde se incluye información acerca del evento que ha tenido lugar. El objeto Event se puede utilizar dentro de la función callback del detector para acceder al nombre del evento que se difundió o al nombre de instancia del componente que difundió el evento. Por ejemplo, el código siguiente utiliza la propiedad `target` del objeto de evento `evtObj` para acceder a la propiedad `label` de la instancia `myButton` y enviar el valor al panel Salida:

```
listener = new Object();
listener.click = function(evtObj){
    trace("clic en el botón " + evtObj.target.label + " button was clicked");
}
myButton.addEventListener("click", listener);
```

Algunas propiedades de objetos Event se definen en la [especificación W3C](#), pero no están implementadas en la versión 2 (v2) de la arquitectura de componentes de Macromedia. Cada objeto Event de la v2 tiene las propiedades que se especifican en la tabla que se muestra a continuación. Algunos eventos tienen propiedades adicionales definidas y, en este caso, las propiedades se enumeran en la entrada del evento.

Propiedades del objeto Event

Propiedad	Descripción
<code>type</code>	Cadena que indica el nombre del evento.
<code>target</code>	Referencia a la instancia del componente que difunde el evento.

Clase UIObject

Herencia MovieClip > UIObject

Namespace de la clase de ActionScript mx.core.UIObject

UIObject es la clase base para todos los componentes de v2; no es un componente visual. La clase UIObject ajusta el objeto MovieClip de ActionScript y contiene funciones y propiedades que permiten a los componentes de la v2 de Macromedia compartir algunos comportamientos comunes. La clase UIObject incorpora lo siguiente:

- Estilos
- Eventos
- Cambiar el tamaño mediante la escala

Para utilizar los métodos y propiedades de UIObject, puede llamarlos directamente desde cualquier componente en uso. Por ejemplo, para llamar al método `UIObject.setSize()` desde el componente `RadioButton`, deberá escribir el código siguiente:

```
myRadioButton.setSize(30, 30);
```

Si utiliza la arquitectura de componentes de Macromedia V2, sólo necesita crear una instancia de UIObject para crear un componente nuevo. Incluso en este caso, UIObject se crea con frecuencia de forma implícita por medio de otras subclases, por ejemplo `Button`. Si necesita crear una instancia de UIObject, utilice el código siguiente:

```
class MyComponent extends UIObject;
```

Resumen de métodos de la clase UIObject

Método	Descripción
<code>UIObject.createObject()</code>	Crea un subobjeto en un objeto.
<code>UIObject.createClassObject()</code>	Crea un objeto en la clase especificada.
<code>UIObject.destroyObject()</code>	Elimina una instancia de componente.
<code>UIObject.invalidate()</code>	Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.
<code>UIObject.move()</code>	Mueve el objeto a la posición indicada.
<code>UIObject.redraw()</code>	Fuerza la validación del objeto, de forma que se pueda volver a dibujar sobre el fotograma actual.
<code>UIObject.setSize()</code>	Cambia el tamaño del objeto al indicado.
<code>UIObject.setSkin()</code>	Define un aspecto en el objeto.

Resumen de propiedades de la clase UIObject

Propiedad	Descripción
<code>UIObject.bottom</code>	Devuelve la posición del borde inferior del objeto con respecto al borde inferior de su principal correspondiente.
<code>UIObject.height</code>	Altura del objeto, expresada en píxeles.
<code>UIObject.left</code>	Posición izquierda del objeto, expresada en píxeles.
<code>UIObject.right</code>	Posición del borde derecho del objeto con respecto al borde derecho de su principal correspondiente.
<code>UIObject.scaleX</code>	Número que indica el factor de escala en la dirección x del objeto con respecto a su principal correspondiente.
<code>UIObject.scaleY</code>	Número que indica el factor de escala en la dirección y del objeto con respecto a su principal correspondiente.
<code>UIObject.top</code>	Posición del borde superior del objeto con respecto a su principal correspondiente.
<code>UIObject.visible</code>	Valor booleano que indica si el objeto es visible (<code>true</code>) o no (<code>false</code>).
<code>UIObject.width</code>	Anchura del objeto, expresada en píxeles.
<code>UIObject.x</code>	Posición izquierda del objeto, expresada en píxeles.
<code>UIObject.y</code>	Devuelve la posición del borde superior del objeto con respecto a su principal correspondiente.

Resumen de eventos de la clase UIObject

Evento	Descripción
<code>UIObject.draw</code>	Se difunde cuando un objeto está a punto de dibujar sus gráficos.
<code>UIObject.load</code>	Se difunde cuando se han creado subobjetos.
<code>UIObject.move</code>	Se difunde cuando se ha movido el objeto.
<code>UIObject.resize</code>	Se difunde cuando se han descargado los subobjetos.
<code>UIObject.unload</code>	Se difunde cuando se han descargado los subobjetos.

UIObject.bottom

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

`componentInstance.bottom`

Descripción

Propiedad (sólo lectura); número expresado en píxeles que indica la posición inferior del objeto con respecto a la parte inferior de su principal correspondiente.

Ejemplo

Define el valor de tmp en la posición inferior de la casilla de verificación:

```
var tmp = checkbox.bottom;
```

UIObject.createObject()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

`componentInstance.createObject(linkageName, instanceName, depth, initObject)`

Parámetros

linkageName Cadena que indica el identificador de vinculación de un símbolo en el panel Biblioteca.

instanceName Cadena que indica el nombre de instancia de la instancia nueva.

depth Número que indica la profundidad de la nueva instancia.

initObject Objeto que contiene propiedades de inicialización para la nueva instancia.

Valor devuelto

UIObject que es una instancia del símbolo.

Descripción

Método; crea un subobjeto en un objeto. Por lo general, sólo lo utilizan los desarrolladores avanzados o de componentes.

Ejemplo

En el ejemplo siguiente se crea una instancia CheckBox en el objeto form:

```
form.createObject("CheckBox", "sym1", 0);
```

UIObject.createClassObject()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
componentInstance.createClassObject(className, instanceName, depth,
    initObject)
```

Parámetros

className Objeto que indica la clase de la nueva instancia.

instanceName Cadena que indica el nombre de instancia de la instancia nueva.

depth Número que indica la profundidad de la nueva instancia.

initObject Objeto que contiene propiedades de inicialización para la nueva instancia.

Valor devuelto

UIObject que es una instancia de la clase especificada.

Descripción

Método; crea un subobjeto de un objeto. Por lo general, sólo lo utilizan los desarrolladores avanzados o de componentes. Este método permite crear componentes en tiempo de ejecución.

Debe especificar el nombre de paquete de clase. Siga uno de estos procedimientos:

```
import mx.controls.Button;
createClassObject(Button,"button2",5,{label:"Test Button"});
```

o bien

```
createClassObject(mx.controls.Button,"button2",5,{label:"Test Button"});
```

Ejemplo

En el ejemplo siguiente se crea un objeto CheckBox:

```
form.createClassObject(CheckBox, "cb", 0, {label:"Check this"});
```

UIObject.destroyObject()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
componentInstance.destroyObject(instanceName)
```

Parámetros

instanceName Cadena que indica el nombre de instancia del objeto que se va a eliminar.

Valor devuelto

Ninguno.

Descripción

Método; elimina una instancia de componente.

UIObject.draw

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
on(draw){  
    ...  
}  
listenerObject = new Object();  
listenerObject.draw = function(eventObject){  
    ...  
}  
componentInstance.addEventListener("draw", listenerObject)
```

Descripción

Evento; notifica a los detectores que el objeto está a punto de dibujar sus gráficos. Se trata de un evento de nivel muy bajo que no debería utilizarse a menos que sea necesario, ya que puede afectar al rendimiento del sistema.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente.

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*componentInstance*) distribuye un evento (en este caso *draw*) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto *Event* (*eventObject*) al método del objeto detector. Cada objeto *Event* tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en [la página 236](#).

Ejemplo

El código siguiente vuelve a dibujar el objeto `form2` cuando se ha dibujado el objeto `form`:

```
formListener.draw = function(eventObj){
    form2.redraw(true);
}
form.addEventListener("draw", formListener);
```

Véase también

`UIEventDispatcher.addEventListener()`

UIObject.height

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

componentInstance.height

Descripción

Propiedad (sólo lectura); número que indica la altura del objeto, expresada en píxeles.

Ejemplo

En el ejemplo siguiente se define el valor de `tmp` en la altura de la instancia `checkbox`:

```
var tmp = checkbox.height;
```

UIObject.getStyle()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

componentInstance.getStyle(propertyName)

Parámetros

propertyName Cadena que indica el nombre de la propiedad del estilo (por ejemplo, "fontWeight", "borderStyle", y así sucesivamente).

Valor devuelto

Valor de la propiedad del estilo. El valor puede ser cualquier tipo de datos.

Descripción

Método; obtiene la propiedad del estilo procedente de styleDeclaration o del objeto. Si la propiedad del estilo es un estilo heredado, los principales del objeto pueden ser el origen del valor del estilo.

Para ver una lista de los estilos admitidos por cada componente, consulte sus entradas individuales.

Ejemplo

El código siguiente define la propiedad de estilo `fontWeight` de la instancia `ib` en negrita cuando la propiedad de estilo `fontWeight` de la instancia `cb` sea negrita:

```
if (cb.getStyle("fontWeight") == "bold")
{
    ib.setStyle("fontWeight", "bold");
};
```

UIObject.invalidate()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

componentInstance.invalidate()

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.

Ejemplo

En el ejemplo siguiente se marca la instancia pBar de ProgressBar para volver a dibujarla:

```
pBar.invalidate();
```

UIObject.left

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

componentInstance.left

Descripción

Propiedad (sólo lectura); número que indica el borde izquierdo del objeto, expresado en píxeles.

Ejemplo

En el ejemplo siguiente se establece el valor de tmp en la posición izquierda de la instancia checkbox:

```
var tmp = checkbox.left; // establecer el valor de tmp en la posición izquierda  
de la casilla de verificación;
```

UIObject.load

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(load){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.load = function(eventObject){  
    ...  
}  
componentInstance.addEventListener("load", listenerObject)
```

Descripción

Evento; notifica a los detectores que se ha creado el subobjeto de este objeto.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente.

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*componentInstance*) distribuye un evento (en este caso *load*) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto *Event* (*eventObject*) al método del objeto detector. Cada objeto *Event* tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 236](#).

Ejemplo

En el ejemplo siguiente se crea una instancia de *MySymbol* una vez cargada la instancia *form*:

```
formListener.handleEvent = function(eventObj)
{
    form.createObject("MySymbol", "sym1", 0);
}
form.addEventListener("load", formListener);
```

UIObject.move

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(move){
    ...
}
```

Sintaxis 2:

```
listenerObject = new Object();
listenerObject.move = function(eventObject){
    ...
}
componentInstance.addEventListener("move", listenerObject)
```

Descripción

Evento; notifica a los detectores que el objeto se ha movido.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente.

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*componentInstance*) distribuye un evento (en este caso *move*) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto *Event* (*eventObject*) al método del objeto detector. Cada objeto *Event* tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en [la página 236](#).

Ejemplo

En el ejemplo siguiente se llama al método `move()` para mantener a `form2` 100 píxeles por debajo y a la derecha de `form1`:

```
formListener.handleEvent = function(eventObj)
{
    // eventObj.target es el componente que genera el evento change
    form2.move(form1.x + 100, form1.y + 100);
}
form1.addEventListener("move", formListener);
```

UIObject.move()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

componentInstance.move(*x*, *y*)

Parámetros

- x* Número que indica la posición de la esquina superior izquierda del objeto con respecto a su principal correspondiente.
- y* Número que indica la posición de la esquina superior izquierda del objeto con respecto a su principal correspondiente.

Valor devuelto

Ninguno.

Descripción

Método; mueve el objeto hasta la posición indicada. Debe pasar sólo valores enteros a `UIObject.move()`, o de lo contrario el componente puede aparecer borroso.

De no seguirse estas reglas se pueden obtener controles con aspecto borroso.

Ejemplo

En este ejemplo se mueve la instancia `pBar` de `ProgressBar` a la esquina superior izquierda, a 100, 100:

```
pBar.move(100, 100);
```

UIObject.redraw()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
componentInstance.redraw()
```

Parámetros

always Valor booleano; `true` si se vuelve a dibujar siempre, `false` si sólo se vuelve a dibujar cuando se activa el método `invalidate`.

Valor devuelto

Ninguno.

Descripción

Método; fuerza la validación del objeto, de forma que se pueda volver a dibujar sobre el fotograma actual.

Ejemplo

En este ejemplo se fuerza la instancia `pBar` para que se vuelva a dibujar inmediatamente:

```
pBar.validate(true);
```

UIObject.resize

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(resize){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();
listenerObject.resize = function(eventObject){
    ...
}
componentInstance.addEventListener("resize", listenerObject)
```

Descripción

Evento; notifica a los detectores que se están descargando los subobjetos de este objeto.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente.

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*componentInstance*) distribuye un evento (en este caso `resize`) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (*eventObject*) al método del objeto detector. Cada objeto `Event` tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 236](#).

Ejemplo

En el ejemplo siguiente se llama al método `setSize()` para que `sym1` sea la mitad de ancho y un cuarto de alto cuando `form` se mueva:

```
formListener.handleEvent = function(eventObj) {
    form.sym1.setSize(sym1.width / 2, sym1.height / 4);
}
form.addEventListener("resize", formListener);
```

UIObject.right

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
componentInstance.right
```

Descripción

Propiedad (sólo lectura); número expresado en píxeles que indica la posición derecha del objeto con respecto a la parte derecha de su principal.

Ejemplo

En el ejemplo siguiente se define el valor de `tmp` en la posición derecha de la instancia `checkbox`:

```
var tmp = checkbox.right;
```

UIObject.scaleX

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

componentInstance.scaleX

Descripción

Propiedad; número que indica el factor de escala en la dirección *x* del objeto con respecto a su principal correspondiente.

Ejemplo

En el ejemplo siguiente se hace que la casilla de verificación doble su anchura y se define la variable `tmp` en el factor de escala horizontal:

```
checkbox.scaleX = 200;  
var tmp = checkbox.scaleX;
```

UIObject.scaleY

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

componentInstance.scaleY

Descripción

Propiedad; número que indica el factor de escala en la dirección *y* del objeto con respecto a su principal.

Ejemplo

En el ejemplo siguiente se hace que la casilla de verificación doble su altura y se define la variable `tmp` en el factor de escala vertical:

```
checkbox.scaleY = 200;  
var tmp = checkbox.scaleY;
```


UIObject.setSize()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
componentInstance.setSize(width, height)
```

Parámetros

width Número que indica la anchura del objeto, expresada en píxeles.

height Número que indica la altura del objeto, expresada en píxeles.

Valor devuelto

Ninguno.

Descripción

Método; cambia el tamaño del objeto por el tamaño especificado. Debe pasar sólo valores enteros a `UIObject.setSize()` o de lo contrario el componente puede aparecer borroso. Este método (y todos los métodos y propiedades de UIObject) está disponible desde cualquier instancia de componente.

La llamada a este método desde una instancia de ComboBox hace que se ajuste el tamaño del cuadro combinado y cambie la propiedad `rowHeight` de la lista que contiene.

Ejemplo

En este ejemplo se cambia el tamaño de la instancia del componente `pBar` a 100 píxeles de ancho y 100 píxeles de alto:

```
pBar.setSize(100, 100);
```

UIObject.setSkin()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
componentInstance.setSkin(id, linkageName)
```

Parámetros

id Número que indica la variable. Este valor suele ser una constante definida en la definición de la clase.

linkageName Cadena que indica un elemento de la biblioteca.

Valor devuelto

Ninguno.

Descripción

Método; define un aspecto en la instancia de componente. Este método se utiliza cuando se desarrollan componentes. No es posible utilizar este método para definir los aspectos de un componente en tiempo de ejecución.

Ejemplo

En este ejemplo se define un aspecto en la instancia de `checkbox`:

```
checkbox.setSkin(CheckBox.skinIDCheckMark, "MyCustomCheckMark");
```

UIObject.setStyle()**Disponibilidad**

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
componentInstance.setStyle(propertyName, value)
```

Parámetros

propertyName Cadena que indica el nombre de la propiedad del estilo (por ejemplo, "fontWeight", "borderStyle", y así sucesivamente).

value Valor de la propiedad.

Valor devuelto

UIObject que es una instancia de la clase especificada.

Descripción

Método; define la propiedad de estilo en la declaración de estilos o en el objeto. Si la propiedad de estilo es un estilo heredado, la notificación del nuevo valor se enviará a los elementos secundarios.

Para ver una lista de los estilos admitidos por cada componente, consulte sus entradas individuales.

Ejemplo

El código siguiente define la propiedad de estilo `fontWeight` de la instancia de la casilla de verificación `cb` en negrita:

```
cb.setStyle("fontWeight", "bold");
```

UIObject.top**Disponibilidad**

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

componentInstance.top

Descripción

Propiedad (sólo lectura); número que indica el borde superior del objeto, expresado en píxeles.

Ejemplo

En el ejemplo siguiente se define la variable `tmp` en la posición superior de la instancia de la casilla de verificación:

```
var tmp = checkbox.top; // definir el valor de tmp en la posición superior de
    la casilla de verificación;
```

UIObject.unload

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(unload){
    ...
}
```

Sintaxis 2:

```
listenerObject = new Object();
listenerObject.unload = function(eventObject){
    ...
}
componentInstance.addEventListener("unload", listenerObject)
```

Descripción

Evento; notifica a los detectores que se están descargando los subobjetos de este objeto.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente.

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*componentInstance*) distribuye un evento (en este caso `unload`) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (*eventObject*) al método del objeto detector. Cada objeto `Event` tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en [la página 236](#).

Ejemplo

En el ejemplo siguiente se elimina `sym1` cuando se activa el evento `unload`:

```
formListener.handleEvent = function(eventObj) {  
    // eventObj.target es el componente que ha generado el evento change.  
    form.destroyObject(sym1);  
}  
form.addEventListener("unload", formListener);
```

UIObject.visible

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

componentInstance.visible

Descripción

Propiedad; valor booleano que indica si el objeto es visible (`true`) o no (`false`).

Ejemplo

En el ejemplo siguiente se hace que la instancia de `Loader` `myLoader` sea visible:

```
myLoader.visible = true;
```

UIObject.width

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

componentInstance.width

Descripción

Propiedad (sólo lectura); número que indica la anchura del objeto, expresada en píxeles.

Ejemplo

En el siguiente ejemplo se define la anchura del componente TextArea en 450 píxeles:

```
mytextarea.width = 450;
```

UIObject.x

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

componentInstance.x

Descripción

Propiedad (sólo lectura); número que indica el borde izquierdo del objeto, expresado en píxeles.

Ejemplo

En el siguiente ejemplo se define el borde izquierdo de la casilla de verificación en 150:

```
checkbox.x = 150;
```

UIObject.y

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

componentInstance.y

Descripción

Propiedad (sólo lectura); número que indica el borde superior del objeto, expresado en píxeles.

Ejemplo

En el siguiente ejemplo se define el borde superior de la casilla de verificación en 200:

```
checkbox.y = 200;
```

Paquete WebServices

Para obtener información reciente sobre esta función, haga clic en el botón Actualizar en la parte superior de la ficha Ayuda.

Componente WebServiceConnector

Para obtener información reciente sobre esta función, haga clic en el botón Actualizar en la parte superior de la ficha Ayuda.

Componente Window

El componente Window muestra el contenido de un clip de película en una ventana provista de una barra de título, un borde y un botón de cierre opcional.

El componente Window puede ser modal o amodal. Una ventana modal impide que las entradas realizadas con el ratón o el teclado se apliquen a otros componentes situados fuera de la ventana. También admite la función de arrastre, es decir, el usuario puede hacer clic en la barra de título y arrastrar la ventana y su contenido a otro lugar. El arrastre de los bordes no cambia el tamaño de la ventana.

Si se utiliza PopUpManager para añadir un componente Window a un documento, la instancia Window tendrá su propio FocusManager, distinto al resto del documento. Si no utiliza PopUpManager, el contenido de la ventana participa en la ordenación de la selección. Para más información sobre el control de la selección, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 24](#) o [“Clase FocusManager” en la página 101](#).

La vista previa dinámica de cada instancia de Window refleja los cambios realizados en todos los parámetros (excepto en contentPath) en el inspector de propiedades o el panel Inspector de componentes durante la edición.

Cuando se añade el componente Window a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de pantalla puedan acceder al mismo. En primer lugar, debe añadir la línea de código siguiente para activar la accesibilidad:

```
mx.accessibility.WindowAccImpl.enableAccessibility();
```

La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias. Para más información, consulte [“Creación de contenido accesible” en el apartado Utilización de Flash de la Ayuda](#). Es posible que tenga que actualizar el sistema de Ayuda para consultar esta información.

Utilización del componente Window

Puede utilizar una ventana en una aplicación siempre que desee ofrecer al usuario información o una opción que tenga prioridad sobre cualquier otro elemento de la aplicación. Por ejemplo, puede necesitar que un usuario rellene los datos de una ventana de conexión o de una ventana en la que se cambia una contraseña y se confirma una nueva.

Hay varias formas de añadir una ventana a una aplicación. Puede arrastrar un componente Windows desde el panel Componentes al escenario. También puede llamar a `createClassObject()` (véase [UIObject.createClassObject\(\)](#)) para añadir una ventana a una aplicación. La tercera forma de añadir ventanas a una aplicación consiste en utilizar [Clase PopUpManager](#). El uso de PopUpManager permite crear ventanas modales que se superponen a otros objetos del escenario. Para más información, consulte [Clase Window](#).

Parámetros del componente Window

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente Window en el inspector de propiedades o el panel Inspector de componentes:

contentPath especifica el contenido de la ventana. Puede ser el identificador de vinculación del clip de película o el nombre de símbolo de una pantalla, formulario o diapositiva que incluya el contenido de la ventana. También puede ser una URL absoluta o relativa a un archivo SWF o JPG que se carga en la ventana. El valor predeterminado es "". El contenido cargado se recorta para que quepa en la ventana.

title indica el título de la ventana.

closeButton indica si la ventana va a mostrar un botón de cierre (true) o no (false). Al hacer clic en el botón de cierre se difunde un evento `click`, pero no se cierra la ventana. Debe escribir un controlador que invoque a `Window.deletePopUp()` para cerrar la ventana explícitamente. Para más información sobre el evento `click`, consulte [Window.click](#).

Puede escribir código `JavaScript` para controlar éstas y otras opciones adicionales para los componentes de Window utilizando sus propiedades, métodos y eventos. Para más información, consulte [Clase Window](#).

Creación de aplicaciones con el componente Window

El procedimiento siguiente explica cómo añadir un componente Window a una aplicación. En este ejemplo se solicita al usuario que cambie su contraseña y confirme la nueva.

Para crear una aplicación con el componente Window, siga este procedimiento:

- 1 Cree un clip de película nuevo que contenga campos para insertar y confirmar una contraseña, además de botones Aceptar y Cancelar. Asigne al clip de película el nombre **PasswordForm**.
- 2 En la biblioteca, seleccione el clip de película PasswordForm y elija Vinculación en el menú de opciones.
- 3 Marque Exportar para `JavaScript` e introduzca **PasswordForm** en el cuadro Identificador.
- 4 Introduzca **mx.core.View** en el campo Clase de AS 2.0.
- 5 Arrastre un componente Window desde el panel Componentes hasta el escenario y elimine el componente del escenario. De esta manera, añadirá el componente a la biblioteca.
- 6 En la biblioteca, seleccione el archivo SWC de Window y elija Vinculación en el menú de opciones.
- 7 Seleccione Exportar para `JavaScript`.
- 8 Abra el panel Acciones e introduzca el controlador de acciones del ratón siguiente en el fotograma 1:

```
buttonListener = new Object();
buttonListener.click = function(){
    mx.managers.PopUpManager.createPopUp(_root, mx.containers.Window, true, {
        title:"Cambiar contraseña", contentPath:"PasswordForm" })
}
button.addEventListener("click", buttonListener);
```

Este controlador llama a `PopUpManager.createPopUp()` para crear una instancia del componente Window con la barra de título “Cambiar contraseña” que muestre el contenido del clip de película PasswordForm.

Personalización del componente Window

El componente Window puede transformarse horizontal o verticalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice `UIObject.setSize()` o cualquiera de las propiedades y métodos aplicables de la clase Window. Para más información, consulte [Clase Window](#).

Cuando se cambia el tamaño de la ventana, no se modifica el tamaño del botón de cierre ni del título. El título se alinea a la izquierda y el botón de cierre a la derecha.

Utilización de estilos con el componente Window

La declaración de estilos de la barra de título de un componente Window se indica mediante la propiedad `Window.titleStyleDeclaration`.

Un componente Window admite los siguientes estilos de Halo:

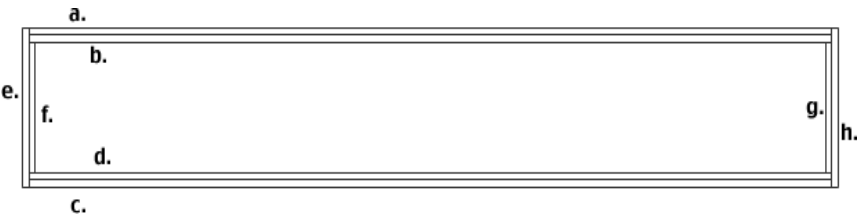
Estilo	Descripción
<code>borderStyle</code>	Borde del componente; "none", "inset", "outset" o "solid". Este estilo no hereda su valor.

Utilización de aspectos con el componente Window

El componente Window utiliza la clase `RectBorder` que se vale de la función de dibujo API de `ActionScript` para dibujar los bordes. Puede utilizar el método `setStyle()` (véase `UIObject.setStyle()`) para cambiar las propiedades de estilo `RectBorder` siguientes:

Estilos <code>RectBorder</code>
<code>borderColor</code>
<code>highlightColor</code>
<code>borderColor</code>
<code>shadowColor</code>
<code>borderCapColor</code>
<code>shadowCapColor</code>
<code>shadowCapColor</code>
<code>borderCapColor</code>

Las propiedades de estilo definen las siguientes posiciones en el borde:



Si utiliza `UIObject.createClassObject()` o `PopUpManager.createPopUp()` para crear una instancia de `Window` de forma dinámica (en tiempo de ejecución), podrá aplicar también el aspecto de forma dinámica. Para asignar aspectos a un componente en tiempo de ejecución, defina las propiedades de aspecto en el parámetro *initObject* que pasa al método `createClassObject()`. Estas propiedades de aspecto definen los nombres de los símbolos que se utilizarán como estados del botón, ya sea con o sin icono. Para más información, consulte `UIObject.createClassObject()` y `PopUpManager.createPopUp()`.

Un componente `Window` utiliza las propiedades de aspecto siguientes:

Propiedad	Descripción
<code>skinTitleBackground</code>	Barra de título. El valor predeterminado es <code>TitleBackground</code> .
<code>skinCloseUp</code>	Botón de cierre. El valor predeterminado es <code>CloseButtonUp</code> .
<code>skinCloseDown</code>	Botón de cierre en estado Presionado. El valor predeterminado es <code>CloseButtonDown</code> .
<code>skinCloseDisabled</code>	Botón de cierre en estado Desactivado. El valor predeterminado es <code>CloseButtonDisabled</code> .
<code>skinCloseOver</code>	Botón de cierre en estado Sobre. El valor predeterminado es <code>CloseButtonOver</code> .

Clase Window

Herencia `UIObject` > `UIComponent` > `View` > `ScrollView` > `Window`

Namespace de clase de ActionScript `mx.containers.Window`

Las propiedades de la clase `Window` permiten definir el título, añadir un botón de cierre y definir el contenido que deberá aparecer en tiempo de ejecución. Si una propiedad de la clase `Window` se define con `ActionScript`, sustituye el parámetro del mismo nombre definido en el inspector de propiedades o en el panel Inspector de componentes.

La mejor manera de crear una instancia de `Window` es llamando a `PopUpManager.createPopUp()`. Este método crea una ventana que puede ser modal (que solapa y desactiva los objetos existentes de la aplicación) o amodal. Por ejemplo, el código siguiente crea una instancia `Window` modal (el último parámetro indica modalidad):

```
var newWindow = PopUpManager.createPopUp(this, Window, true);
```

La modalidad se simula creando una ventana transparente grande por debajo del componente `Window`. Debido a la forma en la que se representan las ventanas transparentes, es posible que advierta una ligera atenuación de los objetos situados debajo. Para definir una transparencia efectiva, cambie el valor de `_global.style.modalTransparency` de 0 (completamente transparente) a 100 (opaco). Si hace que la ventana sea parcialmente transparente, también podrá definir el color de la ventana cambiando el aspecto `Modal` del tema predeterminado.

Si utiliza `PopUpManager.createPopUp()` para crear una ventana modal, deberá llamar a `Window.deletePopUp()` para eliminarlo y eliminar también la ventana transparente. Por ejemplo, si utiliza `closeButton` en la ventana, deberá escribir el código siguiente:

```
obj.click = function(evt){
    this.deletePopUp();
}
window.addEventListener("click", obj);
```

Nota: la ejecución del código no se interrumpe cuando se crea una ventana modal. En otros entornos, como Microsoft Windows, si crea una ventana modal, las líneas de código siguientes a la creación de la ventana no se ejecutan hasta que ésta se cierra. En Flash, las líneas de código siguen ejecutándose después de crear la ventana y antes de que se cierre.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.containers.Window.version);
```

Nota: el código siguiente devuelve `undefined`: `trace(myWindowInstance.version);`.

Resumen de métodos de la clase Window

Método	Descripción
<code>Window.deletePopUp()</code>	Elimina una instancia de Window creada por <code>PopUpManager.createPopUp()</code> .

Hereda todos los métodos de [Clase UIObject](#), [Clase UIComponent](#) y [View](#).

Resumen de propiedades de la clase Window

Propiedad	Descripción
<code>Window.closeButton</code>	Indica si la barra de título incluye un botón de cierre (<code>true</code>) o no (<code>false</code>).
<code>Window.content</code>	Referencia al contenido especificado en la propiedad <code>contentPath</code> .
<code>Window.contentPath</code>	Ruta de acceso al contenido que se muestra en la ventana.
<code>Window.title</code>	Texto que aparece en la barra de título.
<code>Window.titleStyleDeclaration</code>	Declaración de estilos que asigna formato al texto de la barra de título.

Hereda todas las propiedades de [Clase UIObject](#), [Clase UIComponent](#) y [ScrollView](#).

Resumen de eventos de la clase Window

Evento	Descripción
<code>Window.click</code>	Se activa al soltar el botón de cierre.
<code>Window.mouseDownOutside</code>	Se activa cuando se presiona el ratón fuera de la ventana modal.

Hereda todos los eventos de [Clase UIObject](#), [Clase UIComponent](#), [View](#) y [ScrollView](#).

Window.click

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(click){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.click = function(eventObject){  
    ...  
}  
windowInstance.addEventListener("click", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se hace clic con el ratón (se suelta) sobre el botón de cierre.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente `Window`. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado a la instancia `myWindow` del componente `Window`, envía “_level0.myWindow” al panel Salida:

```
on(click){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (`windowInstance`) distribuye un evento (en este caso `click`) y una función asociada al objeto detector creado (`listenerObject`) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (`eventObject`) al método del objeto detector. El objeto `Event` tiene un conjunto de propiedades que contiene información acerca del mismo. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en [la página 236](#).

Ejemplo

En el ejemplo siguiente se crea una ventana modal y se define un controlador de acciones del ratón que elimina la ventana. Se debe añadir un componente `Window` al escenario y, a continuación, eliminarlo para añadir el componente a la biblioteca del documento y añadir el código siguiente al fotograma 1:

```
import mx.managers.PopUpManager  
import mx.containers.Window  
var myTW = PopUpManager.createPopUp(_root, Window, true, {closeButton: true,  
    title:"My Window"});
```

```

windowListener = new Object();
windowListener.click = function(evt){
    _root.myTW.deletePopUp();
}
myTW.addEventListener("click", windowListener);

```

Véase también

[UIEventDispatcher.addEventListener\(\)](#), [Window.closeButton](#)

Window.closeButton

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

windowInstance.closeButton

Descripción

Propiedad; valor booleano que indica si la barra de título debe llevar un botón de cierre (`true`) o no (`false`). La propiedad debe definirse en el parámetro *initObject* del método [PopUpManager.createPopUp\(\)](#). El valor predeterminado es `false`.

Ejemplo

El código siguiente crea una ventana en la que se muestra el contenido del clip de película “LoginForm” y que tiene un botón de cierre en la barra de título:

```

var myTW = PopUpManager.createPopUp(_root, Window, true,
    {contentPath:"LoginForm", closeButton:true});

```

Véase también

[Window.click](#), [PopUpManager.createPopUp\(\)](#)

Window.content

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

windowInstance.content

Descripción

Propiedad; referencia al contenido (clip de película raíz) de la ventana. Esta propiedad devuelve un objeto `MovieClip`. Cuando se asocia un símbolo de la biblioteca, el valor predeterminado es una instancia del símbolo asociado. Si se carga el contenido de una URL, el valor predeterminado está sin definir hasta que comienza la operación de carga.

Ejemplo

Define el valor de la propiedad de texto desde el interior del contenido del componente Window:

```
loginForm.content.password.text = "secret";
```

Window.contentPath

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
windowInstance.contentPath
```

Descripción

Propiedad; define el nombre del contenido que ha de aparecer en la ventana. Este valor puede ser el identificador de vinculación de un clip de película de la biblioteca o la URL absoluta o relativa de un archivo SWF o JPG que se carga. El valor predeterminado es "" (cadena vacía).

Ejemplo

El código siguiente crea una instancia de Window que muestra el clip de película con el identificador de vinculación "LoginForm":

```
var myTW = PopUpManager.createPopUp(_root, Window, true,  
    {contentPath:"LoginForm"});
```

Window.deletePopUp()

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

```
windowInstance.deletePopUp();
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; elimina la instancia de Window y quita el estado modal. Este método sólo se puede llamar para instancias de Window creadas por medio de [PopUpManager.createPopUp\(\)](#).

Ejemplo

El código siguiente crea una ventana modal y luego genera un detector que elimina la ventana cuando se hace clic en el botón de cierre:

```
var myTW = PopUpManager.createPopUp(_root, Window, true);
twListener = new Object();
twListener.click = function(){
    myTW.deletePopUp();
}
myTW.addEventListener("click", twListener);
```

Window.mouseDownOutside

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

Sintaxis 1:

```
on(mouseDownOutside){
    ...
}
```

Sintaxis 2:

```
listenerObject = new Object();
listenerObject.mouseDownOutside = function(eventObject){
    ...
}
windowInstance.addEventListener("mouseDownOutside", listenerObject)
```

Descripción

Evento; se difunde a todos los detectores registrados cuando se hace clic con el ratón (se suelta) fuera de la ventana modal. Este evento se utiliza muy raramente, pero puede servir para descartar una ventana si el usuario trata de interactuar con algún elemento situado fuera de la misma.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente a una instancia del componente `Window`. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado a la instancia `myWindowComponent` del componente `Window`, envía “_level0.myWindowComponent” al panel Salida:

```
on(click){
    trace(this);
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*windowInstance*) distribuye un evento (en este caso `mouseDownOutside`) y una función asociada al objeto detector creado (*listenerObject*) lo gestiona. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto Event (*eventObject*) al método del objeto detector. El objeto Event tiene un conjunto de propiedades que contiene información acerca del mismo. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en [la página 236](#).

Ejemplo

En el ejemplo siguiente se crea una instancia de Window y se define un controlador `mouseDownOutside` que llama al método `beep()` cuando el usuario hace clic fuera de la ventana:

```
var myTW = PopUpManager.createPopUp(_root, Window, true, undefined, true);
// crear un detector
twListener = new Object();
twListener.mouseDownOutside = function()
{
    beep(); // emitir un sonido si el usuario hace clic fuera
}
myTW.addEventListener("mouseDownOutside", twListener);
```

Véase también

[UIEventDispatcher.addEventListener\(\)](#)

Window.title

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

windowInstance.title

Descripción

Propiedad; cadena que indica el texto de la barra de título. El valor predeterminado es "" (cadena vacía).

Ejemplo

El código siguiente establece el título de la ventana en “Hello World”:

```
myTW.title = "Hello World";
```

Window.titleStyleDeclaration

Disponibilidad

Flash Player 6.0.79.

Edición

Flash MX 2004.

Sintaxis

windowInstance.titleStyleDeclaration

Descripción

Propiedad; cadena que indica la declaración de estilos que asigna formato a la barra de título de una ventana. El valor predeterminado es indefinido, que significa texto blanco, negrita.

Ejemplo

El código siguiente crea una ventana que muestra el contenido del clip de película con el identificador de vinculación “ChangePassword” y utiliza la titleStyleDeclaration “MyTWStyles”:

```
var myTW = PopUpManager.createPopUp(_root, Window, true,  
    {contentPath:"LoginForm",  
      titleStyleDeclaration:"MyTWStyles"});
```

Para más información sobre estilos, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 27](#).

Componente XMLConnector

Para obtener información reciente sobre esta función, haga clic en el botón Actualizar en la parte superior de la ficha Ayuda.

Componente XUpdateResolver

Para obtener información reciente sobre esta función, haga clic en el botón Actualizar en la parte superior de la ficha Ayuda.

CAPÍTULO 5

Creación de componentes

En este capítulo se describe cómo puede crear sus propios componentes, permitir que los utilicen otros desarrolladores y empaquetarlos para su despliegue.

Novedades

La versión actual (versión 2) de la arquitectura de componentes de Macromedia es muy diferente de la versión Macromedia Flash MX (versión 1). Macromedia ha realizado modificaciones para mejorar la escalabilidad y la extensibilidad de los componentes para los desarrolladores. La lista siguiente proporciona información general sobre algunas de las modificaciones:

- Panel Inspector de componentes que reconoce metadatos de ActionScript
- Administradores y clases básicas ampliables
- Vista previa dinámica incorporada
- Mensajes de compilador mejorados
- Nuevo modelo de eventos
- Gestión de la selección
- Estilos basados en CSS

Trabajo con el entorno de Flash

El entorno de Macromedia Flash MX 2004 y Flash MX Professional 2004 está configurado para crear una estructura lógica de clases y componentes. En esta sección se describe dónde debe almacenar los archivos de componentes.

Activos de los archivos FLA

Cuando crea un componente, debe empezar con un archivo FLA y añadirle aspectos, gráficos y otros activos. Puede almacenar estos activos en cualquier lugar del archivo FLA, ya que los usuarios de los componentes de Flash sólo necesitan un archivo de componente recopilado y no los activos de origen.

Cuando cree componentes de Flash, debe utilizar archivos SWF de dos capas y dos fotogramas. La primera capa es una capa de acciones, que apunta al archivo de clase ActionScript del componente. La segunda es la capa de los activos, que contiene gráficos, símbolos y otros activos que utiliza el componente.

Archivos de clase

El archivo FLA incluye una referencia al archivo de clase ActionScript para el componente. Esto se conoce como vinculación del componente al archivo de clase.

El código ActionScript especifica las propiedades y los métodos del componente y define qué clases (si las hay) hereda el componente. Debe utilizar la convención de asignación de nombres de archivo *.as para el código fuente ActionScript y asignar al archivo de código fuente el nombre del propio componente. Por ejemplo, MyComponent.as contiene el código fuente del componente MyComponent.

Los archivos .as de clase principal de Flash MX 2004 residen en una única carpeta denominada Clases/mx/Core. Otros archivos de clase ActionScript se organizan por su nombre de paquete en sus propias carpetas en /Classes.

Para un componente personalizado, cree un nuevo directorio en /Classes y almacene allí el archivo de clase ActionScript.

La ruta de clases

En esta sección se describe la ruta de clases de Flash.

Aspectos básicos de la ruta de clases

La ruta de clases es una lista ordenada de directorios en la que Flash busca archivos de clase durante la exportación de componentes o la generación de archivos SWF. El orden de las entradas de la ruta de clases es importante porque Flash utiliza las clases por orden de llegada. En el momento de la exportación, las clases que se encuentran en la ruta de clases y que coinciden con los identificadores de vínculo del archivo FLA se importan al archivo FLA y se registran con sus símbolos.

Una ruta de clase global hace referencia a todos los archivos FLA que se generan mediante Flash. Una ruta de clase local sólo se aplica al archivo FLA actual.

La ruta de clases local predeterminada está vacía. La ruta de clases global predeterminada consta de las dos rutas siguientes:

- \$(UserConfig)/Classes
- .

El punto (.) indica el directorio de trabajo actual. Flash busca clases de ActionScript en el directorio actual del archivo FLA.

La ruta \$(UserConfig)/Classes indica el directorio de configuración del usuario. Este directorio apunta a las ubicaciones siguientes:

- En Windows, este directorio es C:\Archivos de programa\Macromedia\Flex MX 2004\es\First Run.
- En Macintosh, este directorio es *volume:Users:nombre de usuario:Library:Application Support:Macromedia:Flash MX 2004:es:configuration*.

Los directorios UserConfig reflejan los directorios ubicados en *Flash_root/es/First Run*. Sin embargo, la ruta de clases no incluye estos directorios directamente, y es relativa al directorio UserConfig.

Modificación de la ruta de clases

Puede cambiar la ruta de clases de un archivo FLA independiente (ruta de clases local) o la de todos los archivos FLA con los que trabaja en Flash (ruta de clases global).

Para cambiar la ruta de clases global:

- 1 Seleccione Edición > Preferencias.
Aparecerá el cuadro de diálogo Preferencias.
- 2 Seleccione la ficha ActionScript.
- 3 Haga clic en el botón Configuración de ActionScript 2.0.
Aparecerá el cuadro de diálogo Configuración de ActionScript.
- 4 Añada, elimine o edite entradas en el cuadro Ruta de clases.
- 5 Guarde los cambios.

Para cambiar la ruta de clases local:

- 1 Seleccione Archivo > Configuración de publicación.
Aparecerá el cuadro de diálogo Configuración de publicación.
- 2 Seleccione la ficha Flash.
- 3 Pulse el botón Configuración.
Aparecerá el cuadro de diálogo Configuración de ActionScript.
- 4 Añada, elimine o edite entradas en el cuadro Ruta de clases.
- 5 Guarde los cambios.

Búsqueda de los archivos de origen de los componentes

Cuando desarrolle un componente, puede almacenar los archivos de origen en cualquier directorio. Sin embargo, debe incluir ese directorio en la configuración de la ruta de clases de Flash MX 2004 para asegurarse de que Flash encuentre los archivos de clase necesarios cuando exporte el componente. Asimismo, para probar el componente, debe almacenarlo en el directorio Flash Components. Para más información sobre el almacenamiento de archivos SWC, consulte [“Utilización de archivos SWC” en la página 291](#).

Edición de símbolos

Cada símbolo tiene su propia línea de tiempo. Así como puede añadir fotogramas, fotogramas clave y capas a la línea de tiempo principal, también puede efectuar la misma operación en la línea de tiempo de un símbolo.

Cuando crea componentes, siempre se inicia con un símbolo. Flash proporciona los tres métodos siguientes para editar símbolos:

- Editar el símbolo en el contexto de los otros objetos del escenario mediante el comando Editar en contexto. Otros objetos aparecen atenuados para distinguirlos del símbolo que se está editando. El nombre del símbolo que se está editando se muestra en una barra de edición situada en la parte superior del escenario, a la derecha del nombre de la escena actual.

- Edite un símbolo en una ventana distinta mediante el comando Editar en nueva ventana. La edición de un símbolo en una ventana independiente le permite ver a la vez el símbolo y la línea de tiempo principal. El nombre del símbolo que se está editando se muestra en la barra de edición situada en la parte superior del escenario.
- El símbolo se edita cambiando la ventana de la vista del escenario a una vista de sólo el símbolo, con el modo de edición de símbolos. El nombre del símbolo que se está editando se muestra en la barra de edición situada en la parte superior del escenario, a la derecha del nombre de la escena actual.

Ejemplos de código de componentes

Flash MX 2004 y Flash MX Professional 2004 incluyen los siguientes archivos de origen de componentes para ayudarle a desarrollar sus propios componentes:

- Código fuente del archivo FLA: *Flash MX 2004_install_dir/es/First Run/ComponentFLA/StandardComponents fla*
- Archivos de clase ActionScript: *Flash MX 2004_install_dir/es/First Run/Classes/mx*

Creación de componentes

En esta sección se describe cómo crear un componente que será una subclase de una clase existente de Flash MX 2004. En las secciones posteriores se describe cómo escribir el archivo de clase ActionScript del componente y cómo editar dicho componente para mejorar su calidad y su facilidad de uso.

Creación de un símbolo de un nuevo componente

Todos los componentes son objetos MovieClip, que son un tipo de símbolo. Para crear un nuevo componente, primero debe insertar un nuevo símbolo en un nuevo archivo FLA.

Para añadir el símbolo de un nuevo componente:

- 1 En Flash, cree un documento Flash vacío.
- 2 Seleccione Insertar > Nuevo símbolo.
Aparecerá el cuadro de diálogo Crear un nuevo símbolo.
- 3 Especifique un nombre de símbolo. Asigne un nombre al componente poniendo en mayúscula la primera letra de cada una de las palabras del componente (por ejemplo, MyComponent).
- 4 Seleccione el botón de opción Clip de película para el comportamiento.
Un objeto MovieClip dispone de su propia línea de tiempo de varios fotogramas, que se reproduce de forma independiente de la línea de tiempo principal.
- 5 Haga clic en el botón Avanzado.
El cuadro de diálogo mostrará la configuración avanzada.
- 6 Seleccione Exportar para ActionScript. Esto hace que Flash empaquete el componente de forma predeterminada con cualquier contenido de Flash que utilice el componente.
- 7 Introduzca un identificador de vínculo.
El identificador se utiliza como nombre de símbolo, nombre de vínculo y nombre de clase asociado.
- 8 En el cuadro de texto Clase de AS 2.0, introduzca la ruta completa en la clase ActionScript 2.0, relativa a la configuración de su ruta de acceso.

Nota: no incluya la extensión del nombre de archivo; el cuadro de texto Clase de AS 2.0 apunta a la ubicación del paquete de la clase y no al nombre que ha otorgado al archivo el sistema de archivos.

Si el archivo ActionScript se encuentra en un paquete, debe incluir el nombre del paquete. El valor de este campo puede ser relativo a la ruta de clases o también una ruta de paquetes absoluta (por ejemplo, myPackage.MyComponent).

Para más información sobre la configuración de la ruta de clases de Flash MX 2004 consulte [“Aspectos básicos de la ruta de clases” en la página 266](#).

- 9 En la mayoría de los casos, debe anular la selección de Exportar en primer fotograma (está seleccionada de forma predeterminada). Para más información, consulte [“Recomendaciones para el diseño de un componente” en la página 292](#).

- 10 Haga clic en Aceptar.

Flash añade los símbolos a la biblioteca y cambia al modo de edición de símbolos. En este modo, el nombre del símbolo aparece encima de la esquina superior izquierda del escenario y una cruz filar indica el punto de registro del símbolo.

Ahora puede editar este símbolo y añadirlo al archivo FLA de su componente.

Edición de capas de símbolos

Una vez haya creado el nuevo símbolo y definido sus vínculos, puede definir los activos del componente en la línea de tiempo del símbolo.

El símbolo de un componente debe tener dos capas. En esta sección se describe qué capas deben insertarse y qué debe añadirse a estas capas.

Para obtener más información sobre cómo editar los símbolos, consulte [“Edición de símbolos” en la página 267](#).

Para editar las capas de los símbolos:

- 1 Introduzca el modo de edición de símbolos.
- 2 Asigne un nuevo nombre a una capa vacía o cree una capa llamada Acciones.
- 3 En el panel Acciones, añada una línea que importe el archivo de clase ActionScript completo del componente.

Esta sentencia se basa en la configuración de la ruta de clases de Flash MX 2004. Para más información, consulte [“Aspectos básicos de la ruta de clases” en la página 266](#). El ejemplo siguiente importa el archivo MyComponent.as que se encuentra en el paquete myPackage:

```
import myPackage.MyComponent;
```

Nota: utilice la sentencia `import` y no la sentencia `include` cuando importe un archivo de clase ActionScript. No coloque comillas en el nombre de clase o en el paquete.

- 4 Asigne un nuevo nombre a una capa vacía o cree una capa llamada Activos.
La capa Activos incluye todos los activos que utiliza este componente.
- 5 En el primer fotograma, añada una acción `stop()` en el panel Acciones, como se muestra en el ejemplo siguiente:

```
stop();
```

No añada ningún activo gráfico a este fotograma. Flash Player se detendrá antes del segundo fotograma, en el que podrá añadir los activos.

- 6 Si está ampliando un componente existente, ubique este componente y todas las demás clases base que utilice y coloque una instancia de ese símbolo en el segundo fotograma de su capa. Para hacerlo, seleccione el símbolo en el panel Componentes y arrástrelo hasta el escenario en el segundo fotograma de la capa Activos de su componente.
Todos los activos que el componente utilice (ya sea otro componente o medios como mapas de bits) deben disponer de una instancia colocada dentro del componente.
- 7 Añada los activos gráficos que utilice este componente en el segundo fotograma de la capa Activos de su componente. Por ejemplo, si está creando un botón personalizado, añada los gráficos que representan los estados del botón (arriba, presionado, etc.).
- 8 Cuando haya terminado de crear el contenido del símbolo, siga uno de estos procedimientos para volver al modo de edición de documentos:
 - Haga clic en el botón Atrás situado en la parte izquierda de la barra de edición, encima del escenario.
 - Seleccione Edición > Editar documento.
 - Haga clic en el nombre de escena, en la barra de información, encima del escenario.

Adición de parámetros

El siguiente paso en el desarrollo de componentes es definir los parámetros del componente. Los parámetros son el método principal mediante el que los usuarios modifican las instancias de los componentes que ha creado.

En ediciones anteriores de Flash, solían definirse los parámetros mediante el panel Inspector de componentes. En Flash MX 2004 y Flash MX Professional 2004, se definen los parámetros en el archivo de clase ActionScript y el panel Inspector de componentes identifica los que son públicos para mostrarlos a los usuarios.

La sección siguiente trata la escritura del archivo ActionScript externo del componente, que incluye información sobre la adición de parámetros de componentes.

Escritura de ActionScript del componente

La mayoría de los componentes incluyen algún tipo de código ActionScript. El tipo de componente determina la ubicación en la que escribirá el ActionScript y la cantidad que debe escribirse. Existen dos enfoques básicos para desarrollar componentes:

- Crear nuevos componentes sin clases principales
- Ampliar las clases de los componentes existentes

Esta sección se centra en la ampliación de los componentes existentes. Si está creando un componente que se basa en el archivo de clase de otro componente, debe escribir un archivo de clase ActionScript externo como se describe en esta sección.

Ampliación de las clases de los componentes existentes

Cuando se crea el símbolo de un componente que procede de una clase principal, se vincula a un archivo de clase ActionScript 2.0 externo. Para más información sobre la definición de este archivo, consulte [“Creación de un símbolo de un nuevo componente” en la página 268](#).

La clase `ActionScript` externa amplía otra clase, añade métodos, captadores y definidores y define controladores de eventos para el componente. Para editar los archivos de clase `ActionScript`, puede utilizar Flash, cualquier editor de texto o cualquier IDE (entorno de desarrollo integrado, Integrated Development Environment).

Sólo puede heredar de una clase. `ActionScript 2.0` no permite la herencia múltiple.

Ejemplo sencillo de un archivo de clase

A continuación se muestra un ejemplo sencillo de un archivo de clase denominado `MyComponent.as`. Este ejemplo contiene un conjunto mínimo de importaciones, métodos y declaraciones para un componente que hereda de la clase `UIObject`.

```
import mx.core.UIObject;

class myPackage.MyComponent extends UIObject {
    static var symbolName:String = "MyComponent";
    static var symbolOwner:Object = Object(myPackage.MyComponent);
    var className:String = "MyComponent";
    #include "../core/ComponentVersion.as"
    function MyComponent() {
    }
    function init(Void):Void {
        super.init();
    }
    function size(Void):Void {
        super.size();
    }
}
```

Proceso general para escribir un archivo de clase

Utilice el siguiente proceso general cuando escriba el `ActionScript` de un componente. Algunos de los pasos pueden ser opcionales, en función del tipo de componente que haya creado.

Este proceso se analiza más detalladamente en el resto de este capítulo.

Para escribir el archivo `ActionScript` de un componente:

- 1 Importe todas las clases necesarias.
- 2 Defina la clase mediante la palabra clave `class`; si es necesario, amplíe una clase principal.
- 3 Defina las variables `symbolName` y `symbolOwner`; éstos son, respectivamente, el nombre del símbolo de la clase `ActionScript` y el nombre completo del paquete de la clase.
- 4 Defina el nombre de la clase como la variable `className`.
- 5 Añada la información sobre el control de versiones.
- 6 Introduzca las variables de miembro predeterminadas.
- 7 Cree variables para los elementos y los vínculos de cada aspecto que se utilizan en el componente. Esto permite a los usuarios configurar un elemento de aspecto distinto mediante la modificación de un parámetro del componente.
- 8 Añada las constantes de las clases.
- 9 Añada una palabra clave y una declaración de metadatos para cada variable que disponga de un captador/definidor.
- 10 Defina las variables de miembros no inicializadas.

- 11 Defina los captadores y los definidores.
- 12 Escriba un constructor. Normalmente debería estar vacío.
- 13 Añada un método de inicialización. Este método se llama cuando se crea la clase.
- 14 Añada un método de delimitación del tamaño.
- 15 Añada los métodos personalizados o sustituya los métodos heredados.

Importación de clases

La primera línea del archivo de clase ActionScript externo debe importar los archivos de clase necesarios que utiliza la clase. Esto incluye la clases que proporcionan la funcionalidad, así como la superclase que la clase amplía, si la hay.

Cuando utilice la sentencia `import`, debe importar el nombre de clase completo en lugar del nombre de archivo de la clase, como se muestra en el ejemplo siguiente:

```
import mx.core.UIObject;  
import mx.core.ScrollView;  
import mx.core.ext.UIObjectExtensions;
```

También puede utilizar el carácter comodín (*) para importar todas las clases de un paquete determinado. Por ejemplo, la sentencia siguiente importa todas las clases del paquete `mx.core`:

```
import mx.core.*;
```

Si una clase importada no se utiliza en un script, la clase no se incluirá en el código de bytes del archivo SWF resultante. Como resultado, la importación de un paquete entero con un carácter comodín no crea un archivo SWF innecesariamente grande.

Selección de una clase principal

La mayoría de los componentes tienen algunas funciones y comportamientos comunes. Flash incluye dos clases básicas para proporcionar estas características comunes. Si se convierten estas clases en subclases, los componentes empiezan con un conjunto básico de métodos, propiedades y eventos.

La tabla siguiente describe brevemente las dos clases base:

Clase completa	Amplía	Descripción
mx.core.UIObject	MovieClip	UIObject es la clase base para todos los objetos gráficos. Puede tener forma, dibujarse a sí misma y ser invisible. UIObject proporciona las funciones siguientes: <ul style="list-style-type: none">• Edición de estilos• Gestión de eventos• Cambio del tamaño mediante la escala
mx.core.UIComponent	UIObject	UIComponent es la clase base para todos los componentes. Puede participar en la tabulación, aceptar eventos de bajo nivel como las entradas de teclado y de ratón y estar desactivada para no tener que recibir entradas de teclado y de ratón. UIComponent proporciona las funciones siguientes: <ul style="list-style-type: none">• Creación de desplazamientos de la selección• Activación y desactivación de componentes• Cambio de tamaño de los componentes

Aspectos básicos de la clase UIObject

Los componentes que se basan en la versión 2 de la arquitectura de componentes de Macromedia proceden de la clase UIObject, que ajusta la clase MovieClip. La clase MovieClip es la clase base de todas las clases de Flash que pueden realizar dibujos en la pantalla. Muchas de las propiedades y los métodos de MovieClip están relacionados con la línea de tiempo, que es una herramienta con la que no están familiarizados los desarrolladores que empiezan a trabajar con Flash. La clase UIObject fue creada para extraer muchos de estos detalles. Las subclases de MovieClip no documentan las propiedades y los métodos de MovieClip innecesarios. Sin embargo, puede acceder a estas propiedades y métodos si lo desea.

UIObject intenta ocultar la gestión del ratón y de los fotogramas en MovieClip. Envía eventos a sus detectores justo antes de dibujar (el equivalente de `onEnterFrame`), cuando se carga y se descarga y cuando su diseño cambia (`move`, `resize`).

UIObject proporciona variables de sólo lectura alternativas para determinar la posición y el tamaño del clip de película. Puede utilizar los métodos `move()` y `setSize()` para modificar la posición y el tamaño de un objeto.

Aspectos básicos de la clase UIComponent

UIComponent es una clase secundaria de UIObject. Es la clase base de todos los componentes que interactúan con el usuario (entradas de teclado y de ratón).

Ampliación de otras clases

Para facilitar la construcción de componentes, puede crear una subclase para cualquier clase; no es obligatorio ampliar las clases UIObject o UIComponent directamente. Si amplía la clase de cualquier otro componente, ampliará estas clases de forma predeterminada. Cualquier clase de componente que aparezca en el diccionario de componentes puede ampliarse para crear una nueva clase de componente.

Flash incluye un grupo de clases que dibujan en la pantalla y se heredan desde `UIObject`. Por ejemplo, la clase `Border` dibuja bordes alrededor de otros objetos. Otro ejemplo es `RectBorder`, que es una subclase de `Border` y sabe cómo modificar correctamente el tamaño de sus elementos visuales. Todos los componentes que admiten bordes deben utilizar una de las clases o subclases de borde. Para obtener una descripción detallada de estas clases, consulte el [Capítulo 4, “Diccionario de componentes”](#), en la página 45.

Por ejemplo, si desea crear un componente que se comporte de forma casi exacta que el componente `Button`, puede ampliar la clase `Button` en lugar de volver a crear todas las funciones de la clase `Button` desde las clases base.

Escritura del constructor

Los constructores son métodos que tienen una única función: configurar propiedades y llevar a cabo otras tareas cuando se crea una instancia nueva para un componente. Puede reconocer un constructor porque tiene el mismo nombre que la clase del componente. Por ejemplo, el código siguiente muestra el constructor del componente `ScrollBar`:

```
function ScrollBar() {  
}
```

En este caso, cuando se crea una nueva instancia para el componente `ScrollBar`, se llama al constructor `ScrollBar()`.

Normalmente, los constructores de componentes deben estar vacíos, de modo que el objeto pueda personalizarse con la interfaz de propiedades. Asimismo, la configuración de propiedades en los constructores puede conducir a menudo a la sobrescritura de los valores predeterminados, en función del orden de las llamadas de inicialización.

Una clase sólo puede contener una función constructora; `ActionScript 2.0` no admite funciones constructoras sobrecargadas.

Control de versiones

Cuando cree nuevos componentes, debe definir un número de versión. Esto permite a los desarrolladores saber si deben actualizarlo y les ayuda en caso de consultas al servicio de asistencia técnica. Cuando establezca el número de versión de un componente, utilice la variable estática `version`, como se muestra en el ejemplo siguiente:

```
static var version:String = "1.0.0.42";
```

Si crea muchos componentes como parte de un paquete de componentes, puede incluir el número de versión en un archivo externo. Así, actualiza el número de versión en una sola ubicación. Por ejemplo, el código siguiente importa el contenido de un archivo externo que almacena el número de versión en una sola ubicación:

```
#include "../myPackage/ComponentVersion.as"
```

El contenido del archivo `ComponentVersion.as` es idéntico a la declaración de variables anterior, como muestra el siguiente ejemplo:

```
static var version:String = "1.0.0.42";
```

Nombres de clase, símbolo y propietario

Para ayudar a Flash a encontrar las clases y los paquetes ActionScript y para conservar el nombre del componente, debe establecer las propiedades `symbolName`, `symbolOwner` y `className` en el archivo de clase ActionScript del componente.

La tabla siguiente describe estas variables:

Variable	Descripción
<code>symbolName</code>	Nombre del símbolo para el objeto. Esta variable es estática y de tipo <code>String</code> .
<code>symbolOwner</code>	Clase que se utiliza en la llamada interna al método <code>createClassObject()</code> . Este valor debería ser el nombre de clase completo, que incluye la ruta del paquete. Esta variable es estática y de tipo <code>Object</code> .
<code>className</code>	Nombre de la clase del componente. Esta variable también se utiliza para calcular los valores de estilo. Si <code>_global.styles[className]</code> existe, establece valores predeterminados para un componente. Esta variable es de tipo <code>String</code> .

El ejemplo siguiente muestra la denominación de un componente personalizado:

```
static var symbolName:String = "MyComponent";
static var symbolOwner:Object = custom.MyComponent;
var className:String = "MyComponent";
```

Definición de captadores y definidores

Los captadores y los definidores permiten visualizar las propiedades de los componentes y controlan el acceso de otros objetos a estas propiedades.

La convención para definir métodos de captadores y definidores es colocar antes del nombre del método `get` o `set`, seguido de un espacio y el nombre de la propiedad. Es recomendable utilizar letras en mayúsculas al inicio de las palabras que vayan a continuación de `get` o `set`.

La variable que almacena el valor de la propiedad no puede tener el mismo nombre que el captador o el definidor. Normalmente, antes del nombre de las variables captador o definidor se sitúan dos guiones bajos.

El ejemplo siguiente muestra la declaración de `initialColor`, así como los métodos de captador o definidor que obtienen y establecen el valor de esta propiedad:

```
...
public var __initialColor:Color = 42;
...
public function get initialColor():Number {
    return __initialColor;
}
public function set initialColor(newColor:Number) {
    __initialColor = newColor;
}
```

Los captadores y los definidores suelen utilizarse junto con palabras claves de metadatos para definir las propiedades que son visibles, que se pueden vincular y que cuentan con otras propiedades.

Metadatos de componente

Flash identifica las sentencias de metadatos de componente en los archivos de clase ActionScript externos. Las etiquetas de metadatos pueden definir los atributos de los componentes, las propiedades de las vinculaciones de datos y los eventos. Flash interpreta estas sentencias y actualiza el entorno de desarrollo en consecuencia. Esto permite definir estos miembros una sola vez, en lugar de en el código ActionScript y en los paneles de desarrollo.

Las etiquetas de metadatos sólo pueden utilizarse en los archivos de clase ActionScript externos. No puede utilizar etiquetas de metadatos en los fotogramas de acción de los archivos FLA.

Utilización de palabras clave de metadatos

Los metadatos se asocian con una declaración de clases o con un único campo de datos. Si el valor de un atributo es del tipo String, debe ponerlo entre comillas.

Las sentencias de metadatos están vinculadas a la siguiente línea del archivo ActionScript. Cuando defina la propiedad de un componente, añada la etiqueta de metadatos en la línea antes de la declaración de la propiedad. Cuando defina eventos de los componentes, añada la etiqueta de metadatos fuera de la definición de la clase, de modo que el evento se vincule a toda la clase.

En el ejemplo siguiente, las palabras clave de metadatos Inspectable se aplican a los parámetros flavorStr, colorStr y shapeStr:

```
[Inspectable(defaultValue="strawberry")]
public var flavorStr:String;
[Inspectable(defaultValue="blue")]
public var colorStr:String;
[Inspectable(defaultValue="circular")]
public var shapeStr:String;
```

En el inspector de propiedades y en la ficha Parámetros del panel Inspector de componentes, Flash muestra todos estos parámetros como de tipo String.

Etiquetas de metadatos

La tabla siguiente describe las etiquetas de metadatos que puede utilizar en los archivos de clase ActionScript:

Etiqueta	Descripción
Inspectable	Define un atributo expuesto a los usuarios de componentes en el panel Inspector de componentes. Véase “Inspectable” en la página 277 .
InspectableList	Identifica qué subconjunto de parámetros Inspectable deben enumerarse en el inspector de propiedades. Si no añade un atributo InspectableList a la clase del componente, todos los parámetros Inspectable aparecen en el inspector de propiedades. Véase “InspectableList” en la página 279 .
Event	Define los eventos de los componentes. Véase “Event” en la página 279 .
Bindable	Identifica una propiedad en la ficha Vinculaciones del panel Inspector de componentes. Véase “Bindable” en la página 280 .
ChangeEvent	Identifica los eventos que provocan la vinculación de datos. Véase “ChangeEvent” en la página 281 .

Etiqueta	Descripción
ComponentTask	Apunta a uno o varios archivos JSFL asociados con el componente que realiza tareas para la instancia de un componente determinado. Véase "ComponentTask" en la página 282 .
IconFile	Nombre de archivo del icono que representa este componente en el panel Componentes de Flash. Véase "Adición de un icono" en la página 292 .

Las secciones siguientes describen con mayor detalle las etiquetas de los metadatos de componente.

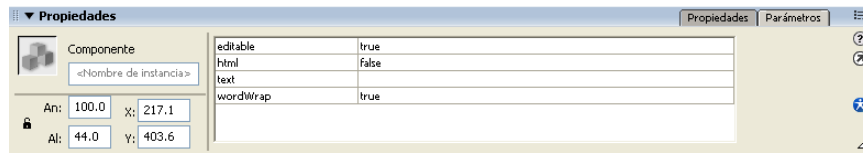
Inspectable

Se especifican los parámetros que puede editar el usuario (o "Inspectable") de un componente en la definición de su clase. Dichos parámetros aparecen en el panel Inspector de componentes. Esto permite mantener las propiedades Inspectable y el código ActionScript subyacente en la misma ubicación. Para visualizar las propiedades de los componentes, arrastre una instancia del componente hasta el escenario y seleccione la ficha Parámetros del panel Inspector de componentes.

La siguiente ilustración muestra la ficha Parámetros del panel Inspector de componentes para el control TextArea:



De forma alternativa, puede visualizar un subconjunto de propiedades del componente en la ficha Parámetros del inspector de propiedades, como muestra la ilustración siguiente:



Cuando se determinan los parámetros que deben mostrarse en el entorno de edición, Flash utiliza la palabra clave de metadatos `Inspectable`. La sintaxis de esta palabra clave es la siguiente:

```
[Inspectable(value_type=value[,attribute=value,...])]  
property_declaration name:type;
```

El ejemplo siguiente define el parámetro `enabled` como `Inspectable`:

```
[Inspectable(defaultValue=true, verbose=1, category="Other")]  
var enabled:Boolean;
```

La palabra clave `Inspectable` también es compatible con atributos sueltos como el siguiente:

```
[Inspectable("danger", 1, true, maybe)]
```

La sentencia de metadatos debe preceder a la declaración de variables de la propiedad que debe vincularse a dicha propiedad.

La tabla siguiente describe los atributos de la palabra clave de metadatos `Inspectable`:

Atributo	Tipo	Descripción
name	Cadena	(Opcional) Un nombre de visualización de la propiedad. Por ejemplo, "Font Width". Si no lo especifica, utilice el nombre de la propiedad, como por ejemplo "_fontWidth".
type	Cadena	(Opcional) Un especificador del tipo. Si se omite, utilice el tipo de la propiedad. Los valores siguientes son aceptables: <ul style="list-style-type: none"> • Matriz • Objeto • Lista • Cadena • Número • Booleano • Nombre de fuente • Color
defaultValue	Cadena o número	(Obligatorio) Valor predeterminado para la propiedad <code>Inspectable</code> .
enumeration	Cadena	(Opcional) Especifica una lista separada por comas de valores legales de la propiedad.
verbose	Número	(Opcional) Indica que esta propiedad <code>Inspectable</code> sólo debe mostrarse cuando el usuario indica que deben incluirse propiedades detalles. Si no se especifica este atributo, Flash presupone que la propiedad debe mostrarse.

Atributo	Tipo	Descripción
category	Cadena	(Opcional) Agrupa la propiedad en una subcategoría en el inspector de propiedades.
listOffset	Número	(Opcional) Se añade para que sea compatible con los componentes de Flash MX. Se utiliza como el índice predeterminado en un Valor de lista.
variable	Cadena	(Opcional) Se añade para que sea compatible con los componentes de Flash MX. Se utiliza para especificar la variable con la que está vinculado este parámetro.

InspectableList

Utilice la palabra clave de metadatos `InspectableList` para especificar exactamente qué subconjunto de parámetros `Inspectable` deben aparecer en el inspector de propiedades. Utilice `InspectableList` junto con `Inspectable`, de modo que pueda ocultar los atributos heredados para los componentes que se han convertido en subclases. Si no añade una palabra clave de metadatos `InspectableList` a la clase del componente, todos los parámetros `Inspectable`, incluidos los de las clases principales del componente, aparecen en el inspector de propiedades.

La sintaxis de `InspectableList` es la siguiente:

```
[InspectableList("attribute1"[...])]
// definición de clase
```

La palabra clave `InspectableList` debe preceder a la definición de la clase, ya que se aplica a toda la clase.

El ejemplo siguiente permite mostrar las propiedades `flavorStr` y `colorStr` en el inspector de propiedades, pero excluye otras propiedades `Inspectable` de la clase `DotParent`:

```
[InspectableList("flavorStr","colorStr")]
class BlackDot extends DotParent {
    [Inspectable(defaultValue="strawberry")]
    public var flavorStr:String;
    [Inspectable(defaultValue="blue")]
    public var colorStr:String;
    ...
}
```

Event

Utilice la palabra clave de metadatos `Event` para definir los eventos que emite este componente.

La sintaxis de esta palabra clave es la siguiente:

```
[Event("event_name")]
```

Por ejemplo, el código siguiente define un evento `click`:

```
[Event("click")]
```

Añada las sentencias de `Event` fuera de la definición de la clase en el archivo `ActionScript`, de modo que se vinculen a la clase y no a un miembro concreto de dicha clase.

Los ejemplos siguientes muestran los metadatos Event para la clase UIObject, que gestiona los eventos `resize`, `move` y `draw`:

```
...
import mx.events.UIEvent;
[Event("resize")]
[Event("move")]
[Event("draw")]
class mx.core.UIObject extends MovieClip {
    ...
}
```

Bindable

La vinculación de datos conecta los componentes entre sí. La vinculación de datos visual se consigue a través de la ficha Vinculaciones del panel Inspector de componentes. Desde esta ficha puede añadir, visualizar y eliminar vinculaciones para un componente.

Aunque la vinculación de datos funciona con cualquier componente, su objetivo principal es conectar los componentes de la interfaz de usuario a los orígenes de datos externos, como los servicios Web o los documentos XML. Estos orígenes de datos están disponibles como componentes con propiedades, que se pueden vincular con las propiedades de otros componentes. El panel Inspector de componentes es la principal herramienta que se utiliza en Flash MX Professional 2004 para realizar la vinculación de datos.

Utilice la palabra clave de metadatos Bindable para que las propiedades y las funciones de captadores/definidores de las clases ActionScript aparezcan en la ficha Vinculaciones del panel Inspector de componentes.

La palabra clave de metadatos Bindable tiene la sintaxis siguiente:

```
[Bindable[readonly|writeonly[,type="datatype"]]]
```

La palabra clave Bindable debe preceder a una propiedad, a una función de captador/definidor o a otra palabra clave de metadatos que preceda a una propiedad o a una función de captador/definidor.

El ejemplo siguiente define la variable `flavorStr` como una variable pública e Inspectable a la que también se puede acceder desde la ficha Vinculaciones del panel Inspector de componentes:

```
[Bindable]
[Inspectable(defaultValue="strawberry")]
public var flavorStr:String = "strawberry";
```

La palabra clave de metadatos Bindable utiliza tres opciones que especifican el tipo de acceso a la propiedad y el tipo de datos de dicha propiedad. La tabla siguiente describe estas opciones:

Opción	Descripción
<code>readonly</code>	Indica a Flash que permita que la propiedad sea sólo el origen de una vinculación, como se muestra en el ejemplo: <code>[Bindable("readonly")]</code>

Opción	Descripción
<code>readonly</code>	Indica a Flash que permita que la propiedad sea sólo el destino de una vinculación, como se muestra en el ejemplo: [Bindable("readonly")]
<code>type="datatype"</code>	Especifica el tipo de datos de la propiedad que se está vinculando. Si no especifica esta opción, la vinculación de datos utiliza el tipo de datos de la propiedad como se declara en el código ActionScript. Si <i>datatype</i> es un tipo de datos registrado, puede utilizar la función del menú emergente Tipo de datos de la ficha Esquema. El ejemplo siguiente establece el tipo de datos de la propiedad en String: [Bindable(type="String")]

Puede combinar la opción de acceso y la opción de tipo de datos, como se muestra en el ejemplo siguiente:

```
[Bindable(param1="readonly", type="DataProvider")]
```

La palabra clave Bindable es obligatoria cuando utilice la palabra clave de metadatos ChangeEvent. Para más información, consulte [“ChangeEvent” en la página 281](#).

Para obtener información sobre la creación de vinculaciones de datos en el entorno de edición de Flash, consulte “Vinculación de datos (sólo en Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

ChangeEvent

Utilice la palabra clave de metadatos ChangeEvent para generar uno o varios eventos de componentes cuando se efectúen cambios en las propiedades vinculables.

La sintaxis de esta palabra clave es la siguiente:

```
[Bindable]
[ChangeEvent("event"[,...])]
property_declaration or get/set function
```

Como Bindable, esta palabra clave sólo puede utilizarse con declaraciones de variables o con funciones de captadores o definidores.

En el ejemplo siguiente, el componente genera el evento `change` cuando el valor de la propiedad vinculable `flavorStr` se modifica:

```
[Bindable]
[ChangeEvent("change")]
public var flavorStr:String;
```

Cuando tiene lugar el evento especificado en los metadatos, Flash informa al elemento que está vinculado a la propiedad de que la propiedad se ha modificado.

También puede indicar al componente que genere un evento cuando se llame a una función de captador o definidor, como se muestra en el ejemplo siguiente:

```
[Bindable]
[ChangeEvent("change")]
function get selectedDate():Date
...

```

En la mayoría de los casos, el evento `change` se establece en el captador y se distribuye en el definidor.

Puede registrar varios eventos `change` en los metadatos, como se muestra en el ejemplo siguiente:

```
[ChangeEvent("change1", "change2", "change3")]
```

Cualquiera de estos eventos indica un cambio en la variable. No es necesario que tengan lugar todos para indicar un cambio.

ComponentTask

Un componente puede contar con uno o varios archivos JSFL asociados que realizan tareas de utilidad para la instancia de un componente determinado. Puede definir estas tareas para el componente y declararlas mediante la palabra clave de metadatos `ComponentTask`.

La palabra clave de metadatos `ComponentTask` utiliza la sintaxis siguiente:

```
[ComponentTask(taskName,taskFile,otherFile1[...])]
```

La tabla siguiente describe los atributos de la palabra clave `ComponentTask`:

Atributo	Tipo	Descripción
taskName	Cadena	(Obligatorio) Nombre de la tarea, mostrado como una cadena.
taskFile	Cadena	(Obligatorio) Nombre del archivo JSFL que implementa la tarea.
otherFile1, ...	Cadena	(Obligatorio) Uno o varios nombres de archivos que necesita el archivo JSFL; por ejemplo, la tarea puede requerir archivos de descripción XML2UI.

Añada las sentencias `ComponentTask` fuera de la definición de la clase en el archivo `ActionScript`, de modo que se vinculen a la clase y no a un miembro concreto de dicha clase.

En el ejemplo siguiente se definen dos sentencias `ComponentTask` para la clase `MyComponent`:

```
[ComponentTask("Do Some  
Setup","MyComponentSetup.jsfl","myForm.xml","myOtherForm.xml")]  
[ComponentTask("Do Some More Setup","MyOtherComponentSetup.jsfl")]  
class myComponent {  
    ...  
}
```

El menú emergente de tareas aparece en la ficha Esquema del panel Inspector de componentes. Para activar este menú, seleccione una instancia de componente en el escenario y haga clic en el botón situado a la derecha del panel. El menú contiene los nombres de todas las tareas definidas en los metadatos del componente.

Cuando selecciona el nombre de una tarea en el menú de tareas, se invoca el archivo JSFL correspondiente. Desde el interior del archivo JSFL, puede acceder al componente seleccionado del modo siguiente:

```
var aComponent = fl.getDocumentDOM().selection[0];
```

Flash incluye los archivos JSFL asociados con el componente cuando se exporta un archivo SWC. El archivo JSFL y los archivos auxiliares deben estar en la misma carpeta que el archivo FLA cuando exporte el componente como archivo SWC. Para más información sobre la generación de un archivo SWC, consulte [“Exportación del componente” en la página 290](#).

Definición de parámetros de componentes

Cuando crea un componente, puede añadir parámetros que definan su aspecto y su comportamiento. Las propiedades utilizadas más habitualmente aparecen como parámetros de edición en el panel Inspector de componentes. Estas propiedades se definen mediante la palabra clave `Inspectable` (consulte [“Inspectable” en la página 277](#)). También puede establecer todos los parámetros `Inspectable` con `ActionScript`. Los valores de los parámetros establecidos mediante `ActionScript` sustituyen cualquier otro valor que se haya establecido durante la edición.

El ejemplo siguiente establece varios parámetros de componentes en el archivo de clase `JellyBean` y los expone con la palabra clave de metadatos `Inspectable` en el panel Inspector de componentes:

```
class JellyBean{
    // un parámetro de cadena
    [Inspectable(defaultValue="strawberry")]
    public var flavorStr:String;
    // un parámetro de lista de cadenas
    [Inspectable(enumeration="sour,sweet,juicy,rotten",defaultValue="sweet")]
    public var flavorType:String;
    // un parámetro de matriz
    [Inspectable(name="Flavors", defaultValue="strawberry,grape,orange",
    verbose=1, category="Fruits")]
    var flavorList:Array;
    // un parámetro de objeto
    [Inspectable(defaultValue="belly:flop,jelly:drop")]
    public var jellyObject:Object;
    // un parámetro de color
    [Inspectable(defaultValue="#ffffff")]
    public var jellyColor:Color;
}
```

Los parámetros pueden ser de cualquier de los siguientes tipos compatibles:

- Matriz
- Objeto
- Lista
- Cadena
- Número
- Booleano
- Nombre de fuente
- Color

Implementación de métodos principales

Existen dos métodos principales que todos los componentes deben implementar: el método de tamaño y el método de inicialización. Si no suplanta estos dos métodos en un componente personalizado, puede producirse un error en Flash Player.

Implementación del método de inicialización

Flash llama al método de inicialización cuando se crea la clase. Como mínimo, el método de inicialización debe llamar al método de inicialización de la superclase. Los parámetros `width`, `height` y `clip` no están configurados correctamente hasta que se llama a este método.

El siguiente método de inicialización de ejemplo de la clase Button llama al método de inicialización de la superclase, configura la escala y otros valores predeterminados y obtiene el valor del atributo de color del objeto UIObject:

```
function init(Void):Void {
    super.init();
    labelField.selectable = false;
    labelField.styleName = this;
    useHandCursor = false;
    // marca como si se utilizara color "color"
    _color = UIObject.textColorList;
}
```

Implementación del método de tamaño

Flash llama al método de tamaño del componente desde el método `setSize()` para disponer el contenido del componente. Como mínimo, el método de tamaño debe llamar al método de tamaño de la superclase, como se muestra en el ejemplo siguiente:

```
function size(Void):Void {
    super.size();
}
```

Gestión de eventos

Los eventos permiten a un componente conocer el momento en el que un usuario ha interactuado con la interfaz, así como los cambios importantes producidos en el aspecto o el ciclo de vida de un componente, como la creación o eliminación de un componente o el cambio de su tamaño.

El modelo de eventos es un modelo distribuidor/detector basado en la especificación Eventos de XML. El usuario escribe un código que registra a los detectores con el objeto de destino, de modo que cuando el objeto de destino distribuya un evento se llame a los detectores.

Los detectores pueden ser funciones u objetos, pero no métodos. El detector recibe un único objeto de evento como el parámetro que contiene el nombre del evento e incluye toda la información relevante sobre éste.

Los componentes generan y distribuyen eventos y consumen (detectan) otros eventos. Un objeto que desea conocer los eventos de otro objeto se registra con este último. Cuando tiene lugar un evento, el objeto distribuye el evento a todos los detectores registrados mediante la llamada de una función solicitada durante el registro. Para recibir varios eventos desde el mismo objeto, debe registrarse para cada evento.

Flash MX 2004 amplía el controlador `on()` de ActionScript para admitir los eventos de componentes. Se admite cualquier componente que declare eventos en su archivo de clase e implemente el método `addEventListener()`.

Eventos frecuentes

A continuación, se muestra una lista de difusión de eventos frecuentes para varias clases. Cada componente debe intentar difundir estos eventos si tienen algún significado para ese componente. Esta lista no incluye los eventos de todos los componentes, sino solamente los que otros componentes pueden utilizar. Aunque algunos eventos no especifican ningún parámetro, todos los eventos cuentan con un parámetro implícito: una referencia al objeto que está difundiendo el evento.

Evento	Parámetros	Utilización
click	Ninguno	Utilizado por Button o siempre que un clic del ratón no tenga otro significado.
scroll	Scrollbar.lineUp, lineDown, pageUp, pageDown, thumbTrack, thumbPosition, endScroll, toTop, toBottom, lineLeft, lineRight, pageLeft, pageRight, toLeft, toRight	Utilizado por ScrollBar y por otros controles de desplazamiento (puntos de desplazamiento en un menú emergente de desplazamiento).
change	Ninguno	Utilizado por List, ComboBox y otros componentes de entrada de texto.
maxChars	Ninguno	Se utiliza cuando el usuario intenta especificar demasiados caracteres en los componentes de entrada de texto.

Asimismo, debido a la herencia de UIComponent, todos los componentes difunden los eventos siguientes:

Evento UIComponent	Descripción
load	El componente está creando o cargando sus subobjetos.
unload	El componente está descargando sus subobjetos.
focusIn	Ahora el componente tiene la selección de entrada. Algunos componentes equivalentes a HTML (ListBox, ComboBox, Button, Text) también pueden emitir la selección, pero todos emiten DOMFocusIn
focusOut	El componente ha perdido la selección de entrada.
move	El componente se ha desplazado a otra ubicación.
resize	Se ha cambiado el tamaño del componente.

La tabla siguiente describe eventos clave frecuentes:

Eventos clave	Descripción
keyDown	Se ha presionado una tecla. La propiedad <code>code</code> contiene el código clave y la propiedad <code>ascii</code> contiene el código ASCII de la tecla que se ha presionado. No consulte el objeto <code>Key</code> de bajo nivel, ya que éste tal vez no haya generado el evento.
keyUp	Se ha soltado una tecla.

Utilización del objeto Event

Un objeto `Event` pasa al detector como un parámetro. El objeto `Event` es un objeto de `ActionScript` cuyas propiedades contienen información sobre el evento que ha tenido lugar. El objeto `Event` se puede utilizar dentro de la función `callback` del detector para acceder al nombre del evento que se difundió o al nombre de instancia del componente que difundió el evento.

Por ejemplo, el código siguiente utiliza la propiedad `target` del objeto `Event` `evtObj` para acceder a la propiedad `label` de la instancia `myButton` y rastrear el valor:

```
listener = new Object();
listener.click = function(evtObj){
    trace("clic en el botón " + evtObj.target.label + " button was clicked");
}
myButton.addEventListener("click", listener);
```

La tabla siguiente enumera las propiedades más frecuentes en los objetos de evento:

Propiedad	Descripción
type	Cadena que indica el nombre del evento. Esta propiedad es obligatoria.
target	Referencia a la instancia del componente que está difundiendo el evento. En general, no es obligatorio que describa este objeto de referencia de forma explícita.

Los eventos más frecuentes, como `click` y `change`, no tienen ninguna propiedad obligatoria a excepción de `type`.

Puede crear de forma explícita un objeto `Event` antes de distribuir el evento, como se muestra en el ejemplo siguiente:

```
var eventObj = new Object();
eventObj.type = "myEvent";
eventObj.target = this;
dispatchEvent(eventObj);
```

También puede utilizar una sintaxis de método abreviado que establezca el valor de la propiedad `type` y distribuya el evento en una sola línea:

```
ancestorSlide.dispatchEvent({type:"revealChild", target:this});
```

En el ejemplo anterior, no es obligatorio definir la propiedad `target`, ya que es implícita.

La descripción de cada evento en la documentación de `Flash MX 2004` muestra una lista con las propiedades de eventos opcionales y las obligatorias. Por ejemplo, el evento `ScrollBar.scroll` utiliza una propiedad `detail` además de las propiedades `type` y `target`. Para más información, consulte las descripciones de eventos en el [Capítulo 4, “Diccionario de componentes”](#), en la [página 45](#).

Distribución de eventos

En el cuerpo del archivo de clase `ActionScript` del componente, se difunden eventos mediante el método `dispatchEvent()`. La firma para el método `dispatchEvent()` es la siguiente:

```
dispatchEvent(eventObj)
```

El parámetro `eventObj` es el objeto `Event` que describe el evento (consulte [“Utilización del objeto Event” en la página 286.](#))

Identificación de controladores de eventos

Debe definir el objeto del controlador de eventos o la función del controlador de eventos que detecta los eventos de componente en el `ActionScript` de la aplicación.

En el ejemplo siguiente se crea un objeto detector, se gestiona un evento `click` y se añade como un detector de eventos a `myButton`:

```
listener = new Object();
listener.click = function(evtObj){
    trace("clic en el botón " + evtObj.target.label + " button was clicked");
}
myButton.addEventListener("click", listener);
```

Además de utilizar un objeto detector, también puede utilizar una función como detector.

Un detector es una función si no pertenece a ningún objeto. Por ejemplo, el código siguiente crea la función de detección `myHandler()` y la registra en `myButton`:

```
function myHandler(eventObj){
    if (eventObj.type == "click"){
        // aquí código propio
    }
}
myButton.addEventListener("click", myHandler);
```

Para más información sobre la utilización del método `addEventListener()`, consulte [“Utilización de detectores de eventos de componentes” en la página 23.](#)

Cuando descubre que un objeto concreto es el único detector de un evento, puede aprovechar que el nuevo modelo de eventos siempre llama a un método en la instancia de componente. Este método es el nombre del evento más la palabra `Handler`. Por ejemplo, para gestionar el evento `click`, escriba el código siguiente:

```
myComponentInstance.clickHandler = function(o){
    // introduzca aquí el código propio
}
```

En el código anterior, la palabra clave `this`, si se utiliza en la función `callback`, tiene el ámbito de `myComponentInstance`.

También puede utilizar objetos detectores que admitan el método `handleEvent()`. Con independencia del nombre del evento, se llama al método `handleEvent()` del objeto detector. Para gestionar varios eventos, es preciso utilizar una sentencia `if...else` o `switch`, por lo que la sintaxis es complicada. Por ejemplo, el código siguiente utiliza una sentencia `if...else` para gestionar los eventos `click` y `enter`:

```
myObj.handleEvent = function(o){
    if (o.type == "click"){
        // aquí código propio
    } else if (o.type == "enter"){
        // aquí código propio
    }
}
```

```

    }
}
target.addEventListener("click", myObj);
target2.addEventListener("enter", myObj);

```

Utilización de metadatos de evento

Añada metadatos de evento al archivo de clase ActionScript para cada detector de eventos. El valor de la palabra clave `Event` es el primer argumento en las llamadas al método `addEventListener()`, como se muestra en el ejemplo siguiente:

```

[Event("click")] // declaración de evento
...
class FCheckBox{
    function addEventListener(eventName:String, eventHandler:Object) {
        ... //eventName es String
    }
}

```

Para más información sobre la palabra clave de metadatos `Event`, consulte [“Event” en la página 279](#).

Aplicación de aspectos

Un control de interfaz de usuario está formado en su totalidad por clips de película adjuntos. Esto significa que todos los activos de un control de interfaz de usuario pueden ser externos al clip de película del control de interfaz de usuario, de modo que otros componentes puedan utilizarlos. Por ejemplo, si el componente necesita funcionalidad de botón, puede reutilizar los activos del componente `Button` existente.

El componente `Button` utiliza un clip de película independiente para representar todos sus estados (`FalseDown`, `FalseUp`, `Disabled`, `Selected`, etc.). Sin embargo, puede asociar los clips de película personalizados, llamados *aspectos*, con estos estados. Durante el tiempo de ejecución, los clips de película antiguos y nuevos se exportan al archivo SWF. Los estados antiguos pasan a ser invisibles para dejar paso a los nuevos clips de película. Esta capacidad para cambiar los aspectos durante la edición y durante el tiempo de ejecución se denomina *aplicar aspectos*.

Para utilizar la aplicación de aspectos en los componentes, cree una variable para los elementos/vinculaciones de cada aspecto que se utilizan en el componente. Esto permite establecer un elemento de aspecto distinto simplemente modificando un parámetro del componente, como se muestra en el ejemplo siguiente:

```
var falseUpSkin = "mySkin";
```

El nombre `“mySkin”` se utiliza como el nombre de la vinculación del símbolo `MovieClip` para mostrar el aspecto `FalseUp`.

El ejemplo siguiente muestra las variables de aspecto para varios estados del componente `Button`:

```

var falseUpSkin:String = "ButtonSkin";
var falseDownSkin:String = "ButtonSkin";
var falseOverSkin:String = "ButtonSkin";
var falseDisabledSkin:String = "ButtonSkin";
var trueUpSkin:String = "ButtonSkin";
var trueDownSkin:String = "ButtonSkin";
var trueOverSkin:String = "ButtonSkin";
var trueDisabledSkin:String = "ButtonSkin";
var falseUpIcon:String = "";

```



```

var falseDownIcon:String = "";
var falseOverIcon:String = "";
var falseDisabledIcon:String = "";
var trueUpIcon:String = "";
var trueDownIcon:String = "";
var trueOverIcon:String = "";
var trueDisabledIcon:String = "";

```

Adición de estilos

Añadir estilos es el proceso que permite registrar todos los elementos gráficos del componente con una clase; esa clase controla entonces los esquemas de color de los gráficos durante el tiempo de ejecución. No es necesario ningún código especial en las implementaciones del componente para admitir los estilos. Los estilos se implementan en su totalidad en las clases y los aspectos base.

Para más información sobre estilos, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 27](#).

Cómo facilitar el acceso a los componentes

Cada vez resulta más importante que el contenido Web sea accesible para las personas con discapacidades. Las personas con deficiencias visuales pueden utilizar el contenido visual de las aplicaciones Flash mediante un software de lector de pantalla, que proporciona una descripción en audio del material que aparece en pantalla.

Flash MX 2004 incluye las siguientes funciones de accesibilidad:

- Desplazamiento personalizado de la selección
- Métodos abreviados de teclado personalizados
- Documentos basados en pantallas y el entorno de edición de pantallas
- La clase Accessibility

Al crear un componente podrá incluir ActionScript que permita que el componente y el lector de pantalla se comuniquen. Cuando los desarrolladores utilicen el componente para crear una aplicación en Flash, emplearán el panel Accesibilidad para configurar la instancia de cada componente.

Añada la línea siguiente al archivo FLA de su componente, en la misma capa en la que va a añadir otras llamadas de ActionScript:

```
mx.accessibility.ComponentName.enableAccessibility();
```

Por ejemplo, la línea siguiente permite la accesibilidad al componente MyButton:

```
mx.accessibility.MyButton.enableAccessibility();
```

Cuando los desarrolladores añaden el componente MyButton a una aplicación, pueden utilizar el panel Accesibilidad para hacerlo accesible a los lectores de pantalla.

Para obtener información sobre el panel Accesibilidad y otras funciones de accesibilidad de Flash, consulte [“Creación de contenido accesible”](#), en el apartado Utilización de Flash de la Ayuda.

Exportación del componente

Flash MX 2004 exporta componentes como paquetes de componentes (archivos SWC). Cuando distribuye un componente, sólo necesita entregar a los usuarios el archivo SWC. Este archivo contiene todo el código, los archivos SWF, las imágenes y los metadatos asociados con el componente, de modo que los usuarios puedan colocarlo fácilmente en su entorno de Flash.

En esta sección se describe un archivo SWC y se explica cómo importar y exportar archivos SWC en Flash.

Aspectos básicos de los archivos SWC

Un archivo SWC es un archivo zip (empaquetado y ampliado mediante el formato de archivos PKZip) que genera la herramienta de edición de Flash.

En la tabla siguiente se describen los contenidos de un archivo SWC.

Archivo	Descripción
catalog.xml	(Obligatorio) Muestra una lista con el contenido del paquete de componentes y cada uno de sus componentes y funciona como directorio para los otros archivos del archivo SWC.
Código fuente	<p>Si el componente se crea mediante Flash MX 2004, el código fuente es uno o varios archivos ActionScript que contienen una declaración de clase para el componente.</p> <p>El código fuente sólo se utiliza para la verificación de tipos cuando se crean subclases para los componentes y la herramienta de edición no lo compila, ya que el código de bits ya se encuentra en el archivo SWF que se implementa.</p> <p>El código fuente puede contener definiciones de clases intrínsecas que no contienen cuerpos de función y sólo se proporcionan para la verificación de tipos.</p>
Implementación de archivos SWF	(Obligatorio) Archivos SWF que implementan los componentes. En un solo archivo SWF pueden definirse uno o varios componentes. Si el componente se crea mediante Flash MX 2004, sólo se exporta un componente por archivo SWF.
Archivos SWF de vista previa dinámica	(Opcional) Si se especifica, estos archivos SWF se utilizan para realizar una Vista previa dinámica en la herramienta de edición. Si se omiten, se utilizarán los archivos SWF que se van a implementar para realizar la vista previa dinámica. El archivo SWF de vista previa dinámica puede omitirse en casi todas las clases; debe incluirse sólo si la apariencia del componente depende de datos dinámicos (por ejemplo, un campo de texto que muestra el resultado de una llamada de servicio Web).
Información de depuración	(Opcional) Archivo SWD que corresponde al archivo SWF que se va a implementar. El nombre de archivo siempre es el mismo que el del archivo SWF, pero con la extensión .swd. Si se incluye en el archivo SWC, se permite la depuración del componente.

Archivo	Descripción
Icono	(Opcional) Archivo PNG que contiene el icono de 18 x 18 de 8 bits por píxel que se utiliza para mostrar un componente en la interfaz de usuario de la herramienta de edición. Si no se proporciona ningún icono, se muestra un icono predeterminado. (Véase “Adición de un icono” en la página 292.)
Inspector de propiedades	(Opcional) Si se especifica, este archivo SWF se utiliza como el inspector de propiedades personalizado en la herramienta de edición. Si se omite, se muestra al usuario el inspector de propiedades predeterminado.

Para visualizar el contenido de un archivo SWC, puede abrirlo mediante cualquier utilidad de compresión que admita el formato PKZip (incluido WinZip).

De forma opcional, puede incluir otros archivos en el archivo SWC una vez que lo haya generado en el entorno Flash. Por ejemplo, es posible que desee incluir un archivo Readme (Léame) o el archivo FLA si desea que los usuarios puedan acceder al código fuente del componente.

Los diversos archivos SWC se expanden en un único directorio, por lo que cada componente debe disponer de un nombre de archivo exclusivo para evitar conflictos.

Utilización de archivos SWC

En esta sección se describe cómo crear e importar archivos SWC. Debe proporcionar instrucciones para importar archivos SWC a los usuarios de sus componentes.

Creación de archivos SWC

Flash MX 2004 y Flash MX Professional 2004 permiten crear archivos SWC mediante la exportación de un clip de película como archivo SWC. Cuando crea un archivo SWC, Flash informa de los errores en el tiempo de compilación como si estuviera probando una aplicación Flash.

Para exportar un archivo SWC:

- 1 Seleccione un elemento en la biblioteca Flash.
- 2 Haga clic con el botón derecho del ratón (Windows) o haga clic con la tecla Control presionada (Macintosh) sobre el elemento y seleccione Exportar archivo SWC.
- 3 Guarde el archivo SWC.

Importación de archivos SWC de componente a Flash

Cuando distribuya sus componentes a otros desarrolladores, puede incluir las siguientes instrucciones para que puedan instalarlos y utilizarlos inmediatamente.

Para utilizar un archivo SWC en el entorno de edición de Flash, haga lo siguiente:

- 1 Cierre el entorno de edición de Flash.
- 2 Copie el archivo SWC en el directorio `flash_root/es/First Run/Components`.
- 3 Inicie el entorno de edición de Flash o vuelva a cargar el panel Componentes.
El icono del componente debe aparecer en el panel Componentes.

Cómo facilitar la utilización del componente

Una vez que haya creado el componente y lo haya preparado para su empaquetado, puede facilitar su utilización por parte de los usuarios. En esta sección se describen algunas técnicas para facilitar la utilización del componente.

Adición de un icono

Puede añadir un icono que represente el componente en el panel Componentes del entorno de edición de Flash.

Para añadir un icono del componente, haga lo siguiente:

- 1 Cree una imagen nueva.
La imagen debe medir 18 píxeles cuadrados y guardarse en formato PNG. Debe ser de 8 bits y con transparencia alfa, y el píxel superior izquierdo debe ser transparente para admitir máscaras.
- 2 Añada la definición siguiente al archivo de clase ActionScript del componente antes de la definición de la clase:

```
[IconFile("component_name.png")]
```
- 3 Añada la imagen al mismo directorio en el que se encuentra el archivo FLA. Cuando exporte el archivo SWC, Flash incluirá la imagen en la raíz del archivo.

Utilización de la vista previa dinámica

La función Vista previa dinámica, que se activa de forma predeterminada, permite ver los componentes del escenario tal como aparecerán en el contenido de Flash publicado, con su tamaño aproximado.

Ya no es necesario añadir una vista previa dinámica cuando se crean componentes mediante la arquitectura de la versión 2. Los archivos SWC del componente incluyen el archivo SWF que va a implementarse y el componente utiliza dicho archivo SWF en el escenario de Flash.

Adición de sugerencias

Las sugerencias aparecen cuando un usuario pasa el ratón por encima del nombre o del icono del componente en el panel Componentes del entorno de edición de Flash.

Para añadir sugerencias al componente, utilice la palabra clave `tiptext` fuera de la definición de la clase en el archivo de clase ActionScript del componente. Debe marcar esta palabra clave mediante un asterisco (*) y precederla del símbolo @ para que el compilador la reconozca adecuadamente.

En el ejemplo siguiente se muestra la sugerencia para el componente CheckBox:

```
* @tiptext Componente CheckBox básico. Amplía Button.
```

Recomendaciones para el diseño de un componente

Utilice las siguientes recomendaciones cuando diseñe un componente:

- Intente que el tamaño del archivo sea lo más reducido posible.

- Intente que su componente sea lo más reutilizable posible mediante la generalización de sus funciones.
- Utilice el nuevo modelo de eventos en lugar de la sintaxis `on(event)`.
- Utilice la clase `Border`, en lugar de los elementos gráficos, cuando dibuje los bordes de los objetos.
- Utilice la aplicación de aspectos basados en etiquetas.
- Utilice la propiedad `symbolName`.
- Asuma un estado inicial. Puesto que las propiedades de estilo se encuentran ahora en el objeto, puede establecer la configuración inicial para los estilos y las propiedades para que su código de inicialización no tenga que establecerlo cuando se cree el objeto, a no ser que el usuario sustituya el estado predeterminado.
- Cuando defina el símbolo no seleccione la opción Exportar en primer fotograma, a no ser que sea absolutamente necesario. Flash carga el componente justo antes de que se utilice en la aplicación Flash, por lo que si selecciona esta opción Flash precargará el componente en el primer fotograma de su principal. La razón por la que habitualmente no se precarga el componente en el primer fotograma tiene que ver con la Web: el componente se carga antes de que se inicie el precargador, lo que inutiliza el objetivo de este último.

ÍNDICE ALFABÉTICO

A

- accesibilidad
 - editar 14
 - para componentes personalizados 289
 - y componentes 14
- ActionScript
 - escribir para un nuevo componente 270
 - flujo de trabajo para escribir para un nuevo componente 271
- Activos de los archivos FLA, almacenamiento para los archivos de componentes 265
- addEventListener 287
- ampliar clases 273
- añadir componentes mediante ActionScript 20
- aplicar aspectos 37
 - para componentes personalizados 288
- archivos de componentes, almacenar 267
- archivos de origen de componentes 268
- archivos SWC 14
 - crear 291
 - importar 291
 - se explica el formato del archivo 290
 - trabajar 18
 - y clips compilados 14
- arquitectura de componentes de la versión 1, diferencias con la versión 2 265
- arquitectura de componentes de la versión 2
 - modificaciones de la versión 1 265
 - utilizar el archivo SWC para la vista previa dinámica 292
- aspectos 37
 - aplicar 39
 - aplicar a subcomponentes 40
 - editar 38

B

- Biblioteca, panel 16

C

- cambiar el tamaño de los componentes 21
- capas de símbolos, editar para un nuevo componente 269
- captadores, definir para propiedades 275
- categorías
 - administradores 47
 - controles de la IU 45
 - datos 47
 - medios 46
 - pantallas 47
- Centro de soporte de Macromedia Flash 10
- clase
 - archivos, almacenamiento para componentes 266
 - nombre, para un componente personalizado 275
- clase de componente de diapositivas 47
- clase de componente de formularios 47
- clase Form 107
- clase Label 110
- clase List 116
- clase NumericStepper
 - métodos 155
 - propiedades 155
- clase PopUpManager, métodos 160
- clase principal, seleccionar para un nuevo componente 272
- clase Screen 186
- clase Slide 204
- clase UIComponent
 - definida 273
- clase UIComponent y herencia de componente 13
- clase UIObject, definida 273

- clases
 - ampliar 270, 273
 - CheckBox 62
 - clase Button 52
 - clase Form 107
 - clase Label 110
 - clase List 116
 - ComboBox 71
 - componente ProgressBar 166
 - componente ScrollPane 189
 - crear subclases 273
 - FocusManager 102
 - importar 272
 - Loader 143
 - NumericStepper 155
 - pantalla 186
 - RadioButton 180
 - seleccionar una clase principal 272
 - Slide 204
 - TextArea 207
 - TextInput 219
 - UIComponent 273
 - UIObject 273
 - y herencia de componente 13
- clases de componente de pantalla 47
- className 275
- clickHandler 24
- clips compilados 14
 - del panel Biblioteca 16
 - trabajar 18
- colores
 - definir propiedades de estilo 31
 - herencia, seguir 202
- componente Accordion 47
- componente Button 48
 - clase Button 52
 - crear aplicaciones 49
 - eventos 53
 - métodos 52
 - parámetros 49
 - personalizar 50
 - propiedades 53
 - utilizar 48
 - utilizar aspectos 51
 - utilizar estilos 50
- componente CellRenderer 59
- componente CheckBox 59
 - clase CheckBox 62
 - crear aplicaciones 60
 - eventos 63
 - métodos 63
 - parámetros 60
 - propiedades 63
 - utilizar 60
 - utilizar aspectos 62
 - utilizar estilos 61
- componente ComboBox 67
 - clase ComboBox 71
 - crear aplicaciones 69
 - métodos 72
 - parámetros 69
 - propiedades 72
 - utilizar 69
 - utilizar aspectos 71
 - utilizar estilos 70
- componente DataGrid 95
- componente DataHolder 95
- componente DataProvider 95
- componente DataSet 95
- componente DateChooser 95
- componente DateField 95
- componente Label 107
 - clase Label 110
 - crear aplicaciones 109
 - eventos 110
 - métodos 110
 - parámetros 108
 - personalizar 109
 - propiedades 110
 - utilizar 108
 - utilizar estilos 109
- componente List 112
 - crear aplicaciones 114
 - eventos 119
 - métodos 117
 - parámetros 113
 - personalizar 114
 - propiedades 118
 - utilizar 113
 - utilizar estilos 115
- componente Loader 141
 - clase Loader 143
 - crear aplicaciones 142
 - eventos 144
 - métodos 143
 - parámetros 141
 - personalizar 142
 - propiedades 143
 - utilizar 141
- componente MediaDisplay 151

- componente MediaPlayer 151
- componente Menu 151
- componente MenuBar 151
- componente NumericStepper 151
 - clase NumericStepper 155
 - crear aplicaciones 153
 - eventos 156
 - parámetros 152
 - personalizar 153
 - utilizar 152
 - utilizar aspectos 154
 - utilizar estilos 154
- componente PopUpManager 160
- componente ProgressBar 162
 - clase ProgressBar 166
 - crear aplicaciones 163
 - eventos 167
 - métodos 167
 - parámetros 163
 - personalizar 165
 - propiedades 167
 - utilizar 162
 - utilizar aspectos 166
 - utilizar estilos 165
- componente RadioButton 176
 - clase RadioButton 180
 - crear aplicaciones 177
 - eventos 181
 - métodos 180
 - parámetros 177
 - personalizar 178
 - propiedades 180
 - utilizar 177
 - utilizar aspectos 179
 - utilizar estilos 178
- componente RDBMSResolver 186
- componente RemoteProcedureCall 186
- componente ScrollPane 186
 - clase ScrollPane 189
 - crear aplicaciones 188
 - eventos 191
 - métodos 190
 - parámetros 187
 - personalizar 189
 - propiedades 190
 - utilizar 187
 - utilizar aspectos 189
 - utilizar estilos 189
- componente TextArea 204
 - clase TextArea 207
 - crear aplicaciones 205
 - eventos 208
 - parámetros 205
 - personalizar 206
 - propiedades 208
 - utilizar aspectos 207
 - utilizar estilos 206
- componente TextInput 216
 - clase TextInput 219
 - crear aplicaciones 217
 - eventos 220
 - métodos 219
 - parámetros 217
 - personalizar 218
 - propiedades 220
 - utilizar 216
 - utilizar estilos 218
- componentes
 - añadir a documentos de Flash 18
 - añadir dinámicamente 20
 - arquitectura 12
 - cambiar el tamaño 21
 - categorías 45
 - categorías, descripción 12
 - clase Form 107
 - compatibilidad con Flash Player 12
 - DepthManager 95
 - disponibles en Flash MX 2004 8
 - disponibles en Flash MX Professional 2004 8
 - eliminar 21
 - FocusManager 101
 - herencia 13
 - instalar 15
- componentes DataBinding 95
- componentes de datos 47
- componentes de la versión 1 (v1) 26
- componentes de la versión 1 (v1), actualizar 26
- componentes de la versión 2 (v2)
 - ventajas y descripción 11
 - y Flash Player 12
- componentes de medios 46
- componentes del administrador 47
- Componentes, panel 15
- constructor, escribir para un nuevo componente 274
- controles de la interfaz de usuario (IU) 45
- convenciones tipográficas, en la documentación de componentes 10

- crear componentes
 - accesibilidad 289
 - añadir eventos 287
 - añadir texto de información 292
 - aplicar aspectos 288
 - crear archivos SWC 291
 - definir parámetros 283
 - estilos 289
 - eventos frecuentes 285
 - exportar 290
 - gestionar eventos 284
 - implementar métodos principales 283
 - importar archivos SWC 291
 - metadatos de evento 288
 - seleccionar el propietario de un símbolo 275
 - seleccionar un nombre de clase 275
 - seleccionar un nombre de símbolo 275
 - utilizar sentencias de metadatos 276
 - vista previa dinámica con el archivo SWC 292
- crear un componente
 - ampliar la clase de un componente 270
 - añadir parámetros 270
 - añadir un icono 292
 - clase UIComponent definida 273
 - clase UIObject definida 273
 - crear una subclase de una clase 273
 - definir un número de versión 274
 - editar capas de símbolos 269
 - ejemplo de código para el archivo de clase 271
 - escribir ActionScript 270
 - escribir un constructor 274
 - proceso para escribir ActionScript 271
 - seleccionar una clase principal 272
 - símbolo del componente 268
- CSSStyleDeclaration 28, 29

D

- declaración de estilo global 27
- declaraciones de estilo
 - clase predeterminada 30
 - crear personalizadas 29
 - definir clase 30
 - global 28
- defaultPushButton 25
- definidores, definir para propiedades 275
- DepthManager 25
 - clase 95
 - métodos 96
- desplazamiento de selección
 - crear 24

- detectores 23
 - registrar 23
- detectores de eventos 23
- documentación
 - guía de términos 10
 - información general 9

E

- editar símbolos, para componentes 267
- ejemplo de código del archivo de clase
 - del componente 271
- ejemplos de código para desarrollar componentes 268
- estilos 27
 - admitidos 33
 - definir en una instancia 28
 - definir global 28
 - definir personalizados 29
 - determinar precedencia 31
 - establecer 27, 32
 - herencia, seguir 202
 - para componentes personalizados 289
- estilos de instancia 27
- etiquetas 21
- etiquetas para metadatos 276
- evento
 - metadatos 279, 288
- eventos 22
 - añadir 287
 - difundir 23
 - eventos frecuentes 285
 - gestión 284
- eventos ComboBox 73
- exportar componentes personalizados 290

F

- Flash MX 2004, componentes disponibles 8
- Flash MX Professional 2004, componentes disponibles 8
- Flash Player
 - soporte 26
 - y componentes 12
- FocusManager 24, 101
 - clase 102
 - crear aplicaciones 102
 - parámetros 102
 - personalizar 102
 - utilizar 101
- funciones de detector 24

G

global
ruta de clases 266

H

handleEvent 24
herencia
 en componentes de la v2 13
 seguir, para estilos y colores 202
hoja de estilo de clase predeterminada 30
hojas de estilo de clase 27
hojas de estilo personalizadas 27

I

icono para un componente personalizado 292
identificadores de vinculación
 para aspectos 37
Importar clases 272
Inspectable en sentencias de metadatos,
 propiedades 277
Inspector de componentes, panel 16
Inspector de propiedades 16
instalación
 instrucciones 9
 verificar 9
instalar componentes 8
instancias
 definir estilo 28
 definir estilos 28
interfaz API de pantalla 47

L

lectores de pantalla
 accesibilidad 14

M

Macromedia DevNet 10
metadatos 276–282
 ComponentTask 282
 etiquetas 276
 evento 279, 288
 explicación 276
 Inspectable, propiedades 277
 sintaxis 276
método de inicialización, implementar 283
método de tamaño, implementar 284
método handleEvent 24

métodos

 definir captadores y definidores 275
 implementar 283
 inicialización, implementar 283
 tamaño, implementar 284

N

name
 símbolo, para un componente personalizado 275
nombre
 clase 275
números de versión para componentes 274

O

objetos de evento 23
objetos detectores 23
on() 22
orden de tabulación, para componentes 101

P

paquetes 13
parámetros
 añadir a un nuevo componente 270
 definir 21, 283
 establecer 16
 Inspectable, en sentencias de metadatos 277
 visualizar 16
personalizar color 27
personalizar texto 27
profundidad
 administrar 25
propiedades de aspecto
 cambiar en el prototipo 42
 establecer 37
propiedades de estilo
 color 31
 establecer 32
 obtener 32
propiedades, para estilos 28
prototipo 42

R

recursos, adicionales 10
requisitos del sistema 8
ruta de clase
 global 266
 local 266

- ruta de clases
 - aspectos básicos 266
 - cambiar 267
 - UserConfig, directorio 266
- ruta de clases local 266

S

- selección 24
- setSize() 21
- símbolo
 - nombre, para un componente personalizado 275
 - propietario, para un componente personalizado 275
- símbolo del componente, crear 268
- sintaxis, para sentencias de metadatos 276
- StyleManager, métodos 202
- subclases, utilizar para sustituir aspectos 42
- subcomponentes, aplicar aspectos 40
- sugerencias para el código, activar 22

T

- tabIndex 25
- tareas, metadatos 282
- tema Halo 35
- tema Sample 35
- temas 35
 - aplicar 35
 - crear 36
- términos utilizados en la documentación 10
- texto de información, para un componente personalizado 292
- tipos de componente
 - Accordion 47
 - administradores 47
 - CellRenderer 59
 - CheckBox 59
 - clase Screen 186
 - clase Slide 204
 - ComboBox 67
 - componente Alert 48
 - componente Button 48
 - controles de la IU 45
 - DataGrid 95
 - DataHolder 95
 - DataProvider 95
 - DataSet 95
 - DateField 95
 - datos 47
 - Label 107
 - List 112

- Loader 141
- MediaController 151
- MediaDisplay 151
- MediaPlayerback 151
- medios 46
- Menu 151
- MenuBar 151
- NumericStepper 151
- pantallas 47
- paquete DataBinding 95
- PopUpManager 160
- ProgressBar 162
- RadioButton 176
- RDBMSResolver 186
- RemoteProcedureCall 186
- ScrollPane 186
- StyleManager 202
- TextArea 204
- TextInput 216

V

- versión 1, componentes
 - actualizar 26
- vista previa de componentes 17
- vista previa dinámica 17
 - para un componente personalizado 292