macromedia®

FLASH® 8

Ampliación de Flash

Agradecimientos

Dirección del proyecto: Sheila McGinn

Redacción: Jay Armstrong

Directora de edición: Rosana Francescato

Redactora jefe: Lisa Stanziano

Edición: Geta Carlson, Evelyn Eldridge, Mark Nigara

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

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

Agradecimientos especiales a Jody Bleyle, Mary Burger, Lisa Friendly, Stephanie Gowin, Bonnie Loo, Mary Ann Walsh, Erick Vera, que han probado la versión beta, así como a todos los equipos de ingeniería y control de calidad de Flash y Flash Player.

Primera edición: Septiembre de 2005

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

Contenido

Introducción

Como usuario de Macromedia Flash, probablemente estará familiarizado con ActionScript, que le permite crear scripts que se ejecutan en tiempo de ejecución en Macromedia Flash Player. La interfaz de programación de aplicaciones JavaScript (API JavaScript) de Flash es una herramienta de programación complementaria que le permite crear scripts que se ejecutan en el entorno de edición.

En este documento se describen los objetos, métodos y propiedades disponibles en la API JavaScript. Se da por sentado que conoce la forma de utilizar los comandos que se describen en este documento cuando trabaja en el entorno de edición. Si tiene alguna duda sobre la función de un determinado comando, consulte otros documentos de la Ayuda de Flash, como el manual *Utilización de Flash*, donde podrá buscar dicha información.

En este documento también se da por sentado que el lector conoce la sintaxis de JavaScript o de ActionScript, además de conceptos básicos de programación como funciones, parámetros y tipos de datos.

Este capítulo contiene las siguientes secciones:

Información general sobre la API JavaScript de Macromedia Flash

El lenguaje ActionScript permite escribir scripts para realizar acciones en el entorno de Flash Player (es decir, mientras se reproduce un archivo SWF). Con la API JavaScript de Flash se pueden escribir scripts para realizar diversas acciones en el entorno de edición de Flash (es decir, mientras el usuario tiene abierto el programa Flash). Estos scripts sirven para aumentar la eficacia del proceso de edición. Por ejemplo, se pueden escribir scripts para automatizar tareas repetitivas, añadir herramientas personalizadas al panel Herramientas o incorporar efectos de línea de tiempo.

La API JavaScript de Flash es similar a la API JavaScript de Macromedia Dreamweaver y Macromedia Fireworks (que, a su vez, se diseñaron basándose en la API JavaScript de Netscape). La API JavaScript de Flash se basa en un modelo de objetos de documento (DOM o Document Object Model), que permite acceder a los documentos de Flash empleando objetos JavaScript. La API JavaScript de Flash incluye todos los elementos de la API JavaScript de Netscape, además del DOM de Flash. En este documento se describen estos objetos añadidos y sus métodos y propiedades. Puede utilizar cualquiera de los elementos del lenguaje JavaScript nativo en un script de Flash, pero sólo tendrán efecto los elementos que tengan sentido en el contexto de un documento de Flash.

La API JavaScript también contiene una serie de métodos que permiten implementar extensibilidad utilizando una combinación de código JavaScript y C personalizado. Para más información, consulte el Capítulo 3, “Extensibilidad de nivel C”, en la página 543.

El intérprete de JavaScript en Flash es el motor Mozilla SpiderMonkey, versión 1.5, disponible en la Web en www.mozilla.org/js/spidermonkey/. SpiderMonkey es una de las dos implementaciones de referencia del lenguaje JavaScript desarrollado por Mozilla.org. Se trata del mismo motor que incorpora el navegador Mozilla.

SpiderMonkey implementa el lenguaje JavaScript básico que se define en la especificación ECMAScript (ECMA-262) edición 3 y es totalmente compatible con la especificación. Sólo son incompatibles los objetos host específicos del navegador que no forman parte de la especificación ECMA-262. Del mismo modo, un gran número de guías de referencia de JavaScript distinguen entre JavaScript básico y de cliente (relacionado con el navegador). Sólo JavaScript básico se aplica al intérprete de JavaScript de Flash.

Creación de archivos JSFL

Puede utilizar Macromedia Flash 8 o el editor de texto que prefiera para escribir y editar archivos JavaScript de Flash (JSFL). Si utiliza Flash, estos archivos usan la extensión .jsfl de forma predeterminada.

También puede crear un archivo JSFL seleccionando comandos del panel Historial y, a continuación, haciendo clic en el botón Guardar del panel Historial o seleccionando Guardar como comando en el menú emergente Opciones. El archivo de comando (JSFL) se guardará en la carpeta Commands (consulte "Almacenamiento de archivos JSFL" en la página 7). A continuación, podrá abrir el archivo y editarlo de la misma forma que cualquier otro archivo de script.

El panel Historial ofrece también otras opciones que resultan muy útiles. Se pueden copiar los comandos seleccionados en el portapapeles, así como ver los comandos JavaScript que se generan mientras se está trabajando con Flash.

Para copiar comandos del panel Historial en el portapapeles:

1. Seleccione uno o varios comandos en el panel Historial.
2. Siga uno de estos procedimientos:
   - Haga clic en el botón Copiar.
   - Seleccione Copiar pasos en el menú emergente Opciones.

Para ver los comandos JavaScript en el panel Historial:

- Seleccione Ver > JavaScript en el panel del menú emergente Opciones.

Almacenamiento de archivos JSFL

Puede tener disponibles scripts JSFL dentro del entorno de edición de Flash; para ello, debe almacenarlos en una de las distintas carpetas dentro de la carpeta Configuration. De forma predeterminada, la carpeta Configuration se encuentra en la siguiente ubicación:

- Windows 2000 o Windows XP:

  unidad de inicio\Documents and Settings*usuario*\Configuración local\Datos de programa\Macromedia\ Flash 8*idioma*\Configuration\
- Mac OS X:

  Macintosh HD/Users/*nombreUsuario*/Library/Application Support/Macromedia/Flash 8/*idioma*/Configuration/

Para determinar la ubicación de la carpeta Configuration, utilice fl.configDirectory o fl.configURI.

Dentro de la carpeta Configuration, las carpetas siguientes pueden contener scripts a los que puede acceder en el entorno de edición: Behaviors, Commands (para scripts que aparecen en el menú Comandos), Effects (para efectos de línea de tiempo), JavaScript (para scripts que utiliza el asistente de script), Tools (para herramientas ampliables del panel Herramientas) y WindowSWF (para paneles que aparecen en el menú Ventana). Este documento se centra en los scripts utilizados para comandos, efectos y herramientas.

Si se edita un script en la carpeta Commands, el nuevo script quedará disponible de inmediato en Flash. Si se edita un script para un efecto o una herramienta ampliable, se deberá cerrar y reiniciar Flash, o bien utilizar el comando fl.reloadEffects() o fl.reloadTools(). Sin embargo, si se ha utilizado un script para añadir una herramienta ampliable al panel Herramientas y se edita después el script, se deberá quitar para volver a añadir la herramienta al panel Herramientas, o bien cerrar y reiniciar Flash para que la herramienta revisada quede disponible.

Puede guardar los archivos de comandos, efectos y herramientas en tres lugares en los que se encontrarán accesibles en el entorno de edición.

- Para los scripts que aparecerán como elementos en el menú Comandos, guarde el archivo JSFL en la carpeta Commands en la siguiente ubicación:
  - Windows 2000 o Windows XP:

    unidad de inicio\Documents and Settings*usuario*\Configuración local\Datos de programa\Macromedia\ Flash 8*idioma*\Configuration\Commands
  - Mac OS X:

    Macintosh HD/Users/*nombreUsuario*/Library/Application Support/Macromedia/Flash 8/*idioma*/Configuration/Commands
- Para los scripts que aparecerán como herramientas ampliables en el panel Herramienta, guarde el archivo JSFL en la carpeta Tools en la ubicación siguiente:
  - Windows 2000 o Windows XP:

    unidad de inicio\Documents and Settings*usuario*\Configuración local\Datos de programa\Macromedia\ Flash 8*idioma*\Configuration\Tools
  - Mac OS X:

    Macintosh HD/Users/*nombreUsuario*/Library/Application Support/Macromedia/Flash 8/*idioma*/Configuration/Tools

- Para los scripts que aparecerán como efectos de línea de tiempo en el panel Efectos, guarde el archivo JSFL en la carpeta Effects en la ubicación siguiente:
  - Windows 2000 o Windows XP:

    unidad de inicio\Documents and Settings*usuario*\Configuración local\Datos de programa\Macromedia\ Flash 8*idioma*\Configuration\Effects
  - Mac OS X:

    Macintosh HD/Users/*nombreUsuario*/Library/Application Support/Macromedia/Flash 8/*idioma*/Configuration/Effects

Si un archivo JSFL va acompañado de otros archivos, por ejemplo, de archivos XML, éstos deberán almacenarse en el mismo directorio que el archivo JSFL.

Ejecución de archivos JSFL

Existen varias formas de ejecutar archivos JSLF. En esta sección se describen las más frecuentes.

Para ejecutar un script que se encuentra en la carpeta Commands, siga uno de los estos procedimientos:

- Seleccione Comandos > *Nombre de script*.
- Utilice el método abreviado de teclado que haya asignado al script. Para asignar un método abreviado, utilice Edición > Métodos abreviados de teclado y seleccione Menú de Comandos de Dibujo en el menú Comandos emergente. Expanda el nodo Comandos en el árbol de menús para ver una lista de los scripts disponibles.

Para ejecutar un script de comando que no se encuentra en la carpeta Commands, siga uno de estos procedimientos:

- En el entorno de edición, seleccione Comandos > Ejecutar comando y, a continuación, seleccione el script que desea ejecutar.
- Dentro del script, utilice el comando fl.runScript().
- En el sistema de archivos, haga doble clic en el archivo de script.

Para añadir al panel Herramientas una herramienta implementada en un archivo JSFL:

1. Copie en la carpeta Tools el archivo JSFL de la herramienta y los restantes archivos asociados (consulte "Almacenamiento de archivos JSFL" en la página 7).
2. Seleccione Edición > Personalizar panel de herramientas (Windows) o Flash > Personalizar panel de herramientas (Macintosh).
3. Añada la herramienta a la lista de herramientas disponibles.
4. Haga clic en Aceptar.

Puede añadir comandos API JavaScript individuales a archivos de ActionScript empleando la función `MMExecute()`, que se documenta en *Referencia del lenguaje ActionScript 2.0*. Sin embargo, la función `MMExecute()` sólo tiene efecto cuando se utiliza en el contexto de un elemento de la interfaz de usuario personalizada, como un inspector de propiedades de componentes o un panel SWF en el entorno de edición. Aunque se llamen desde ActionScript, los comandos API JavaScript no tienen efecto en Flash Player ni fuera del entorno de edición.

Para enviar un comando desde un script de ActionScript:

- Utilice la siguiente sintaxis (puede concatenar varios comandos en una cadena):

```
MMExecute(Javascript command string);
```

También se puede ejecutar un script desde la línea de comandos.

Para ejecutar un script desde la línea de comandos de Windows:

- Utilice la siguiente sintaxis (añada información de la ruta según sea necesario):

```
"flash.exe" myTestFile.jsfl
```

Para ejecutar un script desde la línea de comandos de Macintosh:

- Utilice la siguiente sintaxis (añada información de la ruta según sea necesario):

```
osascript -e 'tell application "flash" to open alias "Mac OS
  X:Users:user:myTestFile.jsfl" '
```

El comando `osascript` también puede ejecutar AppleScript en un archivo. Por ejemplo, podría incluir el siguiente texto en un archivo denominado myScript:

```
tell application "flash"
  open alias "Mac OS X:Users:user:myTestFile.jsfl"
end tell
```

A continuación, para invocar el script, utilizaría este comando:

```
osascript myScript
```

Novedades de la API JavaScript

En Flash 8, se han añadido varias funciones y objetos de nivel superior. Además, algunos objetos existentes tienen ahora nuevos métodos o propiedades. A continuación se resumen estas adiciones, junto con otros cambios. También se proporcionan nuevos ejemplos; consulte "Implementaciones de muestra" en la página 20.

Si es la primera vez que utiliza la API JavaScript, puede omitir esta sección y pasar directamente a "El modelo de objetos de documento de Flash" en la página 14.

Nuevos métodos de nivel superior

El siguiente método de nivel superior aparece por primera vez en Flash 8:

```
confirm()
```

Los siguientes métodos de nivel superior se implementaron en Flash MX 2004 pero no se han documentado hasta esta versión:

```
alert()
prompt()
```

Nuevos objetos

Los objetos siguientes aparecen por primera vez en Flash 8:

Objeto Filter

Objeto Project

Objeto ProjectItem

El siguiente objeto se implementó en la versión actualizada de Flash MX 2004, pero no se ha documentado hasta esta versión:

Objeto FLfile

Nuevos métodos y propiedades

Los siguientes métodos y propiedades aparecen por primera vez en Flash 8:

```
componentsPanel.reload()
document.addFilter()
document.changeFilterOrder()
document.crop()
document.deleteEnvelope()
```

```
document.disableAllFilters()
document.disableFilter()
document.disableOtherFilters()
document.enableAllFilters()
document.enableFilter()
document.exportPNG()
document.getBlendMode()
document.getFilters()
document.getMetadata()
document.importFile()
document.intersect()
document.punch()
document.removeAllFilters()
document.removeFilter()
document.setBlendMode()
document.setFilterProperty()
document.setFilters()
document.setMetadata()
document.swapStrokeAndFill()
document.union()
document.zoomFactor
element.layer
element.selected
fill.focalPoint
fill.linearRGB
fill.overflow
fl.browseForFolderURL()
fl.closeProject()
fl.contactSensitiveSelection
fl.createProject()
fl.objectDrawingMode
fl.getAppMemoryInfo()
fl.getProject()
```

```
fl.objectDrawingMode
fl.showIdleMessage()
frame.getCustomEase()
frame.hasCustomEase
frame.setCustomEase()
frame.useSingleEaseCurve
shape.isDrawingObject
stroke.capType
stroke.joinType
stroke.miterLimit
stroke.strokeHinting
stroke.scaleType
stroke.shapeFill
symbolInstance.blendMode
symbolInstance.cacheAsBitmap
symbolInstance.filters
symbolItem.scalingGrid
symbolItem.scalingGridRect
text.antiAliasSharpness
text.antiAliasThickness
textAttrs.letterSpacing
text.fontRenderingMode
videoItem.sourceFilePath
videoItem.videoType
xmlui.getControlItemElement()
xmlui.getEnabled()
xmlui.getVisible()
xmlui.setControlItemElement()
xmlui.setControlItemElements()
xmlui.setEnabled()
xmlui.setVisible()
```

Otros cambios

Los siguientes elementos tienen nuevos parámetros, valores aceptables adicionales para los parámetros existentes u otros cambios de implementación en Flash 8:

- `document.setSelectionBounds()`
- `document.setSelectionRect()`
- `instance.instanceType`
- `outputPanel.save()`
- `fl.openProject()`
- `text.border`, `text.useDeviceFonts`, `textAttrs.autoKern` (ya no se aplica solamente al texto estático)

Propiedades no admitidas

En esta versión no se admite la siguiente propiedad:

`textAttrs.characterSpacing` (se recomienda utilizar la propiedad `textAttrs.letterSpacing`)

El modelo de objetos de documento de Flash

El modelo de objetos de documento (DOM) de la API JavaScript de Flash se compone de una serie de funciones de nivel superior (consulte "Funciones y métodos de nivel superior" en la página 25) y dos objetos de nivel superior: FLfile y Flash (fl). Cada uno de estos objetos se encuentran disponibles en todo momento en un script porque siempre están presentes cuando se abre el entorno de edición de Flash. Para más información, consulte Objeto FLfile y Objeto Flash (fl).

Para hacer referencia al objeto Flash, puede utilizar `flash` o `fl`. Por ejemplo, para cerrar todos los archivos abiertos, puede utilizar cualquiera de las sentencias siguientes:

```
flash.closeAll();
fl.closeAll();
```

El objeto Flash contiene los siguientes objetos *secundarios*:

Objeto	Modo de acceso
Objeto componentsPanel	Utilice fl.componentsPanel para acceder al objeto componentsPanel. Este objeto corresponde al panel Componentes en el entorno de edición de Flash.
Objeto Document	Utilice fl.documents para recuperar una matriz de todos los documentos abiertos; utilice `fl.documents[index]` para acceder a un determinado documento; utilice fl.getDocumentDOM() para acceder al documento actual (el que está seleccionado).
Objeto drawingLayer	Utilice fl.drawingLayer para acceder al objeto drawingLayer.
Objeto Effect	Utilice fl.effects para recuperar una matriz de descriptores de efectos que corresponda a los efectos registrados cuando se inicia Flash; utilice `fl.effects[index]` para acceder a un determinado efecto; utilice fl.activeEffect para acceder al descriptor de efectos del efecto que se está aplicando.
Objeto Math	Utilice fl.Math para acceder al objeto Math.
Objeto outputPanel	Utilice fl.outputPanel para acceder al objeto outputPanel. Este objeto corresponde al panel Salida en el entorno de edición de Flash.
Objeto Project	Utilice fl.getProject() para devolver un objeto Project para el proyecto abierto actualmente.
Objeto Tools	Utilice fl.tools para acceder a una matriz de objetos Tools.
Objeto XMLUI	Utilice fl.xmlui para acceder a un objeto Interfaz de usuario XML (XMLUI). El objeto XMLUI permite obtener y establecer las propiedades de un cuadro de diálogo XMLUI.

El objeto Document

Una propiedad importante del objeto Flash de nivel superior es la propiedad `fl.documents`. Consulte la propiedad fl.documents. La propiedad `fl.documents` contiene una matriz de objetos Document en la que cada uno representa uno de los archivos FLA abiertos actualmente en el entorno de edición. Las propiedades de cada objeto Document representan la mayoría de los elementos que puede contener un archivo FLA. Por tanto, gran parte del DOM se compone de objetos y propiedades secundarios del objeto Document. Para más información, consulte Objeto Document.

Para hacer referencia al primer documento abierto, por ejemplo, utilice la sentencia `flash.documents[0]` o `fl.documents[0]`. El primer documento es el primer documento de Flash que se abrió en la sesión actual en el entorno de edición. Cuando se cierra el primer documento que se abrió, se reducen los índices de los otros documentos abiertos.

Para buscar el índice de un determinado documento, utilice `flash.findDocumentIndex(nameOfDocument)` o `fl.findDocumentIndex(nameOfDocument)`. Véase fl.findDocumentIndex().

Para acceder al documento seleccionado actualmente, utilice la sentencia `flash.getDocumentDOM()` o `fl.getDocumentDOM()`. Véase fl.getDocumentDOM(). El segundo es la sintaxis empleada en la mayoría de los ejemplos de este documento.

Para buscar un determinado documento en la matriz `fl.documents`, repita a través de la matriz y pruebe en cada documento su propiedad `document.name`. Véase fl.documents y document.name.

El acceso a todos los objetos del DOM que no figuran en la tabla anterior (consulte "El modelo de objetos de documento de Flash" en la página 14) se realiza desde el objeto Document. Por ejemplo, para acceder a la biblioteca de un documento, se emplea la propiedad `document.library`, que recupera un objeto Library:

```
fl.getDocumentDOM().library
```

Para acceder a la matriz de elementos de la biblioteca, utilice la propiedad `library.items` ; cada elemento de la matriz es un objeto Item:

```
fl.getDocumentDOM().library.items
```

Para acceder a un determinado elemento de la biblioteca, deberá especificar un miembro de la matriz `library.items`:

```
fl.getDocumentDOM().library.items[0]
```

En otras palabras, el objeto Library es un elemento secundario del objeto Document, y el objeto Item es un elemento secundario del objeto Library. Para más información, consulte document.library, Objeto library, library.items y Objeto Item.

Especificación del destino de una acción

A menos que se indique lo contrario, los métodos afectan al enfoque o la selección actual. Por ejemplo, el script siguiente dobla el tamaño de la selección actual porque no se especifica ningún objeto concreto:

```
fl.getDocumentDOM().scaleSelection(2, 2);
```

En algunos casos conviene que una acción se realice específicamente sobre el elemento seleccionado actualmente en el documento de Flash. Para ello, utilice la matriz que contiene la propiedad `document.selection` (véase document.selection). El primer elemento de la matriz representa el elemento seleccionado actualmente, como se muestra en el ejemplo siguiente:

```
var accDescription = fl.getDocumentDOM().selection[0].description;
```

El script siguiente dobla el tamaño del primer elemento en el escenario almacenado en la matriz de elementos, en lugar de la selección actual:

```
var element =
  fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0];
if (element) {
  element.width = element.width*2;
  element.height = element.height*2;
}
```

También puede realizar acciones como establecer bucles a través de todos los elementos del escenario o incrementar el ancho y el alto con un valor determinado, como se muestra en el ejemplo siguiente:

```
var elementArray =
  fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements;
  for (var i=0; i < elementArray.length; i++) {
    var offset = 10;
    elementArray[i].width += offset;
    elementArray[i].height += offset;
  }
```

Resumen de la estructura del DOM

La lista siguiente muestra la estructura del DOM en forma esquemática. Los números al principio de cada línea representan el nivel del objeto. Por ejemplo, un objeto precedido de “03” es un elemento secundario del siguiente objeto “02” de nivel superior, que a su vez, es un elemento secundario del siguiente objeto “01” de nivel superior.

En algunos casos, un objeto está disponible especificando una propiedad de su objeto principal. Por ejemplo, la propiedad `document.timelines` contiene una matriz de objetos Timeline (véase document.timelines y Objeto Timeline). Estas propiedades se indican en el esquema siguiente.

Por último, algunos objetos son subclases de otros objetos, en lugar de elementos secundarios de otros objetos. Un objeto que es una subclase de otro objeto tiene métodos y/o propiedades propios además de los métodos y propiedades del otro objeto (la superclase). Las subclases comparten el mismo nivel en la jerarquía que su superclase. Por ejemplo, el objeto Item es una superclase del objeto BitmapItem (véase Objeto Item y Objeto BitmapItem). Estas relaciones se ilustran en el esquema siguiente:

01 Funciones y métodos de nivel superior

01 Objeto FLfile

01 Objeto Flash (fl)

  02 Objeto componentsPanel

  02 Objeto Document (matriz fl.documents)

    03 Objeto Filter

    03 Objeto Matrix

    03 Objeto Fill

    03 Objeto Stroke

    03 Objeto library

      04 Objeto Item (matriz library.items)

      04 Objeto BitmapItem (subclase del Objeto Item)

      04 Objeto folderItem (subclase del Objeto Item)

      04 Objeto fontItem (subclase del Objeto Item)

      04 Objeto SoundItem (subclase del Objeto Item)

      04 Objeto SymbolItem (subclase del Objeto Item)

      04 Objeto VideoItem (subclase del Objeto Item)

    03 Objeto Timeline (matriz document.timelines)

      04 Objeto Layer (matriz timeline.layers)

        05 Objeto Frame (matriz layer.frames)

          06 Objeto Element (matriz frame.elements)

            07 Objeto Matrix (Element.matrix)

          06 Objeto Instance (clase abstracta, subclase del Objeto Element)

          06 Objeto BitmapInstance (subclase del Objeto Instance)

          06 Objeto CompiledClipInstance (subclase del Objeto Instance)

          06 Objeto ComponentInstance (subclase del Objeto SymbolInstance)

07 Objeto Parameter (componentInstance.parameters)
06 Objeto SymbolInstance (subclase del Objeto Instance)
06 Objeto Text (subclase del Objeto Element)
07 Objeto TextRun (matriz text.textRuns)
08 Objeto TextAttrs (matriz textRun.textAttrs)
06 Objeto Shape (subclase del Objeto Element)
07 Objeto Contour (matriz shape.contours)
08 Objeto HalfEdge
09 Objeto Vertex
09 Objeto Edge
07 Objeto Edge (matriz shape.edges)
08 Objeto HalfEdge
09 Objeto Vertex
09 Objeto Edge
07 Objeto Vertex (matriz shape.vertices)
08 Objeto HalfEdge
09 Objeto Vertex
09 Objeto Edge
03 Objeto ScreenOutline
04 Objeto Screen (matriz screenOutline.screens)
05 Objeto Parameter (matriz screen.parameters)
02 Objeto drawingLayer
03 Objeto Path
04 Objeto Contour
02 Objeto Effect (matriz fl.effects)
02 Objeto Math
02 Objeto outputPanel
02 Objeto Project
03 Objeto ProjectItem (matriz project.items)
02 Objeto Tools (matriz fl.tools)
03 Objeto ToolObj (matriz tools.toolObjs)
02 Objeto XMLUI

Implementaciones de muestra

En Flash 8 se incluyen varias implementaciones de muestra de JSFL. Puede revisar e instalar estos archivos para familiarizarse con la API JavaScript. Estas muestras están instaladas en la carpeta Samples/ExtendingFlash dentro de la carpeta en la que se ha instalado Flash. Por ejemplo, si se instaló Flash con la configuración predeterminada, las muestras se incluyen en la siguiente ubicación:

- En Windows: *unidad de inicio*\Archivos de programa\Macromedia\Flash 8\Samples and Tutorials\Samples\ExtendingFlash
- En Macintosh: Macintosh HD/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/ExtendingFlash

Comando de muestra Shape

El script de muestra de la API JavaScript denominado Shape.jsfl se ubica en la carpeta ExtendingFlash/Shape (consulte "Implementaciones de muestra" más arriba). Este script muestra información sobre los contornos de la forma en el panel Salida.

Para instalar y ejecutar el script Shape:

1. Copie el archivo Shape.jsfl en la carpeta Configuration/Commands (consulte "Almacenamiento de archivos JSFL" en la página 7).
2. En un documento de Flash (archivo FLA), seleccione un objeto Shape.
3. Seleccione Comandos > Forma para ejecutar el script.

Comando de muestra para obtener y definir filtros

El script de muestra de la API JavaScript denominado GetSet.jsfl se ubica en la carpeta ExtendingFlash/filtersGetSet (consulte "Implementaciones de muestra" en la página 20). Este script añade filtros a un objeto seleccionado y muestra información sobre los filtros que se añaden en el panel Salida.

Para instalar y ejecutar el script filtersGetSet:

1. Copie el archivo filtersGetSet.jsfl en la carpeta Configuration/Commands (consulte "Almacenamiento de archivos JSFL" en la página 7).
2. En un documento de Flash (archivo FLA), seleccione un texto, clip de película u objeto Button.
3. Seleccione Comandos > filtersGetSet para ejecutar el script.

Herramienta de muestra PolyStar

Un script de muestra de la API JavaScript denominado PolyStar.jsfl se ubica en la carpeta ExtendingFlash/PolyStar (consulte "Implementaciones de muestra" en la página 20).

PolyStar.jsfl replica la herramienta PolyStar que se puede encontrar en el panel Herramientas de Flash. El script demuestra cómo crear la herramienta PolyStar con la API JavaScript e incluye comentarios detallados que describen lo que está haciendo el código. Lea este archivo para conocer mejor cómo funciona la API JavaScript. También se recomienda que lea el archivo PolyStar.xml en el directorio Tools para conocer mejor cómo crear su propia herramienta.

Flash incluye una versión anterior (ininteligible) del script PolyStar.jsfl que debe eliminar para poder utilizar el archivo de muestra PolyStar.jsfl.

Para eliminar la versión anterior del archivo PolyStar.jsfl que se instaló con Flash:

1. Seleccione Edición > Personalizar panel de herramientas (Windows) o Flash > Personalizar panel de herramientas (Macintosh).
2. En el cuadro de diálogo Personalizar panel de herramientas, haga clic en la herramienta Rectángulo, en la parte izquierda del cuadro de diálogo.

   La herramienta Rectángulo y la herramienta PolyStar deberán aparecer ahora en la lista Selección actual, en la parte derecha del cuadro de diálogo.
3. Seleccione la herramienta PolyStar en la lista Selección actual.
4. Haga clic en Quitar.
5. Haga clic en Aceptar.
6. Salga de Flash.
7. Elimine únicamente el archivo PolyStar.jsfl de la carpeta Configuration/Tools (consulte "Almacenamiento de archivos JSFL" en la página 7). Los archivos PolyStar.xml y PolyStar.png son necesarios para el nuevo archivo PolyStar.jsfl que instalará más adelante. Cuando reinicie Flash, la herramienta PolyStar ya no aparecerá en el cuadro de diálogo Personalizar panel de herramientas.

Para instalar los archivos PolyStar actualizados de ejemplo:

1. Si se está ejecutando Flash, salga de la aplicación.
2. Copie el nuevo archivo PolyStar.jsfl en la carpeta Configuration/Tools (consulte "Almacenamiento de archivos JSFL" en la página 7). El nuevo archivo PolyStar.jsfl necesita los archivos PolyStar.xml y PolyStar.png que se encuentran en esta carpeta.
3. Reinicie Flash.

4. Seleccione Edición > Personalizar panel de herramientas (Windows) o Flash > Personalizar panel de herramientas (Macintosh). La herramienta PolyStar deberá aparecer en la lista de herramientas disponibles.
5. Haga clic en la herramienta Rectángulo de la parte izquierda del cuadro de diálogo Personalizar panel de herramientas. La herramienta Rectángulo deberá aparecer en la lista Selección actual, en la parte derecha del cuadro de diálogo.
6. Seleccione la herramienta PolyStar en la lista Herramientas disponibles.
7. Haga clic en Añadir.
8. Haga clic en Aceptar.

   La herramienta PolyStar aparecerá ahora en el menú emergente de la herramienta Rectángulo.

Panel de muestra Trazar Mapa de Bits

Un conjunto de archivos denominado TraceBitmap.fla y TraceBitmap.swf se ubican en la carpeta ExtendingFlash/TraceBitmapPanel (consulte "Implementaciones de muestra" en la página 20). Estos archivos muestran cómo diseñar y crear un panel para controlar las funciones de Flash. También muestran el uso de la función `MMExecute()` para llamar a los comandos JavaScript desde un script de ActionScript.

Para ejecutar la muestra TraceBitmap:

1. Si se está ejecutando Flash, salga de la aplicación.
2. Copie el archivo TraceBitmap.swf en la carpeta Configuration/WindowSWF (consulte "Almacenamiento de archivos JSFL" en la página 7).
3. Inicie Flash.
4. Cree o abra un documento de Flash (archivo FLA) e importe una imagen de mapa de bits o JPEG al archivo.

   Puede utilizar el archivo flower.jpg incluido en la carpeta TraceBitmapPanel u otra imagen que elija.
5. Con la imagen importada seleccionada, elija Ventana > Otros paneles > Trazar Mapa de Bits.
6. Haga clic en Enviar.

   La imagen se convierte en un grupo de formas.

DLL de muestra

Una implementación de DLL de muestra se ubica en la carpeta ExtendingFlash/dllSampleComputeSum (consulte “Implementaciones de muestra” en la página 20). Para más información sobre la creación de DLL, consulte Capítulo 3, “Extensibilidad de nivel C”, en la página 543.

CAPÍTULO 1

1 Funciones y métodos de nivel superior

En este capítulo se describen las funciones y los métodos de nivel superior disponibles cuando se utiliza la interfaz de programación de aplicaciones JavaScript (API JavaScript) de Macromedia Flash. Para obtener información sobre dónde almacenar los archivos API JavaScript, consulte "Almacenamiento de archivos JSFL" en la página 7.

Las listas siguientes ofrecen un resumen de las áreas en el entorno de edición relacionadas con cada función o método. Después de las listas se presentan las funciones y los métodos en orden alfabético.

Métodos globales

Los métodos siguientes se pueden llamar desde cualquier script de la API JavaScript.

```
alert()
confirm()
prompt()
```

Efectos de línea de tiempo

Las funciones siguientes son específicas de los efectos de línea de tiempo:

```
configureEffect()
executeEffect()
removeEffect()
```

Herramientas ampliables

Las funciones siguientes están disponibles en scripts que crean herramientas ampliables:

```
activate()
configureTool()
deactivate()
keyDown()
keyUp()
mouseDoubleClick()
mouseDown()
```

```
mouseMove()
mouseUp()
notifySettingsChanged()
setCursor()
```

activate()

Disponibilidad

Flash MX 2004.

Uso

```
function activate() {
  // sentencias
}
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Función; se llama cuando se activa la herramienta ampliable (es decir, cuando se selecciona la herramienta en el panel Herramientas). Utilice esta función para realizar las tareas de inicialización que necesita la herramienta.

Ejemplo

El ejemplo siguiente establece el valor de `tools.activeTool` cuando se selecciona la herramienta ampliable en el panel Herramientas:

```
function activate() {
  var theTool = fl.tools.activeTool
}
```

Véase también

`tools.activeTool`

alert()

Disponibilidad

Flash MX 2004.

Uso

```
alert ( alertText )
```

Parámetros

alertText Una cadena que especifica el mensaje que desea mostrar en el cuadro de diálogo Alerta.

Valor devuelto

Ninguno.

Descripción

Método; muestra una cadena en un cuadro de diálogo modal Alerta, junto con un botón Aceptar.

Ejemplo

El ejemplo siguiente muestra el mensaje "Process Complete" en un cuadro de diálogo Alerta:

```
alert("Process Complete");
```

Véase también

`confirm()`, `prompt()`

configureEffect()

Disponibilidad

Flash MX 2004.

Uso

```
function configureEffect() {
  // Sentencias
}
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Función; se llama una vez cuando se carga Flash; coloque todas las sentencias de inicialización globales para su efecto dentro de esta función. Los datos de parámetro propios de una instancia para un efecto no son accesibles desde aquí.

Véase también

executeEffect(), removeEffect()

configureTool()

Disponibilidad

Flash MX 2004.

Uso

```
function configureTool() {
  // sentencias
}
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Función; se llama cuando se abre Flash y se carga la herramienta ampliable en el panel Herramientas. Utilice esta función para definir la información que Flash necesita conocer sobre la herramienta.

Ejemplo

Los ejemplos siguientes muestran dos implementaciones posibles de esta función:

```
function configureTool() {
  theTool = fl.tools.activeTool;
  theTool.setToolName("myTool");
  theTool.setIcon("myTool.png");
  theTool.setMenuString("My Tool's menu string");
  theTool.setToolTip("my tool's tool tip");
  theTool.setOptionsFile( "mtTool.xml" );
}

function configureTool() {
  theTool = fl.tools.activeTool;
  theTool.setToolName("ellipse");
  theTool.setIcon("Ellipse.png");
  theTool.setMenuString("Ellipse");
  theTool.setToolTip("Ellipse");
  theTool.showTransformHandles( true );
}
```

confirm()

Disponibilidad

Flash 8.

Uso

```
confirm ( strAlert )
```

Parámetros

strAlert Una cadena que especifica el mensaje que desea mostrar en el cuadro de diálogo Alerta.

Valor devuelto

Un valor booleano: `true` si el usuario hace clic en Aceptar, `false` si hace clic en Cancelar.

Descripción

Método; muestra una cadena en un cuadro de diálogo modal Alerta, junto con botones Aceptar y Cancelar.

Ejemplo

El ejemplo siguiente muestra el mensaje “Sort data?” en un cuadro de diálogo Alerta:

```
confirm("Sort data?");
```

Véase también

`alert()`, `prompt()`

deactivate()

Disponibilidad

Flash MX 2004.

Uso

```
function deactivate() {
  // sentencias
}
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Función; se llama cuando la herramienta ampliable se desactiva (es decir, cuando la herramienta activa cambia de esta herramienta a otra). Utilice esta función para realizar la limpieza que necesita la herramienta.

Ejemplo

El siguiente ejemplo muestra un mensaje en el panel Salida cuando la herramienta se vuelve inactiva:

```
function deactivate() {
  fl.trace( "Tool is no longer active" );
}
```

executeEffect()

Disponibilidad

Flash MX 2004.

Uso

```
function executeEffect() {
  // sentencias
}
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Función; se llama cuando el usuario aplica por primera vez un efecto o cambia las propiedades de un efecto. El código que contiene esta función modifica el objeto u objetos originales para crear el efecto deseado. También es responsable de copiar el original en una capa oculta si es necesario para la función `removeEffect`.

Véase también

`configureEffect()`, `removeEffect()`

keyDown()

Disponibilidad

Flash MX 2004.

Uso

```
function keyDown() {
  // sentencias
}
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Función; se llama cuando la herramienta ampliable está activa y el usuario presiona una tecla. El script debe llamar a `tools.getKeyDown()` para determinar qué tecla se ha presionado.

Ejemplo

El ejemplo siguiente muestra información sobre qué tecla se ha presionado cuando la herramienta ampliable está activa y el usuario presiona una tecla.

```
function keyDown() {
  fl.trace("key " + fl.tools.getKeyDown() + " was pressed");
}
```

Véase también

`keyUp()`, `tools.getKeyDown()`

keyUp()

Disponibilidad

Flash MX 2004.

Uso

```
function keyUp() {
  // sentencias
}
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Función; se llama cuando la herramienta ampliable está activa y se suelta una tecla.

Ejemplo

El ejemplo siguiente muestra un mensaje en el panel Salida cuando la herramienta ampliable está activa y se suelta una tecla.

```
function keyUp() {
  fl.trace("Key is released");
}
```

Véase también

keyDown()

mouseDoubleClick()

Disponibilidad

Flash MX 2004.

Uso

```
function mouseDoubleClick() {
  // sentencias
}
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Función; se llama cuando la herramienta ampliable está activa y se hace doble clic en el botón del ratón en el escenario.

Ejemplo

El ejemplo siguiente muestra un mensaje en el panel Salida cuando la herramienta ampliable está activa y se hace doble clic en el botón del ratón.

```
function mouseDoubleClick() {
  fl.trace("Mouse was double-clicked");
}
```

mouseDown()

Disponibilidad

Flash MX 2004.

Uso

```
function mouseDown( [ pt ] ) {
  // sentencias
}
```

Parámetros

pt Un punto que especifica la ubicación del ratón cuando se presiona el botón. Se transfiere a la función cuando se presiona el botón del ratón. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Función; se llama cuando la herramienta ampliable está activa y se presiona el botón del ratón mientras el puntero se encuentra sobre el escenario.

Ejemplo

Los siguientes ejemplos muestran cómo se puede emplear esta función cuando la herramienta ampliable está activa. El primer ejemplo muestra un mensaje en el panel Salida de que se ha presionado el botón del ratón. El segundo ejemplo muestra las coordenadas *x* e *y* de la ubicación del ratón cuando se presionó el botón.

```
function mouseDown() {
  fl.trace("Mouse button has been pressed");
}
function mouseDown(pt) {
  fl.trace("x = "+ pt.x+" :: y = "+pt.y);
}
```

mouseMove()

Disponibilidad

Flash MX 2004.

Uso

```
function mouseMove( [ pt ] ) {
  // sentencias
}
```

Parámetros

pt Un punto que especifica la ubicación actual del ratón. Se transfiere a la función cuando se mueve el ratón, realizando un seguimiento de la ubicación del ratón. Si el escenario se encuentra en modo de edición o de edición en contexto, las coordenadas del punto serán relativas al objeto que se está editando. En caso contrario, las coordenadas del punto serán relativas al escenario. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Función; se llama cada vez que la herramienta ampliable está activa y el ratón se mueve sobre un punto especificado en el escenario. El botón del ratón puede estar presionado o no presionado.

Ejemplo

Los ejemplos siguientes muestran el uso de esta función. El primer ejemplo muestra un mensaje en el panel Salida de que se está moviendo el ratón. El segundo ejemplo muestra las coordenadas *x* e *y* de la ubicación del ratón a medida que se mueve.

```
function mouseMove() {
  fl.trace("moving");
}

function mouseMove(pt) {
  fl.trace("x = "+ pt.x + " :: y = " + pt.y);
}
```

mouseUp()

Disponibilidad

Flash MX 2004.

Uso

```
function mouseUp() {
  // sentencias
}
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Función; se llama cuando la herramienta ampliable está activa y se suelta el botón del ratón después de presionarse en el escenario.

Ejemplo

El ejemplo siguiente muestra un mensaje en el panel Salida cuando la herramienta ampliable está activa y se suelta el botón del ratón.

```
function mouseUp() {
  fl.trace("mouse is up");
}
```

notifySettingsChanged()

Disponibilidad

Flash MX 2004.

Uso

```
function notifySettingsChanged() {
  // sentencias
}
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Función; se llama cuando la herramienta ampliable está activa y el usuario cambia sus opciones en el inspector de propiedades. Puede utilizar la propiedad `tools.activeTool` para consultar los valores actuales de las opciones (véase `tools.activeTool`).

Ejemplo

El ejemplo siguiente muestra un mensaje en el panel Salida cuando la herramienta ampliable está activa y el usuario cambia sus opciones en el inspector de propiedades.

```
function notifySettingsChanged() {
  var theTool = fl.tools.activeTool;
  var newValue = theTool.myProp;
}
```

prompt()

Disponibilidad

Flash MX 2004.

Uso

```
prompt( promptMsg, [ text ] )
```

Parámetros

promptMsg Una cadena que se mostrará en el cuadro de diálogo Mensaje (limitado a 256 caracteres en Mac OS X).

text Una cadena opcional que se mostrará como valor predeterminado para el campo de texto.

Valor devuelto

La cadena que el usuario haya escrito si éste hace clic en Aceptar; `null` si hace clic en Cancelar.

Descripción

Método; muestra un mensaje y texto opcional en un cuadro de diálogo modal Alerta, junto con botones Aceptar y Cancelar.

Ejemplo

El ejemplo siguiente pide al usuario que introduzca un nombre de usuario. Si el usuario escribe un nombre y hace clic en Aceptar, el nombre aparece en el panel Salida.

```
var userName = prompt("Enter user name", "Type user name here");
fl.trace(userName);
```

Véase también

`alert()`, `confirm()`

removeEffect()

Disponibilidad

Flash MX 2004.

Uso

```
function removeEffect() {
  // sentencias
}
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Función; se llama cuando el usuario cambia las propiedades de un efecto o utiliza el elemento de menú Quitar efecto. El código que contiene esta función restablece el estado original del objeto u objetos. Por ejemplo, si el efecto dividiese una cadena de texto, el método `removeEffect()` quitaría la cadena de texto que se ha separado y la reemplazaría por la cadena original.

Véase también

`configureEffect()`, `executeEffect()`

setCursor()

Disponibilidad

Flash MX 2004.

Uso

```
function setCursor() {
  // sentencias
}
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Función; se llama cuando la herramienta ampliable está activa y se mueve el ratón, para permitir que el script establezca punteros personalizados. El script debe llamar a `tools.setCursor()` para especificar el puntero que se va a utilizar. Para obtener una lista de qué punteros corresponden a qué valores enteros, consulte `tools.setCursor()`.

Ejemplo

```
function setCursor() {
  fl.tools.setCursor( 1 );
}
```

CAPÍTULO 2

Objetos

2

En este capítulo se describe brevemente cada uno de los objetos disponibles en la interfaz de programación de aplicaciones JavaScript (API JavaScript) de Flash. Los objetos figuran en orden alfabético en la tabla siguiente:

Objeto	Descripción
Objeto BitmapInstance	El objeto BitmapInstance es una subclase del Objeto Instance y representa un mapa de bits en un fotograma.
Objeto BitmapItem	Un objeto BitmapItem hace referencia a un mapa de bits en la biblioteca de un documento. El objeto BitmapItem es una subclase del Objeto Item.
Objeto CompiledClipInstance	El objeto CompiledClipInstance es una subclase del Objeto Instance.
Objeto ComponentInstance	El objeto ComponentInstance es una subclase del Objeto SymbolInstance y representa un componente en un fotograma.
Objeto componentsPanel	El objeto componentsPanel, que representa el panel Componentes, es una propiedad del Objeto Flash (fl) y `fl.componentsPanel` puede acceder a él.
Objeto Contour	Un objeto Contour representa una ruta cerrada de lados dirigidos en el límite de una forma.
Objeto Document	El objeto Document representa el escenario.
Objeto drawingLayer	Se puede acceder al objeto drawingLayer desde JavaScript como elemento secundario del objeto Flash.
Objeto Edge	El objeto Edge representa un borde de una forma en el escenario.
Objeto Effect	El objeto Effect representa una instancia de un efecto de línea de tiempo.
Objeto Element	Todo lo que aparece en el escenario es de tipo Element.

Objeto	Descripción
Objeto Fill	El objeto Fill contiene todas las propiedades de la configuración de Color de relleno del panel Herramientas o de una forma seleccionada.
Objeto Filter	El objeto Filter contiene todas las propiedades para todos los filtros.
Objeto Flash (fl)	El objeto Flash representa la aplicación Flash.
Objeto FLfile	El objeto FLfile permite escribir extensiones de Flash que pueden acceder, modificar y eliminar archivos y carpetas en el sistema de archivos local.
Objeto folderItem	El objeto folderItem es una subclase del Objeto Item.
Objeto fontItem	El objeto fontItem es una subclase del Objeto Item.
Objeto Frame	El objeto Frame representa fotogramas en la capa.
Objeto HalfEdge	Lado dirigido del borde de un Objeto Shape.
Objeto Instance	El objeto Instance es una subclase del Objeto Element.
Objeto Item	El objeto Item es una clase base abstracta.
Objeto Layer	El objeto Layer representa una capa en la línea de tiempo.
Objeto library	El objeto Library representa el panel Biblioteca.
Objeto Math	El objeto Math está disponible como propiedad de sólo lectura del objeto Flash; consulte `fl.Math`.
Objeto Matrix	El objeto Matrix representa una matriz de transformación.
Objeto outputPanel	El objeto outputPanel representa el panel Salida, que muestra información de resolución de problemas, como errores de sintaxis.
Objeto Parameter	El acceso al tipo de objeto Parameter se realiza desde la matriz `screen.parameters` (que corresponde al inspector de propiedades de la pantalla en la herramienta de edición de Flash) o la matriz `componentInstance.parameters` (que corresponde al inspector de propiedades del componente en la herramienta de edición).
Objeto Path	El objeto Path define una secuencia de segmentos de línea (recta, curva o ambas) que suele emplearse para crear herramientas ampliables.
Objeto Project	El objeto Project representa un archivo de proyecto de Flash (FLP).
Objeto ProjectItem	El objeto ProjectItem representa un elemento (archivo en el disco) que se ha añadido a un proyecto.

Objeto	Descripción
Objeto Screen	El objeto Screen representa una pantalla única en un documento de diapositivas o formularios.
Objeto ScreenOutline	El objeto ScreenOutline representa el grupo de pantallas en un documento de diapositivas o formularios.
Objeto Shape	El objeto Shape es una subclase del Objeto Element. El objeto Shape proporciona un control más preciso que las API de dibujo para manipular o crear geometría en el escenario.
Objeto SoundItem	El objeto SoundItem es una subclase del Objeto Item. Representa un elemento de biblioteca empleado para crear un sonido.
Objeto Stroke	El objeto Stroke contiene toda la configuración de un trazo, incluida la configuración personalizada.
Objeto SymbolInstance	El objeto SymbolInstance es una subclase del Objeto Instance y representa un símbolo en un fotograma.
Objeto SymbolItem	El objeto SymbolItem es una subclase del Objeto Item.
Objeto Text	El objeto Text representa un elemento de texto único en un documento.
Objeto TextAttrs	El objeto TextAttrs contiene todas las propiedades de texto que se pueden aplicar a una subselección. Este objeto es una subclase del Objeto Text.
Objeto TextRun	El objeto TextRun representa una serie de caracteres que tienen atributos que coinciden con todas las propiedades del Objeto TextAttrs.
Objeto Timeline	El objeto Timeline representa la línea de tiempo de Flash, a la que puede acceder mediante `fl.getDocumentDOM().getTimeline()` para el documento actual.
Objeto ToolObj	Un objeto ToolObj representa una herramienta individual en el panel Herramientas.
Objeto Tools	Se puede acceder al objeto Tools desde el objeto Flash (`fl.tools`).
Objeto Vertex	El objeto Vertex forma parte de la estructura de datos de formas que contiene los datos de coordenadas.
Objeto VideoItem	El objeto VideoItem es una subclase del Objeto Item.
Objeto XMLUI	El objeto XMLUI permite obtener y definir propiedades de un cuadro de diálogo XMLUI, así como aceptar o cancelar una.

Objeto BitmapInstance

Herencia Objeto Element > Objeto Instance > Objeto BitmapInstance

Disponibilidad

Flash MX 2004.

Descripción

El objeto BitmapInstance es una subclase del objeto Instance y representa un mapa de bits en un fotograma (véase Objeto Instance).

Resumen de métodos para el objeto BitmapInstance

Además de los métodos del Objeto Instance, puede emplear los métodos siguientes con el objeto BitmapInstance:

Método	Descripción
`bitmapInstance.getBits()`	Permite crear efectos de mapa de bits tomando los bits del mapa, manipulándolos y devolviéndolos a Flash.
`bitmapInstance.setBits()`	Define los bits de un elemento de mapa de bits existente.

Resumen de propiedades del objeto BitmapInstance

Además de las propiedades del Objeto Instance, puede emplear las propiedades siguientes con el objeto BitmapInstance:

Propiedad	Descripción
`bitmapInstance.hPixels`	De sólo lectura; un entero que representa el ancho del mapa de bits, en píxeles.
`bitmapInstance.vPixels`	De sólo lectura; un entero que representa el alto del mapa de bits, en píxeles.

bitmapInstance.getBits()

Disponibilidad

Flash MX 2004.

Uso

```
bitmapInstance.getBits()
```

Parámetros

Ninguno.

Valor devuelto

Un objeto que contiene las propiedades `width`, `height`, `depth`, `bits` y, si el mapa de bits tiene una tabla de colores, `cTab`. El elemento `bits` es una matriz de bytes. El elemento `cTab` es una matriz de valores de color con el formato `"#RRGGBB"`. La matriz tiene la misma longitud que la tabla de colores.

La matriz de bytes sólo tiene sentido cuando una DLL o biblioteca compartida hace referencia a ella. Sólo suele utilizarse para crear un efecto o una herramienta ampliable. Para más información sobre la creación de DLL para su uso con JavaScript de Flash, consulte el Capítulo 3, “Extensibilidad de nivel C”

Descripción

Método; permite crear efectos de mapa de bits tomando los bits del mapa, manipulándolos y devolviéndolos a Flash. Véase también `bitmapInstance.setBits()`.

Ejemplo

El código siguiente crea una referencia al objeto seleccionado actualmente; comprueba si el objeto es un mapa de bits y traza el alto, el ancho y la profundidad en bits del mapa de bits:

```
var isBitmap = fl.getDocumentDOM().selection[0].instanceType;
if(isBitmap == "bitmap"){
  var bits = fl.getDocumentDOM().selection[0].getBits();
  fl.trace("height = " + bits.height);
  fl.trace("width = " + bits.width);
  fl.trace("depth = " + bits.depth);
}
```

Véase también

`bitmapInstance.setBits()`

bitmapInstance.hPixels

Disponibilidad

Flash MX 2004.

Uso

```
bitmapInstance.hPixels
```

Descripción

Propiedad de sólo lectura; un entero que representa la anchura del mapa de bits, es decir, el número de píxeles en la dimensión horizontal.

Ejemplo

El código siguiente recupera el ancho del mapa de bits en píxeles:

```
// Obtiene el número de píxeles en la dimensión horizontal.
var bmObj = fl.getDocumentDOM().selection[0];
var isBitmap = bmObj.instanceType;
if(isBitmap == "bitmap"){
  var numHorizontalPixels = bmObj.hPixels;
}
```

Véase también

`bitmapInstance.vPixels`

bitmapInstance.setBits()

Disponibilidad

Flash MX 2004.

Uso

`bitmapInstance.setBits(`*`bitmap`*`)`

Parámetros

`bitmap` Un objeto que contiene las propiedades `height`, `width`, `depth`, `bits` y `cTab`. Las propiedades `height`, `width` y `depth` son enteros. La propiedad `bits` es una matriz de bytes. La propiedad `cTab` sólo es necesaria para mapas de bits con una profundidad en bits de 8 o menos y es una cadena que representa un valor de color con el formato `"#RRGGBB"`.

NOTA	La matriz de bytes sólo tiene sentido cuando una biblioteca externa hace referencia a ella. Sólo suele utilizarse para crear un efecto o una herramienta ampliable.

Valor devuelto

Ninguno.

Descripción

Método; define los bits de un elemento de mapa de bits existente. Permite crear efectos de mapa de bits tomando los bits del mapa, manipulándolos y devolviéndolos a Flash.

Ejemplo

El código siguiente comprueba si la selección actual es un mapa de bits y, a continuación, establece la altura del mapa de bits en 150 píxeles:

```
var isBitmap = fl.getDocumentDOM().selection[0].instanceType;
if(isBitmap == "bitmap"){
  var bits = fl.getDocumentDOM().selection[0].getBits();
  bits.height = 150;
  fl.getDocumentDOM().selection[0].setBits(bits);
}
```

Véase también

bitmapInstance.getBits()

bitmapInstance.vPixels

Disponibilidad

Flash MX 2004.

Uso

```
bitmapInstance.vPixels
```

Descripción

Propiedad de sólo lectura; un entero que representa la altura del mapa de bits, es decir, el número de píxeles en la dimensión vertical.

Ejemplo

El código siguiente obtiene el alto del mapa de bits en píxeles:

```
// Obtiene el número de píxeles en la dimensión vertical.
var bmObj = fl.getDocumentDOM().selection[0];
var isBitmap = bmObj.instanceType;
if(isBitmap == "bitmap"){
  var numVerticalPixels = bmObj.vPixels;
}
```

Véase también

bitmapInstance.hPixels

Objeto BitmapItem

Herencia Objeto Item > Objeto BitmapItem

Disponibilidad

Flash MX 2004.

Descripción

Un objeto BitmapItem hace referencia a un mapa de bits en la biblioteca de un documento. El objeto BitmapItem es una subclase del objeto Item (véase Objeto Item).

Resumen de propiedades del objeto BitmapItem

Además de las propiedades del Objeto Item, el objeto BitmapItem tiene las siguientes:

Propiedad	Descripción
`bitmapItem.allowSmoothing`	Un valor booleano que especifica si se permite o no el suavizado de un mapa de bits.
`bitmapItem.compressionType`	Una cadena que determina el tipo de compresión de imagen que se aplica al mapa de bits.
`bitmapItem.quality`	Un entero que especifica la calidad del mapa de bits.
`bitmapItem.useImportedJPEGQuality`	Un valor booleano que especifica si se utiliza o no la calidad JPEG importada predeterminada.

bitmapItem.allowSmoothing

Disponibilidad

Flash MX 2004.

Uso

```
bitmapItem.allowSmoothing
```

Descripción

Propiedad; un valor booleano que especifica si se permite el suavizado de un mapa de bits (`true`) o no (`false`).

Ejemplo

El código siguiente define la propiedad `allowSmoothing` del primer elemento de la biblioteca del documento actual como `true`:

```
fl.getDocumentDOM().library.items[0].allowSmoothing = true;
alert(fl.getDocumentDOM().library.items[0].allowSmoothing);
```

bitmapItem.compressionType

Disponibilidad

Flash MX 2004.

Uso

```
bitmapItem.compressionType
```

Descripción

Propiedad; una cadena que determina el tipo de compresión de imagen que se aplica al mapa de bits. Los valores aceptables son: `"photo"` o `"lossless"`. Si el valor de `bitmapItem.useImportedJPEGQuality` es `false`, `"photo"` corresponderá a JPEG con una calidad de 0 a 100; si `bitmapItem.useImportedJPEGQuality` es `true`, `"photo"` corresponderá a JPEG con un valor de calidad de documento predeterminada. El valor `"lossless"` corresponde al formato GIF o PNG. (Véase `bitmapItem.useImportedJPEGQuality`.)

Ejemplo

El código siguiente define la propiedad `compressionType` del primer elemento de la biblioteca del documento actual como `"photo"`:

```
fl.getDocumentDOM().library.items[0].compressionType = "photo";
alert(fl.getDocumentDOM().library.items[0].compressionType);
```

bitmapItem.quality

Disponibilidad

Flash MX 2004.

Uso

```
bitmapItem.quality
```

Descripción

Propiedad; un entero que especifica la calidad del mapa de bits. Para utilizar la calidad de documento predeterminada, especifique -1; en caso contrario, especifique un entero de 0 a 100. Sólo está disponible para compresión JPEG.

Ejemplo

El código siguiente define la propiedad `quality` del primer elemento de la biblioteca del documento actual como 65:

```
fl.getDocumentDOM().library.items[0].quality = 65;
alert(fl.getDocumentDOM().library.items[0].quality);
```

bitmapItem.useImportedJPEGQuality

Disponibilidad

Flash MX 2004.

Uso

```
bitmapItem.useImportedJPEGQuality
```

Descripción

Propiedad; un valor booleano que especifica si se utiliza la calidad JPEG importada predeterminada (`true`) o no (`false`). Sólo está disponible para compresión JPEG.

Ejemplo

El código siguiente define la propiedad `useImportedJPEGQuality` del primer elemento de la biblioteca del documento actual como `true`:

```
fl.getDocumentDOM().library.items[0].useImportedJPEGQuality = true;
alert(fl.getDocumentDOM().library.items[0].useImportedJPEGQuality);
```

Objeto CompiledClipInstance

Herencia Objeto Element > Objeto Instance > Objeto CompiledClipInstance

Disponibilidad

Flash MX 2004.

Descripción

El objeto CompiledClipInstance es una subclase del objeto Instance. Es fundamentalmente una instancia de un clip de película que se ha convertido en un elemento de biblioteca de clip compilado. (Véase Objeto Instance.)

Resumen de propiedades del objeto CompiledClipInstance

Además de las propiedades del Objeto Instance, el objeto CompiledClipInstance tiene las propiedades siguientes:

Propiedad	Descripción
`compiledClipInstance.accName`	Una cadena que equivale al campo Nombre del panel Accesibilidad.
`compiledClipInstance.actionScript`	Una cadena que representa el ActionScript para esta instancia; equivale a `symbolInstance.actionScript`.
`compiledClipInstance.description`	Una cadena que equivale al campo Descripción del panel Accesibilidad.
`compiledClipInstance.forceSimple`	Un valor booleano que activa y desactiva los elementos secundarios del objeto para que sea accesible.
`compiledClipInstance.shortcut`	Una cadena que equivale al campo Método abreviado del panel Accesibilidad.
`compiledClipInstance.silent`	Un valor booleano que activa o desactiva la accesibilidad del objeto; equivale a la lógica inversa de la opción Hacer que el objeto sea accesible del panel Accesibilidad.
`compiledClipInstance.tabIndex`	Un entero que equivale al campo Índice de fichas del panel Accesibilidad.

compiledClipInstance.accName

Disponibilidad

Flash MX 2004.

Uso

```
compiledClipInstance.accName
```

Descripción

Propiedad; una cadena que equivale al campo Nombre del panel Accesibilidad. Los lectores de pantalla identifican los objetos mediante la lectura del nombre en voz alta.

Ejemplo

El ejemplo siguiente obtiene y define el nombre de accesibilidad del primer objeto seleccionado:

```
// Obtiene el nombre del objeto.
var theName = fl.getDocumentDOM().selection[0].accName;
// Define el nombre del objeto.
fl.getDocumentDOM().selection[0].accName = 'Home Button';
```

compiledClipInstance.actionScript

Disponibilidad

Flash MX 2004.

Uso

```
compiledClipInstance.actionScript
```

Descripción

Propiedad; una cadena que representa el ActionScript para esta instancia; equivale a `symbolInstance.actionScript`.

Ejemplo

El código siguiente asigna ActionScript a los elementos especificados:

```
// Asigna ActionScript a una instancia de clip compilado de un botón
  especificado.
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0]
  .actionScript = "on(click) {trace('button is clicked');}";
// Asigna ActionScript a una instancia de clip compilado de un botón
  seleccionado.
fl.getDocumentDOM().selection[0].actionScript =
  "on(click) {trace('button is clicked');}";
```

compiledClipInstance.description

Disponibilidad

Flash MX 2004.

Uso

```
compiledClipInstance.description
```

Descripción

Propiedad; una cadena que equivale al campo Descripción del panel Accesibilidad. El lector de pantalla lee esta descripción.

Ejemplo

El ejemplo siguiente ilustra la obtención y definición de la propiedad `description`:

```
// Obtiene la descripción de la selección actual.
var theDescription = fl.getDocumentDOM().selection[0].description;
// Define la descripción de la selección actual.
fl.getDocumentDOM().selection[0].description =
  "This is compiled clip number 1";
```

compiledClipInstance.forceSimple

Disponibilidad

Flash MX 2004.

Uso

```
compiledClipInstance.forceSimple
```

Descripción

Propiedad; un valor booleano que activa y desactiva los elementos secundarios del objeto para que sea accesible. Equivale a la lógica inversa de la opción Hacer que los objetos secundarios sean accesibles del panel Accesibilidad. Si `forceSimple` es `true`, equivale a la opción desactivada Hacer que los objetos secundarios sean accesibles. Si `forceSimple` es `false`, equivale a la opción activada Hacer que los objetos secundarios sean accesibles.

Ejemplo

El ejemplo siguiente ilustra la obtención y definición de la propiedad `forceSimple`:

```
// Consulta si los elementos secundarios del objeto son accesibles.
var areChildrenAccessible = fl.getDocumentDOM().selection[0].forceSimple;
// Permite que los elementos secundarios del objeto sean accesibles.
fl.getDocumentDOM().selection[0].forceSimple = false;
```

compiledClipInstance.shortcut

Disponibilidad

Flash MX 2004.

Uso

```
compiledClipInstance.shortcut
```

Descripción

Propiedad; una cadena que equivale al campo Método abreviado del panel Accesibilidad. Los lectores de pantalla leen este método abreviado. Esta propiedad no está disponible para campos de texto dinámicos.

Ejemplo

El ejemplo siguiente ilustra la obtención y definición de la propiedad `shortcut`:

```
// Obtiene la tecla de método abreviado del objeto.
var theShortcut = fl.getDocumentDOM().selection[0].shortcut;
// Define la tecla de método abreviado del objeto.
fl.getDocumentDOM().selection[0].shortcut = "Ctrl+I";
```

compiledClipInstance.silent

Disponibilidad

Flash MX 2004.

Uso

```
compiledClipInstance.silent
```

Descripción

Propiedad; un valor booleano que activa o desactiva la accesibilidad del objeto; equivale a la lógica inversa de la opción Hacer que el objeto sea accesible del panel Accesibilidad. Es decir, si `silent` es `true`, estará desactivada la opción Hacer que el objeto sea accesible. Si `silent` es `false`, estará la activada la opción Hacer que el objeto sea accesible.

Ejemplo

El ejemplo siguiente ilustra la obtención y definición de la propiedad `silent`:

```
// Consulta si el objeto es accesible.
var isSilent =fl.getDocumentDOM().selection[0].silent;
// Define el objeto como accesible.
fl.getDocumentDOM().selection[0].silent = false;
```

compiledClipInstance.tabIndex

Disponibilidad

Flash MX 2004.

Uso

```
compiledClipInstance.tabIndex
```

Descripción

Propiedad; un entero que equivale al campo Índice de fichas del panel Accesibilidad. Crea un orden de tabulación con el que se accede a los objetos cuando el usuario presiona la tecla Tabulador.

Ejemplo

El ejemplo siguiente ilustra la obtención y definición de la propiedad `tabIndex`:

```
// Obtiene el tabIndex del objeto.
var theTabIndex = fl.getDocumentDOM().selection[0].tabIndex;
// Define el tabIndex del objeto.
fl.getDocumentDOM().sele ction[0].tabIndex = 1;
```

Objeto ComponentInstance

Herencia Objeto Element > Objeto Instance > Objeto SymbolInstance > Objeto ComponentInstance

Disponibilidad

Flash MX 2004.

Descripción

El objeto ComponentInstance es una subclase del objeto SymbolInstance y representa un componente en un fotograma. (Véase Objeto SymbolInstance.)

Resumen de propiedades del objeto ComponentInstance

Además de todas las propiedades del Objeto SymbolInstance, el objeto ComponentInstance tiene la propiedad siguiente:

Propiedad	Descripción
`componentInstance.parameters`	De sólo lectura; una matriz de las propiedades de ActionScript 2.0 que son accesibles desde el inspector de propiedades o de componentes.

componentInstance.parameters

Disponibilidad

Flash MX 2004.

Uso

```
componentInstance.parameters
```

Descripción

Propiedad de sólo lectura; una matriz de las propiedades de ActionScript 2.0 que son accesibles desde el inspector de propiedades o de componentes. Consulte "Objeto Parameter" en la página 344.

Ejemplo

El ejemplo siguiente ilustra la obtención y definición de la propiedad `parameters`:

```
var parms = fl.getDocumentDOM().selection[0].parameters;
parms[0].value = "some value";
```

Véase también

Objeto Parameter

Objeto componentsPanel

Disponibilidad

Flash MX 2004.

Descripción

El objeto componentsPanel, que representa el panel Componentes, es una propiedad del objeto Flash (fl) y `fl.componentsPanel` puede acceder a él. (Véase Objeto Flash (fl).)

Resumen de métodos del objeto componentsPanel

Puede emplear los métodos siguientes con el objeto componentsPanel:

Método	Descripción
`componentsPanel.addItemToDocument()`	Añade el componente especificado al documento en la posición especificada.
`componentsPanel.reload()`	Actualiza la lista de componentes del panel Componentes.

componentsPanel.addItemToDocument()

Disponibilidad

Flash MX 2004.

Uso

```
componentsPanel.addItemToDocument( position, categoryName, componentName )
```

Parámetros

`position` Un punto (por ejemplo, {`x`:0, `y`:`100`}) que especifica la ubicación en la que se añade el componente. Especifique *`position`* en relación con el punto central del componente, no el punto de registro del componente.

`categoryName` Una cadena que especifica el nombre de la categoría del componente (por ejemplo, `"Data"`). El panel Componentes muestra los nombres de categoría válidos.

`componentName` Una cadena que especifica el nombre del componente en la categoría especificada (por ejemplo, `"WebServiceConnector"`). El panel Componentes muestra los nombres de componente válidos.

Valor devuelto

Ninguno.

Descripción

Añade el componente especificado al documento en la posición especificada.

Ejemplos

El ejemplo siguiente ilustra algunas formas de utilizar este método:

```
fl.componentsPanel.addItemToDocument({x:0, y:0}, "User Interface",
  "CheckBox");
fl.componentsPanel.addItemToDocument({x:0, y:100}, "Data",
  "WebServiceConnector");
fl.componentsPanel.addItemToDocument({x:0, y:200}, "User Interface",
  "Button");
```

componentsPanel.reload()

Disponibilidad

Flash 8.

Uso

```
componentsPanel.reload()
```

Parámetros

Ninguno.

Valor devuelto

Un valor booleano de `true` si se actualiza la lista del panel Componentes, y de `false` en caso contrario.

Descripción

Método; actualiza la lista de componentes del panel Componentes.

Ejemplo

El ejemplo siguiente actualiza el panel Componentes:

```
fl.componentsPanel.reload();
```

Objeto Contour

Disponibilidad

Flash MX 2004.

Descripción

Un objeto Contour representa una ruta cerrada de lados dirigidos en el límite de una forma.

Resumen de métodos para el objeto Contour

Puede emplear el método siguiente con el objeto Contour:

Propiedad	Descripción
`contour.getHalfEdge()`	Devuelve un Objeto HalfEdge en el contorno de la selección.

Resumen de propiedades para el objeto Contour

Puede emplear las propiedades siguientes con el objeto Contour:

Propiedad	Descripción
`contour.interior`	De sólo lectura; el valor es `true` si el contorno encierra un área, y `false` en caso contrario.
`contour.orientation`	De sólo lectura; un entero que indica la orientación del contorno.

contour.getHalfEdge()

Disponibilidad

Flash MX 2004.

Uso

```
contour.getHalfEdge()
```

Parámetros

Ninguno.

Valor devuelto

Un Objeto HalfEdge.

Descripción

Método; devuelve un Objeto HalfEdge en el contorno de la selección.

Ejemplo

Este ejemplo atraviesa todos los contornos de una forma seleccionada y muestra las coordenadas de los vértices del panel Salida:

```
// con una forma seleccionada

var elt = fl.getDocumentDOM().selection[0];
elt.beginEdit();

var contourArray = elt.contours;
var contourCount = 0;
for (i=0;  i<contourArray.length;  i++)
{
  var contour = contourArray[i];
  contourCount++;
  var he = contour.getHalfEdge();

  var iStart = he.id;
  var id = 0;
  while (id != iStart)
  {
    // obtiene el siguiente vértice.
    var vrt = he.getVertex();


    var x = vrt.x;
    var y = vrt.y;
    fl.trace("vrt: " + x + ", " + y);

    he = he.getNext();
    id = he.id;
  }
}
elt.endEdit();
```

contour.interior

Disponibilidad

Flash MX 2004.

Uso

`contour.interior`

Descripción

Propiedad de sólo lectura; el valor es `true` si el contorno encierra un área y `false` en caso contrario.

Ejemplo

Este ejemplo atraviesa todos los contornos de la forma seleccionada y muestra el valor de la propiedad `interior` para cada contorno del panel Salida:

```
var elt = fl.getDocumentDOM().selection[0];
elt.beginEdit();

var contourArray = elt.contours;

var contourCount = 0;
for (i=0;  i<contourArray.length;  i++) {
  var contour = contourArray[i];
  fl.trace("Next Contour, interior:" + contour.interior );
  contourCount++;
}
elt.endEdit();
```

contour.orientation

Disponibilidad

Flash MX 2004.

Uso

```
contour.orientation
```

Descripción

Propiedad de sólo lectura; un entero que indica la orientación del contorno. El valor del entero es -1 si la orientación es en el sentido contrario a las agujas del reloj, 1 si es en el sentido de las agujas del reloj y 0 si es un contorno sin área.

Ejemplo

El ejemplo siguiente atraviesa todos los contornos de la forma seleccionada y muestra el valor de la propiedad `orientation` para cada contorno del panel Salida:

```
var elt = fl.getDocumentDOM().selection[0];
elt.beginEdit();

var contourArray = elt.contours;

var contourCount = 0;
for (i=0;  i<contourArray.length;  i++) {
  var contour = contourArray[i];
  fl.trace("Next Contour, orientation:" + contour.orientation);
  contourCount++;
}
elt.endEdit();
```

Objeto Document

Disponibilidad

Flash MX 2004.

Descripción

El objeto Document representa el escenario. Es decir, sólo se consideran documentos los archivos FLA.

Resumen de métodos para el objeto Document

Puede emplear los métodos siguientes con el objeto Document.

Método	Descripción
`document.addDataToDocument()`	Almacena datos especificados con un documento.
`document.addDataToSelection()`	Almacena datos especificados con el objeto u objetos seleccionados.
`document.addFilter()`	Aplica un filtro a los objetos seleccionados.
`document.addItem()`	Añade un elemento desde cualquier documento o biblioteca abierta al objeto Document especificado.
`document.addNewLine()`	Añade una nueva ruta entre dos puntos.
`document.addNewOval()`	Añade un nuevo óvalo en el rectángulo de delimitación especificado.
`document.addNewPublishProfile()`	Añade un nuevo perfil de publicación y lo convierte en el actual.
`document.addNewRectangle()`	Añade un nuevo rectángulo o rectángulo redondeado, ajustándolo a los límites especificados.
`document.addNewScene()`	Añade una nueva escena (Objeto Timeline) después de la escena seleccionada y la convierte en la seleccionada actualmente.
`document.addNewText()`	Inserta un nuevo campo de texto vacío.
`document.align()`	Alinea la selección.
`document.allowScreens()`	Utilice este método antes de usar la propiedad `document.screenOutline`.
`document.arrange()`	Organiza la selección en el escenario.

Método	Descripción
`document.breakApart()`	Realiza una operación de separación en la selección actual.
`document.canEditSymbol()`	Indica si el menú Editar símbolos y las funciones están activados.
`document.canRevert()`	Determina si puede utilizar correctamente el método `document.revert()` o `fl.revertDocument()`.
`document.canTestMovie()`	Determina si puede utilizar correctamente el método `document.testMovie()`.
`document.canTestScene()`	Determina si puede utilizar correctamente el método `document.testScene()`.
`document.changeFilterOrder()`	Cambia el índice del filtro de la lista Filtro.
`document.clipCopy()`	Copia la selección actual desde el documento hasta el Portapapeles.
`document.clipCut()`	Corta la selección actual del documento y la escribe en el Portapapeles.
`document.clipPaste()`	Pega el contenido del Portapapeles en el documento.
`document.close()`	Cierra el documento especificado.
`document.convertLinesToFills()`	Convierte líneas en rellenos en los objetos seleccionados.
`document.convertToSymbol()`	Convierte el elemento o elemento de escenario seleccionados en un símbolo nuevo.
`document.crop()`	Utiliza el objeto de dibujo seleccionado en la parte superior para recortar todos los objetos de dibujo seleccionados por debajo.
`document.deleteEnvelope()`	Elimina la envoltura (recuadro de delimitación que contiene uno o varios objetos) del objeto seleccionado.
`document.deletePublishProfile()`	Elimina el perfil activo actualmente, si hay más de uno.
`document.deleteScene()`	Elimina la escena actual (Objeto Timeline) y, si la escena eliminada no era la última, establece la siguiente escena como el objeto Timeline actual.
`document.deleteSelection()`	Elimina la selección actual en el escenario.
`document.disableAllFilters()`	Desactiva todos los filtros de los objetos seleccionados.

Método	Descripción
`document.disableFilter()`	Desactiva el filtro especificado en la lista Filtros.
`document.disableOtherFilters()`	Desactiva todos los filtros salvo el que se encuentra en la posición especificada en la lista Filtros.
`document.distribute()`	Distribuye la selección.
`document.distributeToLayers()`	Realiza una operación de distribución en capas en la selección actual; equivale a seleccionar Distribuir en capas.
`document.documentHasData()`	Comprueba si el documento contiene datos persistentes con el nombre especificado.
`document.duplicatePublishProfile()`	Duplica el perfil activo y selecciona la versión duplicada.
`document.duplicateScene()`	Realiza una copia de la escena seleccionada, asignando un nombre exclusivo a la nueva escena y convirtiéndola en la actual.
`document.duplicateSelection()`	Duplica la selección en el escenario.
`document.editScene()`	Convierte la escena especificada en la escena seleccionada actualmente para editar.
`document.enableAllFilters()`	Activa todos los filtros de la lista Filtros para el objeto u objetos seleccionados.
`document.enableFilter()`	Activa el filtro especificado para el objeto u objetos seleccionados.
`document.enterEditMode()`	Cambia la herramienta de edición al modo de edición especificado por el parámetro.
`document.exitEditMode()`	Sale del modo de edición de símbolos y vuelve a seleccionar el siguiente nivel superior desde el modo de edición.
`document.exportPNG()`	Exporta el documento como uno o varios archivos PNG.
`document.exportPublishProfile()`	Exporta el perfil activo actualmente a un archivo XML.
`document.exportSWF()`	Exporta el documento en formato SWC de Flash.
`document.getAlignToDocument()`	Equivale a recuperar el valor del botón En escenario en el panel Alinear.
`document.getBlendMode()`	Devuelve una cadena que especifica el modo de mezcla para el objeto u objetos seleccionados.

Método	Descripción
`document.getCustomFill()`	Recupera el objeto de relleno de la forma seleccionada o, si se especifica, del panel Herramientas y del inspector de propiedades.
`document.getCustomStroke()`	Devuelve el objeto de trazo de la forma seleccionada o, si se especifica, del panel Herramientas y del inspector de propiedades.
`document.getDataFromDocument()`	Recupera el valor de los datos especificados.
`document.getElementProperty()`	Obtiene la propiedad `Element` especificada para la selección actual.
`document.getElementTextAttr()`	Obtiene la propiedad `TextAttrs` especificada de los objetos de texto seleccionados.
`document.getFilters()`	Devuelve una matriz que contiene la lista de filtros aplicados al objeto u objetos seleccionados actualmente.
`document.getMetadata()`	Devuelve una cadena que contiene los metadatos XML asociados al documento.
`document.getSelectionRect()`	Obtiene el rectángulo de delimitación de la selección actual.
`document.getTextString()`	Obtiene el texto seleccionado actualmente.
`document.getTimeline()`	Recupera el Objeto Timeline actual en el documento.
`document.getTransformationPoint()`	Obtiene la ubicación del punto de transformación de la selección actual.
`document.group()`	Convierte la selección actual en un grupo.
`document.importPublishProfile()`	Importa un perfil desde un archivo.
`document.importFile()`	Importa un archivo al documento.
`document.importSWF()`	Importa un archivo SWF en el documento.
`document.intersect()`	Crea un objeto de dibujo de intersección a partir de todos los objetos de dibujo seleccionados.
`document.match()`	Iguala el tamaño de los objetos seleccionados.
`document.mouseClick()`	Ejecuta un clic de ratón desde la herramienta Flecha.
`document.mouseDblClk()`	Ejecuta un doble clic de ratón desde la herramienta Flecha.

Método	Descripción
`document.moveSelectedBezierPointsBy()`	Si la selección contiene como mínimo una ruta con al menos un punto Bézier seleccionado, este método mueve todos los puntos Bézier seleccionados en todas las rutas seleccionadas con la cantidad especificada.
`document.moveSelectionBy()`	Mueve los objetos seleccionados una distancia especificada.
`document.optimizeCurves()`	Optimiza el suavizado de la selección actual, permitiendo múltiples pasadas, si se especifica, para un suavizado óptimo; equivale a seleccionar Modificar › Forma› Optimizar.
`document.publish()`	Publica el documento de acuerdo con la configuración de publicación actual (Archivo › Configuración de publicación); equivale a seleccionar Archivo › Publicar.
`document.punch()`	Utiliza el objeto de dibujo seleccionado en la parte superior para perforar todos los objetos de dibujo seleccionados por debajo.
`document.removeAllFilters()`	Elimina todos los filtros del objeto u objetos seleccionados.‹
`document.removeDataFromDocument()`	Elimina datos persistentes con el nombre especificado que se han asociado al documento.
`document.removeDataFromSelection()`	Elimina datos persistentes con el nombre especificado que se han asociado a la selección.
`document.removeFilter()`	Elimina el filtro especificado de la lista Filtros del objeto u objetos seleccionados.
`document.renamePublishProfile()`	Cambia el nombre del perfil actual.
`document.renameScene()`	Cambia el nombre de la escena seleccionada actualmente en el panel Escenas.
`document.reorderScene()`	Mueve la escena especificada delante de otra escena especificada.
`document.resetTransformation()`	Restablece la matriz de transformación; equivale a seleccionar Modificar › Transformar › Quitar transformación.
`document.revert()`	Devuelve el documento especificado a su versión guardada anterior; equivale a seleccionar Archivo › Descartar cambios.

Método	Descripción
`document.rotateSelection()`	Gira la selección el número de grados especificado.
`document.save()`	Guarda el documento en su ubicación predeterminada; equivale a seleccionar Archivo > Guardar.
`document.saveAndCompact()`	Guarda y compacta el archivo; equivale a seleccionar Archivo > Guardar y compactar.
`document.scaleSelection()`	Escala la selección en la cantidad especificada; equivale a utilizar la herramienta Transformación libre para escalar el objeto.
`document.selectAll()`	Selecciona todos los elementos del escenario; equivale a presionar Control+A (Windows) o Command+A (Macintosh) o a seleccionar Edición > Seleccionar todo.
`document.selectNone()`	Desactiva la selección de los elementos seleccionados.
`document.setAlignToDocument()`	Establece las preferencias de `document.align()`, `document.distribute()`,`document.match()` y `document.space()` para que actúen sobre el documento; equivale a activar el botón En escenario en el panel Alinear.
`document.setBlendMode()`	Establece el modo de mezcla para los objetos seleccionados.
`document.setCustomFill()`	Establece la configuración de relleno para el panel Herramientas, el inspector de propiedades y cualquier forma seleccionada.
`document.setCustomStroke()`	Establece la configuración de trazo para el panel Herramientas, el inspector de propiedades y cualquier forma seleccionada.
`document.setElementProperty()`	Establece la propiedad `Element` especificada en el objeto u objetos seleccionados en el documento.
`document.setElementTextAttr()`	Establece la propiedad `TextAttrs` especificada de los elementos de texto seleccionados con el valor especificado.
`document.setFillColor()`	Cambia el color de relleno de la selección al especificado.
`document.setFilterProperty()`	Establece una propiedad de filtro especificada para el objeto u objetos seleccionados actualmente.

Método	Descripción
`document.setFilters()`	Aplica filtros a los objetos seleccionados.
`document.setInstanceAlpha()`	Establece la opacidad de la instancia.
`document.setInstanceBrightness()`	Establece el brillo de la instancia.
`document.setInstanceTint()`	Establece la tinta de la instancia.
`document.setMetadata()`	Establece los metadatos XML para el documento especificado, sobrescribiendo los metadatos existentes.
`document.setSelectionBounds()`	Mueve y cambia el tamaño de la selección en una única operación.
`document.setSelectionRect()`	Dibuja un recuadro de delimitación rectangular en relación con el escenario, empleando las coordenadas especificadas.
`document.setStroke()`	Establece el color, el ancho y el estilo de los trazos seleccionados.
`document.setStrokeColor()`	Cambia el color de trazo de la selección al especificado.
`document.setStrokeSize()`	Cambia el tamaño de trazo de la selección al especificado.
`document.setStrokeStyle()`	Cambia el estilo de trazo de la selección al especificado.
`document.setTextRectangle()`	Cambia el rectángulo de delimitación para el elemento de texto seleccionado al tamaño especificado.
`document.setTextSelection()`	Establece la selección de texto del campo de texto seleccionado actualmente con los valores especificados por los valores *`startIndex`* y *`endIndex`*.
`document.setTextString()`	Inserta una cadena de texto.
`document.setTransformationPoint()`	Mueve el punto de transformación de la selección actual.
`document.skewSelection()`	Sesga la selección en la cantidad especificada.
`document.smoothSelection()`	Suaviza la curva de cada línea curva o contorno de relleno seleccionado.
`document.space()`	Distribuye los objetos de la selección de manera uniforme.

Método	Descripción
`document.straightenSelection()`	Endereza los trazos seleccionados actualmente; equivale a utilizar el botón Enderezar del panel Herramientas.
`document.swapElement()`	Cambia la selección actual por la especificada.
`document.swapStrokeAndFill()`	Intercambia los colores de Trazo y Relleno.
`document.testMovie()`	Ejecuta una operación Probar documento en el documento.
`document.testScene()`	Ejecuta una operación Probar escena en la escena actual del documento.
`document.traceBitmap()`	Realiza una operación Trazar mapa de bits en la selección actual; equivale a seleccionar Modificar › Mapa de bits › Trazar mapa de bits.
`document.transformSelection()`	Realiza una transformación general en la selección actual aplicando la matriz especificada en los argumentos.
`document.unGroup()`	Desagrupa la selección actual.
`document.union()`	Combina todas las formas seleccionadas en un objeto de dibujo.
`document.unlockAllElements()`	Desbloquea todos los elementos bloqueados en el fotograma seleccionado actualmente.
`document.xmlPanel()`	Envía un cuadro de diálogo XMLUI.

Resumen de propiedades del objeto Document

Puede emplear las propiedades siguientes con el objeto Document.

Propiedad	Descripción
`document.accName`	Una cadena que equivale al campo Nombre del panel Accesibilidad.
`document.autoLabel`	Un valor booleano que equivale a la casilla de verificación Etiquetado automático del panel Accesibilidad.
`document.backgroundColor`	Una cadena, valor hexadecimal o entero que representa el color de fondo.
`document.currentPublishProfile`	Una cadena que especifica el nombre del perfil de publicación activo para el documento especificado.

Propiedad	Descripción
`document.currentTimeline`	Un entero que especifica el índice de la línea de tiempo activa.
`document.description`	Una cadena que equivale al campo Descripción del panel Accesibilidad.
`document.forceSimple`	Un valor booleano que especifica si los elementos secundarios del objeto especificado son accesibles.
`document.frameRate`	Un valor flotante que especifica el número de fotogramas mostrados por segundo cuando se reproduce el archivo SWF; el valor predeterminado es 12.
`document.height`	Un entero que especifica el alto del documento (escenario) en píxeles.
`document.library`	De sólo lectura; el Objeto library para un documento.
`document.livePreview`	Un valor booleano que especifica si está activada la opción Vista previa dinámica.
`document.name`	De sólo lectura; una cadena que representa el nombre de un documento (archivo FLA).
`document.path`	De sólo lectura; una cadena que representa la ruta del documento.
`document.publishProfiles`	De sólo lectura; una matriz de los nombres del perfil de publicación para el documento.
`document.screenOutline`	De sólo lectura; el Objeto ScreenOutline actual para el documento.
`document.selection`	Una matriz de los objetos seleccionados en el documento.
`document.silent`	Un valor booleano que especifica si el objeto es accesible.
`document.timelines`	De sólo lectura; una matriz de objetos Timeline (véase Objeto Timeline).
`document.viewMatrix`	De sólo lectura; un Objeto Matrix.
`document.width`	Un entero que especifica el ancho del documento (escenario) en píxeles.
`document.zoomFactor`	Especifica el porcentaje de zoom del escenario en tiempo de edición.

document.accName

Disponibilidad

Flash MX 2004.

Uso

```
document.accName
```

Descripción

Propiedad; una cadena que equivale al campo Nombre del panel Accesibilidad. Los lectores de pantalla identifican los objetos mediante la lectura del nombre en voz alta.

Ejemplo

El ejemplo siguiente establece el nombre de accesibilidad del documento como `"Main Movie"`:

```
fl.getDocumentDOM().accName = "Main Movie";
```

El ejemplo siguiente obtiene el nombre de accesibilidad del documento:

```
fl.trace(fl.getDocumentDOM().accName);
```

document.addDataToDocument()

Disponibilidad

Flash MX 2004.

Uso

```
document.addDataToDocument( name, type, data )
```

Parámetros

name Una cadena que especifica el nombre de los datos que se van a añadir.

type Una cadena que define el tipo de datos que se van a añadir. Los valores aceptables de *type* son: `"integer"`, `"integerArray"`, `"double"`, `"doubleArray"`, `"string"` y `"byteArray"`.

data El valor que se va a añadir. Los tipos válidos dependen del parámetro *type*.

Valor devuelto

Ninguno.

Descripción

Método; almacena datos especificados con un documento. Los datos se escriben en el archivo FLA y están disponibles en JavaScript cuando se vuelve a abrir el archivo.

Ejemplo

El ejemplo siguiente añade un valor entero de `12` al documento actual:

```
fl.getDocumentDOM().addDataToDocument("myData", "integer", 12);
```

El ejemplo siguiente devuelve el valor de los datos con el nombre `"myData"` y muestra el resultado en el panel Salida:

```
fl.trace(fl.getDocumentDOM().getDataFromDocument("myData"));
```

Véase también

`document.getDataFromDocument()`, `document.removeDataFromDocument()`

document.addDataToSelection()

Disponibilidad

Flash MX 2004.

Uso

```
document.addDataToSelection( name, type, data )
```

Parámetros

`name` Una cadena que especifica el nombre de los datos persistentes.

`type` Define el tipo de datos. Los valores aceptables son: `"integer"`, `"integerArray"`, `"double"`, `"doubleArray"`, `"string"` y `"byteArray"`.

`data` El valor que se va a añadir. Los tipos válidos dependen del parámetro *`type`*.

Valor devuelto

Ninguno.

Descripción

Método; almacena datos especificados con el objeto u objetos seleccionados. Los datos se escriben en el archivo FLA y están disponibles en JavaScript cuando se vuelve a abrir el archivo. Sólo los símbolos y mapas de bits admiten datos persistentes.

Ejemplo

El ejemplo siguiente añade un valor entero de 12 al objeto seleccionado:

```
fl.getDocumentDOM().addDataToSelection("myData", "integer", 12);
```

Véase también

`document.removeDataFromSelection()`

document.addFilter()

Disponibilidad

Flash 8.

Uso

```
document.addFilter( filterName )
```

Parámetros

`filterName` Una cadena que especifica el filtro que se va a añadir a la lista Filtro y que se activará para el objeto u objetos seleccionados. Los valores aceptables son: `"adjustColorFilter"`, `"bevelFilter"`, `"blurFilter"`, `"dropShadowFilter"`, `"glowFilter"`, `"gradientBevelFilter"` y `"gradientGlowFilter"`.

Valor devuelto

Ninguno.

Descripción

Método; aplica un filtro a los objetos seleccionados y coloca el filtro al final de la lista Filtro.

Ejemplo

El ejemplo siguiente aplica un filtro de iluminado al objeto u objetos seleccionados:

```
fl.getDocumentDOM().addFilter("glowFilter");
```

Véase también

`document.changeFilterOrder()`, `document.disableFilter()`, `document.enableFilter()`, `document.getFilters()`, `document.removeFilter()`, `document.setBlendMode()`, `document.setFilterProperty()`

document.addItem()

Disponibilidad

Flash MX 2004.

Uso

```
document.addItem( position, item )
```

Parámetros

position Un punto que especifica las coordenadas *x* e *y* de la ubicación en la que se desea añadir el elemento. Utiliza el centro de un símbolo o la esquina superior izquierda de un mapa de bits o de un vídeo.

item Un objeto Item que especifica el elemento que se va a añadir y la biblioteca desde la que se va a añadir (véase Objeto Item).

Valor devuelto

Un valor booleano: `true` si es correcto y `false` en caso contrario.

Descripción

Método; añade un elemento desde cualquier documento o biblioteca abierta al objeto Document especificado.

Ejemplo

El ejemplo siguiente añade el primer elemento de la biblioteca al primer documento en la ubicación especificada para el símbolo, mapa de bits o vídeo seleccionado.

```
var item = fl.documents[0].library.items[0];
fl.documents[0].addItem({x:0,y:0}, item);
```

El ejemplo siguiente añade el símbolo `myMovieClip` desde la biblioteca del documento actual hasta el documento actual:

```
var itemIndex = fl.getDocumentDOM().library.findItemIndex("myMovieClip");
var theItem = fl.getDocumentDOM().library.items[itemIndex];
fl.getDocumentDOM().addItem({x:0,y:0}, theItem);
```

El ejemplo siguiente añade el símbolo `myMovieClip` desde el segundo documento de la matriz de documentos hasta el tercer documento de dicha matriz:

```
var itemIndex = fl.documents[1].library.findItemIndex("myMovieClip");
var theItem = fl.documents[1].library.items[itemIndex];
fl.documents[2].addItem({x:0,y:0}, theItem);
```

document.addNewLine()

Disponibilidad

Flash MX 2004.

Uso

```
document.addNewLine( startPoint, endpoint )
```

Parámetros

`startpoint` Un par de números de coma flotante que especifican las coordenadas *x* e *y* donde comienza la línea.

`endpoint` Un par de números de coma flotante que especifican las coordenadas *x* e *y* donde termina la línea.

Valor devuelto

Ninguno.

Descripción

Método; añade una nueva ruta entre dos puntos. El método utiliza los atributos de trazo actuales del documento y añade la ruta en el fotograma y la capa actuales. Este método equivale a hacer clic en la herramienta Línea y dibujar una línea.

Ejemplo

El ejemplo siguiente añade una línea entre el punto de partida y el punto final especificados:

```
fl.getDocumentDOM().addNewLine({x:216.7, y:122.3}, {x:366.8, y:165.8});
```

document.addNewOval()

Disponibilidad

Flash MX 2004.

Uso

```
document.addNewOval( boundingRectangle [, bSuppressFill [, bSuppressStroke
  ]] )
```

Parámetros

boundingRectangle Un rectángulo que especifica los límites del óvalo que desea añadir. Para más información sobre el formato de *boundingRectangle*, consulte `document.addNewRectangle()`.

bSuppressFill Un valor booleano que, si se define como `true`, hace que el método cree la forma sin relleno. El valor predeterminado es `false`. Este parámetro es opcional.

bSuppressStroke Un valor booleano que, si se define como `true`, hace que el método cree la forma sin trazo. El valor predeterminado es `false`. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; añade un nuevo óvalo en el rectángulo de delimitación especificado. Este método realiza la misma operación que la herramienta Óvalo. El método utiliza los atributos predeterminados de trazo y relleno del documento y añade el óvalo en el fotograma y la capa actuales. Si *bSuppressFill* se define como `true`, el óvalo se dibuja sin relleno. Si *bSuppressStroke* se define como `true`, el óvalo se dibuja sin trazo. Si tanto *bSuppressFill* como *bSuppressStroke* se definen como `true`, el método no tiene ningún efecto.

Ejemplo

El ejemplo siguiente añade un nuevo óvalo dentro de las coordenadas especificadas; 164 píxeles de ancho por 178 píxeles de alto:

```
flash.getDocumentDOM().addNewOval({left:72,top:50,right:236,bottom:228});
```

El ejemplo siguiente dibuja el óvalo sin relleno:

```
flash.getDocumentDOM().addNewOval({left:72,top:50,right:236,bottom:228},
  true);
```

El ejemplo siguiente dibuja el óvalo sin trazo:

```
flash.getDocumentDOM().addNewOval({left:72,top:50,right:236,bottom:228},
  false, true);
```

document.addNewPublishProfile()

Disponibilidad

Flash MX 2004.

Uso

```
document.addNewPublishProfile( [profileName ] )
```

Parámetros

profileName El nombre exclusivo del nuevo perfil. Si no especifica un nombre, se suministrará un nombre predeterminado. Este parámetro es opcional.

Valor devuelto

Un entero que es el índice del nuevo perfil en la lista de perfiles. Devuelve -1 si no se puede crear un perfil nuevo.

Descripción

Método; añade un nuevo perfil de publicación y lo convierte en el actual.

Ejemplo

El ejemplo siguiente añade un nuevo perfil de publicación con un nombre predeterminado y, a continuación, muestra el nombre del perfil en el panel Salida:

```
fl.getDocumentDOM().addNewPublishProfile();
fl.outputPanel.trace(fl.getDocumentDOM().currentPublishProfile);
```

El ejemplo siguiente añade un nuevo perfil de publicación con el nombre "my profile":

```
fl.getDocumentDOM().addNewPublishProfile("my profile");
```

Véase también

`document.deletePublishProfile()`

document.addNewRectangle()

Disponibilidad

Flash MX 2004.

Uso

```
document.addNewRectangle( boundingRectangle, roundness
  [, bSuppressFill[, bSuppressStroke ] ] )
```

Parámetros

boundingRectangle Un rectángulo que especifica los límites dentro de los cuales se añade el nuevo rectángulo, con el formato `{left:value1,top:value2,right:value3,bottom:value4}`. Los valores `left` y `top` especifican la ubicación de la esquina superior izquierda (por ej., `left:0,top:0` representa la parte superior izquierda del escenario), y los valores `right` y `bottom` especifican la ubicación de la esquina inferior derecha. Por tanto, la anchura del rectángulo es la diferencia de valor entre `left` y `right`, mientras que la altura es la diferencia entre `top` y `bottom`.

En otras palabras, no todos los límites del rectángulo corresponden a los valores mostrados en el inspector de propiedades. Los valores `left` y `top` corresponden a los valores X e Y del inspector de propiedades, respectivamente. Sin embargo, los valores `right` y `bottom` no corresponden a los valores de anchura y altura del inspector de propiedades. Por ejemplo, considere un rectángulo con los siguientes límites:

```
{left:10,top:10,right:50,bottom:100}
```

Este rectángulo mostraría los siguientes valores en el inspector de propiedades:

```
X = 10, Y = 10, An = 40, Al = 90
```

roundness Un valor entero de 0 a 999 que especifica la redondez que se va a utilizar para las esquinas. El valor se expresa como número de puntos. Cuanto mayor sea el valor, mayor será la redondez.

bSuppressFill Un valor booleano que, si se define como `true`, hace que el método cree la forma sin relleno. El valor predeterminado es `false`. Este parámetro es opcional.

bSuppressStroke Un valor booleano que, si se define como `true`, hace que el método cree el rectángulo sin trazo. El valor predeterminado es `false`. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; añade un nuevo rectángulo o rectángulo redondeado, ajustándolo a los límites especificados. Este método realiza la misma operación que la herramienta Rectángulo. El método utiliza los atributos predeterminados de trazo y relleno del documento y añade el rectángulo en el fotograma y la capa actuales. Si el parámetro *bSuppressFill* se define como `true`, el rectángulo se dibuja sin relleno. Si el parámetro *bSuppressStroke* se define como `true`, el rectángulo se dibuja sin trazo. Si tanto *bSuppressFill* como *bSuppressStroke* se definen como `true`, el método no tiene ningún efecto.

Ejemplo

El ejemplo siguiente añade un rectángulo nuevo sin esquinas redondeadas dentro de las coordenadas especificadas; 100 píxeles de anchura y altura:

```
flash.getDocumentDOM().addNewRectangle({left:0,top:0,right:100,bottom:100},
  0);
```

El ejemplo siguiente añade un rectángulo nuevo sin esquinas redondeadas y sin relleno; 100 píxeles de anchura y 200 de altura:

```
flash.getDocumentDOM().addNewRectangle({left:10,top:10,right:110,bottom:210
  },0, true);
```

El ejemplo siguiente añade un rectángulo nuevo sin esquinas redondeadas y sin trazo; 200 píxeles de anchura y 200 de altura:

```
flash.getDocumentDOM().addNewRectangle({left:20,top:20,right:220,bottom:120
  },0, false, true);
```

document.addNewScene()

Disponibilidad

Flash MX 2004.

Uso

```
document.addNewScene( [name] )
```

Parámetros

name Especifica el nombre de la escena. Si no especifica un nombre, se generará un nombre de escena nuevo.

Valor devuelto

Un valor booleano: `true` si la escena se añade correctamente; `false` en caso contrario.

Descripción

Método; añade una nueva escena (Objeto Timeline) después de la escena seleccionada y la convierte en la seleccionada actualmente. Si el nombre de la escena especificada ya existe, la escena no se añade y el método devuelve un error.

Ejemplo

El ejemplo siguiente añade una nueva escena llamada `myScene` después de la escena actual en el documento actual. La variable `success` será `true` cuando se cree la nueva escena, y `false` en caso contrario.

```
var success = flash.getDocumentDOM().addNewScene("myScene");
```

El ejemplo siguiente añade una nueva escena utilizando la convención de asignación de nombres predeterminada. Si sólo existe una escena, la escena recién creada se llamará `"Scene 2"`.

```
fl.getDocumentDOM().addNewScene();
```

document.addNewText()

Disponibilidad

Flash MX 2004.

Uso

```
document.addNewText( boundingRectangle )
```

Parámetros

`boundingRectangle` Especifica el tamaño y la ubicación del campo de texto; para más información sobre el formato de *`boundingRectangle`*, consulte `document.addNewRectangle()`. Deberá ir seguido de una llamada a `document.seTextString()` para rellenar el nuevo cuadro de texto.

Valor devuelto

Ninguno.

Descripción

Método; inserta un nuevo campo de texto vacío.

Ejemplo

El ejemplo siguiente crea un nuevo campo de texto en la esquina superior izquierda del escenario y, a continuación, define la cadena de texto como `"Hello World"`:

```
fl.getDocumentDOM().addNewText({left:0, top:0, right:100, bottom:100});
fl.getDocumentDOM().setTextString('Hello World!');
```

Véase también

`document.setTextString()`

document.align()

Disponibilidad

Flash MX 2004.

Uso

```
document.align( alignmode [, bUseDocumentBounds ] )
```

Parámetros

`alignmode` Una cadena que especifica cómo se alinea la selección. Los valores aceptables son: `"left"`, `"right"`, `"top"`, `"bottom"`, `"vertical center"` y `"horizontal center"`.

`bUseDocumentBounds` Un valor booleano que, si se define como `true`, hace que el método alinee los límites del documento. En caso contrario, el método utiliza los límites de los objetos seleccionados. El valor predeterminado es `false`. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; alinea la selección.

Ejemplo

El ejemplo siguiente alinea los objetos a la izquierda y con el escenario. Equivale a activar la opción En escenario en el panel Alienar y a hacer clic en el botón Alinear a la izquierda:

```
fl.getDocumentDOM().align("left", true);
```

Véase también

`document.distribute()`, `document.getAlignToDocument()`, `document.setAlignToDocument()`

document.allowScreens()

Disponibilidad

Flash MX 2004.

Uso

```
document.allowScreens()
```

Parámetros

Ninguno.

Valor devuelto

Un valor booleano: `true` si se puede utilizar `document.screenOutline` con seguridad, y `false` en caso contrario.

Descripción

Método; se utiliza antes de usar la propiedad `document.screenOutline`. Si este método devuelve el valor `true`, podrá acceder de forma segura a `document.screenOutline`; Flash muestra un error si accede a `document.screenOutline` en un documento sin pantallas.

Ejemplo

El ejemplo siguiente determina si se pueden emplear métodos `screens` en el documento actual:

```
if(fl.getDocumentDOM().allowScreens()) {
  fl.trace("screen outline is available.");
}
else {
  fl.trace("whoops, no screens.");
}
```

Véase también

`document.screenOutline`

document.arrange()

Disponibilidad

Flash MX 2004.

Uso

```
document.arrange( arrangeMode )
```

Parámetros

arrangeMode Especifica la dirección en la que se mueve la selección. Los valores aceptables son: `"back"`, `"backward"`, `"forward"` y `"front"`. Ofrece las mismas posibilidades que las opciones del menú Modificar > Organizar.

Valor devuelto

Ninguno.

Descripción

Método; organiza la selección en el escenario. Este método sólo se aplica a objetos sin forma.

Ejemplo

El ejemplo siguiente mueve la selección actual a un primer plano:

```
fl.getDocumentDOM().arrange("front");
```

document.autoLabel

Disponibilidad

Flash MX 2004.

Uso

```
document.autoLabel
```

Descripción

Propiedad; un valor booleano que equivale a la casilla de verificación Etiquetado automático del panel Accesibilidad. Puede utilizar esta propiedad para indicar a Flash que etiquete objetos automáticamente en el escenario con el texto asociado a ellos.

Ejemplo

El ejemplo siguiente obtiene el valor de la propiedad `autoLabel` y muestra el resultado en el panel Salida:

```
var isAutoLabel = fl.getDocumentDOM().autoLabel;
fl.trace(isAutoLabel);
```

El ejemplo siguiente define la propiedad `autoLabel` como `true`, lo que indica a Flash que etiquete objetos automáticamente en el escenario:

```
fl.getDocumentDOM().autoLabel = true;
```

document.backgroundColor

Disponibilidad

Flash MX 2004.

Uso

```
document.backgroundColor
```

Descripción

Propiedad; el color del fondo, en uno de los formatos siguientes:

- Una cadena con el formato `"#RRGGBB"` o `"#RRGGBBAA"`
- Un número hexadecimal con el formato `0xRRGGBB`
- Un entero que representa el equivalente decimal del número hexadecimal

Ejemplo

El ejemplo siguiente establece el color de fondo como negro:

```
fl.getDocumentDOM().backgroundColor = '#000000';
```

document.breakApart()

Disponibilidad

Flash MX 2004.

Uso

```
document.breakApart()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; realiza una operación de separación en la selección actual.

Ejemplo

El ejemplo siguiente separa la selección actual:

```
fl.getDocumentDOM().breakApart();
```

document.canEditSymbol()

Disponibilidad

Flash MX 2004.

Uso

```
document.canEditSymbol()
```

Parámetros

Ninguno.

Valor devuelto

Un valor booleano: `true` si el menú Editar símbolos y las funciones están disponibles para utilizar, y `false` en caso contrario.

Descripción

Método; indica si el menú Editar símbolos y las funciones están activados. No indica si la selección se puede editar. Este método no se debe utilizar para comprobar si se permite `fl.getDocumentDOM().enterEditMode()`.

Ejemplo

El ejemplo siguiente muestra en el panel Salida el estado del menú Editar símbolos y las funciones:

```
fl.trace("fl.getDocumentDOM().canEditSymbol() returns: " +
  fl.getDocumentDOM().canEditSymbol());
```

document.canRevert()

Disponibilidad

Flash MX 2004.

Uso

```
document.canRevert()
```

Parámetros

Ninguno.

Valor devuelto

Un valor booleano: `true` si puede utilizar correctamente los métodos `document.revert()` o `fl.revertDocument()`, y `false` en caso contrario.

Descripción

Método; determina si puede utilizar correctamente el método `document.revert()` o `fl.revertDocument()`.

Ejemplo

El ejemplo siguiente comprueba si el documento actual puede volver a la versión guardada anteriormente. Si es así, `fl.getDocumentDOM().revert()` restaura la versión guardada anteriormente.

```
if(fl.getDocumentDOM().canRevert()){
  fl.getDocumentDOM().revert();
}
```

document.canTestMovie()

Disponibilidad

Flash MX 2004.

Uso

```
document.canTestMovie()
```

Parámetros

Ninguno.

Valor devuelto

Un valor booleano: `true` si puede utilizar correctamente el método `document.testMovie()`, y `false` en caso contrario.

Descripción

Método; determina si puede utilizar correctamente el método `document.testMovie()`.

Ejemplo

El ejemplo siguiente comprueba si puede utilizarse `fl.getDocumentDOM().testMovie()`. Si es así, llama al método.

```
if(fl.getDocumentDOM().canTestMovie()){
  fl.getDocumentDOM().testMovie();
}
```

Véase también

`document.canTestScene()`, `document.testScene()`

document.canTestScene()

Disponibilidad

Flash MX 2004.

Uso

```
document.canTestScene()
```

Parámetros

Ninguno.

Valor devuelto

Un valor booleano: `true` si puede utilizar correctamente el método `document.testScene()`, y `false` en caso contrario.

Descripción

Método; determina si puede utilizar correctamente el método `document.testScene()`.

Ejemplo

El ejemplo siguiente comprueba en primer lugar si puede utilizarse correctamente `fl.getDocumentDOM().testScene()`. Si es así, llama al método.

```
if(fl.getDocumentDOM().canTestScene()){
  fl.getDocumentDOM().testScene();
}
```

Véase también

`document.canTestMovie()`, `document.testMovie()`

document.changeFilterOrder()

Disponibilidad

Flash 8.

Uso

```
document.changeFilterOrder( oldIndex, newIndex )
```

Parámetros

`oldIndex` Un entero que representa la posición actual del índice basado en cero del filtro que desea reubicar en la lista Filtros.

`newIndex` Un entero que representa la nueva posición del índice del filtro en la lista.

Valor devuelto

Ninguno.

Descripción

Método; cambia el índice del filtro de la lista Filtro. Todos los filtros por encima o por debajo de *newIndex* se cambian hacia arriba o hacia abajo según corresponda. Por ejemplo, utilizando los filtros mostrados a continuación, si ejecuta el comando `fl.getDocumentDOM().changeFilterOrder(3, 0)`, los filtros se reorganizarán de este modo:

Antes: `blurFilter, dropShadowFilter, glowFilter, gradientBevelFilter`

Después: `gradientBevelFilter, blurFilter, dropShadowFilter, glowFilter`

Si ejecuta después el comando `fl.getDocumentDOM().changeFilterOrder(0, 2)`, los filtros se reorganizan de la manera siguiente:

Antes: `gradientBevelFilter, blurFilter, dropShadowFilter, glowFilter`

Después: `blurFilter, dropShadowFilter, gradientBevelFilter, glowFilter`

Ejemplo

El ejemplo siguiente mueve a la primera posición el filtro que se encuentra actualmente en la segunda posición de la lista Filtro:

```
fl.getDocumentDOM().changeFilterOrder(1,0);
```

Véase también

`document.addFilter()`, `document.disableFilter()`, `document.enableFilter()`, `document.getFilters()`, `document.removeFilter()`, Objeto Filter

document.clipCopy()

Disponibilidad

Flash MX 2004.

Uso

```
document.clipCopy()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; copia la selección actual desde el documento hasta el Portapapeles.

Ejemplo

El ejemplo siguiente copia la selección actual desde el documento hasta el Portapapeles:

```
fl.getDocumentDOM().clipCopy();
```

document.clipCut()

Disponibilidad

Flash MX 2004.

Uso

```
document.clipCut()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; corta la selección actual del documento y la escribe en el Portapapeles.

Ejemplo

El ejemplo siguiente corta la selección actual del documento y la escribe en el Portapapeles:

```
fl.getDocumentDOM().clipCut();
```

document.clipPaste()

Disponibilidad

Flash MX 2004.

Uso

```
document.clipPaste( [bInPlace] )
```

Parámetros

bInPlace Un valor booleano que, cuando se define como `true`, hace que el método realice una operación Pegar in situ. El valor predeterminado es `false`, lo que hace que el método realice una operación de pegado en el centro del documento. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; pega el contenido del Portapapeles en el documento.

Ejemplo

Los ejemplos siguientes pegan el contenido del Portapapeles en el centro del documento:

```
fl.getDocumentDOM().clipPaste();
```

El ejemplo siguiente pega el contenido del Portapapeles en el documento actual:

```
fl.getDocumentDOM().clipPaste(true);
```

document.close()

Disponibilidad

Flash MX 2004.

Uso

```
document.close( [bPromptToSaveChanges] )
```

Parámetros

bPromptToSaveChanges Un valor booleano que, cuando se define como `true`, hace que el método presente al usuario un cuadro de diálogo si hay cambios sin guardar en el documento. Si *bPromptToSaveChanges* se define como `false`, no se pregunta al usuario si desea guardar los documentos modificados. El valor predeterminado es `true`. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; cierra el documento especificado.

Ejemplo

El ejemplo siguiente cierra el documento actual y muestra al usuario un cuadro de diálogo para guardar los cambios:

```
fl.getDocumentDOM().close();
```

El ejemplo siguiente cierra el documento actual sin guardar los cambios:

```
fl.getDocumentDOM().close(false);
```

document.convertLinesToFills()

Disponibilidad

Flash MX 2004.

Uso

```
document.convertLinesToFills()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; convierte líneas en rellenos en los objetos seleccionados.

Ejemplo

El ejemplo siguiente convierte las líneas seleccionadas actualmente en rellenos:

```
fl.getDocumentDOM().convertLinesToFills();
```

document.convertToSymbol()

Disponibilidad

Flash MX 2004.

Uso

```
document.convertToSymbol( type, name, registrationPoint )
```

Parámetros

type Una cadena que especifica el tipo de símbolo que se va a crear. Los valores aceptables son: `"movie clip"`, `"button"` y `"graphic"`.

name Una cadena que especifica el nombre del nuevo símbolo, que debe ser exclusivo. Puede enviar una cadena vacía para que este método cree un nombre de símbolo único.

registration point Especifica el punto que representa la ubicación 0,0 del símbolo. Los valores válidos son: `"top left"`, `"top center"`, `"top right"`, `"center left"`, `"center"`, `"center right"`, `"bottom left"`, `"bottom center"` y `"bottom right"`.

Valor devuelto

Un objeto para el símbolo recién creado o `null` si no puede crear el símbolo.

Descripción

Método; convierte el elemento o elemento de escenario seleccionados en un símbolo nuevo. Para obtener información sobre la definición de propiedades de vinculación y elementos compartidos para un símbolo, consulte Objeto Item.

Ejemplo

Los ejemplos siguientes crean un símbolo de clip de película con un nombre especificado, un símbolo de botón con un nombre especificado o un símbolo de clip de película con un nombre predeterminado:

```
newMc = fl.getDocumentDOM().convertToSymbol("movie clip", "mcSymbolName",
  "top left");
newButton = fl.getDocumentDOM().convertToSymbol("button", "btnSymbolName",
  "bottom right");
newClipWithDefaultName = fl.getDocumentDOM().convertToSymbol("movie clip",
  "", "top left");
```

document.crop()

Disponibilidad

Flash 8.

Uso

```
document.crop()
```

Parámetros

Ninguno.

Valor devuelto

Un valor booleano: `true` si es correcto y `false` en caso contrario.

Descripción

Método; utiliza el objeto de dibujo seleccionado en la parte superior para recortar todos los objetos de dibujo seleccionados por debajo. Este método devuelve `false` si no hay objetos de dibujo seleccionados o si alguno de los elementos seleccionados no es un objeto de dibujo.

Ejemplo

El ejemplo siguiente recorta los objetos seleccionados actualmente:

```
fl.getDocumentDOM().crop();
```

Véase también

`document.deleteEnvelope()`, `document.intersect()`, `document.punch()`, `document.union()`, `shape.isDrawingObject`

document.currentPublishProfile

Disponibilidad

Flash MX 2004.

Uso

```
document.currentPublishProfile
```

Descripción

Propiedad; una cadena que especifica el nombre del perfil de publicación activo para el documento especificado.

Ejemplo

El ejemplo siguiente añade un nuevo perfil de publicación con el nombre predeterminado y, a continuación, muestra el nombre del perfil en el panel Salida:

```
fl.getDocumentDOM().addNewPublishProfile();
fl.outputPanel.trace(fl.getDocumentDOM().currentPublishProfile);
```

El ejemplo siguiente cambia el perfil de publicación seleccionado a `"Default"`:

```
fl.getDocumentDOM().currentPublishProfile = "Default";
```

document.currentTimeline

Disponibilidad

Flash MX 2004.

Uso

```
document.currentTimeline
```

Descripción

Propiedad; un entero que especifica el índice de la línea de tiempo activa. Puede establecer la línea de tiempo activa cambiando el valor de esta propiedad; el efecto es prácticamente equivalente a llamar a `document.editScene()`. La única diferencia es que no aparece un mensaje de error si el índice de la línea de tiempo no es válido, la propiedad simplemente no se establece, lo que provoca un error sin mensaje.

Ejemplo

El ejemplo siguiente muestra el índice de la línea de tiempo actual.

```
var myCurrentTL = fl.getDocumentDOM().currentTimeline;
fl.trace("The index of the current timeline is: "+ myCurrentTL);
```

El ejemplo siguiente cambia la línea de tiempo activa desde la línea de tiempo principal hasta una escena llamada "myScene".

```
var i = 0;
var curTimelines = fl.getDocumentDOM().timelines;
while(i < fl.getDocumentDOM().timelines.length){
  if(curTimelines[i].name == "myScene"){
    fl.getDocumentDOM().currentTimeline = i;
  }
  ++i;
}
```

Véase también

`document.getTimeline()`

document.deleteEnvelope()

Disponibilidad

Flash 8.

Uso

```
document.deleteEnvelope();
```

Parámetros

Ninguno.

Valor devuelto

Un valor booleano: `true` si es correcto y `false` en caso contrario.

Descripción

Método; elimina la envoltura (recuadro de delimitación que contiene uno o varios objetos) de los objetos seleccionados.

Ejemplo

El ejemplo siguiente elimina la envoltura de los objetos seleccionados:

```
fl.getDocumentDOM().deleteEnvelope();
```

Véase también

`document.crop()`, `document.intersect()`, `document.punch()`, `document.union()`, `shape.isDrawingObject`

document.deletePublishProfile()

Disponibilidad

Flash MX 2004.

Uso

```
document.deletePublishProfile()
```

Parámetros

Ninguno.

Valor devuelto

Un entero que es el índice del nuevo perfil actual. Si no hay un nuevo perfil disponible, el método deja el perfil actual sin modificar y devuelve su índice.

Descripción

Método; elimina el perfil activo actualmente, si hay más de uno. Debe quedar un perfil como mínimo.

Ejemplo

El ejemplo siguiente elimina el perfil activo actualmente, si hay más de uno, y muestra el índice del nuevo perfil activo:

```
alert(fl.getDocumentDOM().deletePublishProfile());
```

Véase también

`document.addNewPublishProfile()`

document.deleteScene()

Disponibilidad

Flash MX 2004.

Uso

```
document.deleteScene()
```

Parámetros

Ninguno.

Valor devuelto

Un valor booleano: `true` si la escena se elimina correctamente, y `false` en caso contrario.

Descripción

Método; elimina la escena actual (Objeto Timeline) y, si la escena eliminada no era la última, establece la siguiente escena como el objeto Timeline actual. Si la escena eliminada era la última, establece el primer objeto como objeto Timeline actual. Si sólo existe un objeto Timeline (escena), devuelve el valor `false`.

Ejemplo

Suponiendo que haya tres escenas (`Scene0`, `Scene1` y `Scene2`) en el documento actual, el ejemplo siguiente convierte `Scene2` en la escena actual y, a continuación, la elimina:

```
fl.getDocumentDOM().editScene(2);
var success = fl.getDocumentDOM().deleteScene();
```

document.deleteSelection()

Disponibilidad

Flash MX 2004.

Uso

```
document.deleteSelection()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; elimina la selección actual en el escenario. Muestra un mensaje de error si no hay ninguna selección.

Ejemplo

El ejemplo siguiente elimina la selección actual en el documento:

```
fl.getDocumentDOM().deleteSelection();
```

document.description

Disponibilidad

Flash MX 2004.

Uso

```
document.description
```

Descripción

Propiedad; una cadena que equivale al campo Descripción del panel Accesibilidad. El lector de pantalla lee esta descripción.

Ejemplo

El ejemplo siguiente establece la descripción del documento:

```
fl.getDocumentDOM().description= "This is the main movie";
```

El ejemplo siguiente obtiene la descripción del documento y la muestra en el panel Salida:

```
fl.trace(fl.getDocumentDOM().description);
```

document.disableAllFilters()

Disponibilidad

Flash 8.

Uso

```
document.disableAllFilters()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; desactiva todos los filtros en los objetos seleccionados.

Ejemplo

El ejemplo siguiente desactiva todos los filtros en los objetos seleccionados:

```
fl.getDocumentDOM().disableAllFilters();
```

Véase también

`document.addFilter()`, `document.changeFilterOrder()`, `document.disableFilter()`, `document.disableOtherFilters()`, `document.enableAllFilters()`, `document.getFilters()`, `document.removeAllFilters()`, Objeto Filter

document.disableFilter()

Disponibilidad

Flash 8.

Uso

```
document.disableFilter( filterIndex )
```

Parámetros

filterIndex Un entero que representa el índice basado en cero del filtro en la lista Filtro.

Valor devuelto

Ninguno.

Descripción

Método; desactiva el filtro especificado en la lista Filtros.

Ejemplo

El ejemplo siguiente desactiva el primer y tercer filtros (valores de índice de 0 y 2) de la lista Filtros en el objeto u objetos seleccionados:

```
fl.getDocumentDOM().disableFilter(0);
fl.getDocumentDOM().disableFilter(2);
```

Véase también

`document.addFilter()`, `document.changeFilterOrder()`, `document.disableAllFilters()`, `document.disableOtherFilters()`, `document.enableFilter()`, `document.getFilters()`, `document.removeFilter()`, Objeto Filter

document.disableOtherFilters()

Disponibilidad

Flash 8.

Uso

```
document.disableOtherFilters( enabledFilterIndex )
```

Parámetros

enabledFilterIndex Un entero que representa el índice basado en cero del filtro que debe permanecer activado cuando se desactiven todos los demás filtros.

Valor devuelto

Ninguno.

Descripción

Método; desactiva todos los filtros salvo el que se encuentra en la posición especificada en la lista Filtros.

Ejemplo

El ejemplo siguiente desactiva todos los filtros salvo el segundo de la lista (valor de índice de 1):

```
fl.getDocumentDom().disableOtherFilters(1);
```

Véase también

`document.addFilter()`, `document.changeFilterOrder()`, `document.disableAllFilters()`, `document.disableFilter()`, `document.enableFilter()`, `document.getFilters()`, `document.removeFilter()`, Objeto Filter

document.distribute()

Disponibilidad

Flash MX 2004.

Uso

```
document.distribute( distributemode [, bUseDocumentBounds ] )
```

Parámetros

`distributemode` Una cadena que especifica dónde se distribuye el objeto seleccionado. Los valores aceptables son: `"left edge"`, `"horizontal center"`, `"right edge"`, `"top edge"`, `"vertical center"` y `"bottom edge"`.

`bUseDocumentBounds` Un valor booleano que, si se define como `true`, distribuye los objetos seleccionados empleando los límites del documento. En caso contrario, el método utiliza los límites del objeto seleccionado. El valor predeterminado es `false`.

Valor devuelto

Ninguno.

Descripción

Método; distribuye la selección.

Ejemplo

El ejemplo siguiente distribuye los objetos seleccionados por el borde superior:

```
fl.getDocumentDOM().distribute("top edge");
```

El ejemplo siguiente distribuye los objetos seleccionados por el borde superior y establece expresamente el parámetro *bUseDcoumentBounds*:

```
fl.getDocumentDOM().distribute("top edge", false);
```

El ejemplo siguiente distribuye los objetos seleccionados por sus bordes superiores, empleando los límites del documento:

```
fl.getDocumentDOM().distribute("top edge", true);
```

Véase también

`document.getAlignToDocument()`, `document.setAlignToDocument()`

document.distributeToLayers()

Disponibilidad

Flash MX 2004.

Uso

```
document.distributeToLayers()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; realiza una operación de distribución en capas en la selección actual; equivale a seleccionar Distribuir en capas. Este método muestra un error si no hay ninguna selección.

Ejemplo

El ejemplo siguiente distribuye la selección actual a las capas:

```
fl.getDocumentDOM().distributeToLayers();
```

document.documentHasData()

Disponibilidad

Flash MX 2004.

Uso

```
document.documentHasData( name )
```

Parámetros

name Una cadena que especifica el nombre de los datos que se van a comprobar.

Valor devuelto

Un valor booleano: `true` si el documento tiene datos persistentes, y `false` en caso contrario.

Descripción

Método; comprueba si el documento contiene datos persistentes con el nombre especificado.

Ejemplo

El ejemplo siguiente comprueba si el documento contiene datos persistentes con el nombre `"myData"`::

```
var hasData = fl.getDocumentDOM().documentHasData("myData");
```

Véase también

`document.addDataToDocument()`, `document.getDataFromDocument()`, `document.removeDataFromDocument()`

document.duplicatePublishProfile()

Disponibilidad

Flash MX 2004.

Uso

```
document.duplicatePublishProfile( [profileName ] )
```

Parámetros

profileName Una cadena que especifica el nombre exclusivo del perfil duplicado. Si no especifica un nombre, el método utiliza el nombre predeterminado. Este parámetro es opcional.

Valor devuelto

Un entero que es el índice del nuevo perfil en la lista de perfiles. Devuelve -1 si no se puede duplicar el perfil.

Descripción

Método; duplica el perfil activo y selecciona la versión duplicada.

Ejemplo

El ejemplo siguiente duplica el perfil activo actualmente y muestra el índice del nuevo perfil en el panel Salida:

```
fl.trace(fl.getDocumentDOM().duplicatePublishProfile("dup profile"));
```

document.duplicateScene()

Disponibilidad

Flash MX 2004.

Uso

```
document.duplicateScene()
```

Parámetros

Ninguno.

Valor devuelto

Un valor booleano: `true` si la escena se duplica correctamente, y `false` en caso contrario.

Descripción

Método; realiza una copia de la escena seleccionada, asignando un nombre exclusivo a la nueva escena y convirtiéndola en la actual.

Ejemplo

El ejemplo siguiente duplica la segunda escena del documento actual:

```
fl.getDocumentDOM().editScene(1); //establece la escena central como escena
  actual
var success = fl.getDocumentDOM().duplicateScene();
```

document.duplicateSelection()

Disponibilidad

Flash MX 2004.

Uso

```
document.duplicateSelection()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; duplica la selección en el escenario.

Ejemplo

El ejemplo siguiente duplica la selección actual, lo que equivale a hacer clic mientras se presiona la tecla Alt y, a continuación, se arrastra un elemento:

```
fl.getDocumentDOM().duplicateSelection();
```

document.editScene()

Disponibilidad

Flash MX 2004.

Uso

```
document.editScene( index )
```

Parámetros

index Un entero basado en cero que especifica la escena que desea editar.

Valor devuelto

Ninguno.

Descripción

Método; convierte la escena especificada en la escena seleccionada actualmente para editar.

Ejemplo

Suponiendo que haya tres escenas (`Scene0`, `Scene1` y `Scene2`) en el documento actual, el ejemplo siguiente convierte `Scene2` en la escena actual y, a continuación, la elimina:

```
fl.getDocumentDOM().editScene(2);
fl.getDocumentDOM().deleteScene();
```

document.enableAllFilters()

Disponibilidad

Flash 8.

Uso

```
document.enableAllFilters()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; activa todos los filtros de la lista Filtros para el objeto u objetos seleccionados.

Ejemplo

El ejemplo siguiente activa todos los filtros de la lista Filtros para el objeto u objetos seleccionados:

```
fl.getDocumentDOM().enableAllFilters()
```

Véase también

`document.addFilter()`, `document.changeFilterOrder()`, `document.disableAllFilters()`, `document.enableFilter()`, `document.getFilters()`, `document.removeAllFilters()`, Objeto Filter

document.enableFilter()

Disponibilidad

Flash 8.

Uso

```
document.enableFilter( filterIndex )
```

Parámetros

`filterIndex` Un entero que especifica el índice basado en cero del filtro en la lista Filtro que se desea activar.

Valor devuelto

Ninguno.

Descripción

Método; activa el filtro especificado para el objeto u objetos seleccionados.

Ejemplo

El ejemplo siguiente activa el segundo filtro del objeto u objetos seleccionados:

```
fl.getDocumentDOM().enableFilter(1);
```

Véase también

`document.addFilter()`, `document.changeFilterOrder()`, `document.disableFilter()`, `document.enableAllFilters()`, `document.getFilters()`, `document.removeFilter()`, Objeto Filter

document.enterEditMode()

Disponibilidad

Flash MX 2004.

Uso

```
document.enterEditMode( [editMode] )
```

Parámetros

`editMode` Una cadena que especifica el modo de edición. Los valores aceptables son: `"inPlace"` o `"newWindow"`. Si no se especifica ningún parámetro, la opción predeterminada es el modo de edición de símbolos. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; cambia la herramienta de edición al modo de edición especificado por el parámetro. Si no se especifica ningún parámetro, el método utiliza de forma predeterminada el modo de edición de símbolos, que equivale a hacer clic con el botón derecho del ratón en el símbolo para activar el menú contextual y seleccionar Edición.

Ejemplo

El ejemplo siguiente sitúa a Flash en modo de edición en contexto para el símbolo seleccionado actualmente:

```
fl.getDocumentDOM().enterEditMode('inPlace');
```

El ejemplo siguiente sitúa a Flash en modo de edición en una nueva ventana para el símbolo seleccionado actualmente:

```
fl.getDocumentDOM().enterEditMode('newWindow');
```

Véase también

`document.exitEditMode()`

document.exitEditMode()

Disponibilidad

Flash MX 2004.

Uso

```
document.exitEditMode()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; sale del modo de edición de símbolos y vuelve a seleccionar el siguiente nivel superior desde el modo de edición. Por ejemplo, si está editando un símbolo dentro de otro, este método sube un nivel desde el símbolo que está editando hasta el símbolo principal.

Ejemplo

El ejemplo siguiente sale del modo de edición de símbolos:

```
fl.getDocumentDOM().exitEditMode();
```

Véase también

`document.enterEditMode()`

document.exportPNG()

Disponibilidad

Flash 8.

Uso

```
document.exportPNG([fileURI [, bCurrentPNGSettings [, bCurrentFrame]]])
```

Parámetros

fileURI Una cadena, expresada como archivo:/// URI, que especifica el nombre del archivo exportado. Si *fileURI* es una cadena vacía o no está especificada, Flash mostrará el cuadro de diálogo Exportar película.

bCurrentPNGSettings Un valor booleano que especifica si se utiliza la configuración de publicación actual de PNG (`true`) o se muestra el cuadro de diálogo Exportar PNG (`false`). Este parámetro es opcional. El valor predeterminado es `false`.

bCurrentFrame Un valor booleano que especifica si se exporta sólo el fotograma actual (`true`) o todos los fotogramas, con cada fotograma un archivo PNG independiente (`false`). Este parámetro es opcional. El valor predeterminado es `false`.

Valor devuelto

Un valor booleano de `true` si el archivo se exporta correctamente como archivo PNG, y de `false` en caso contrario.

Descripción

Método; exporta el documento como uno o varios archivos PNG. Si se especifica *fileURI* y el archivo ya existe, se sobrescribe sin mostrar ninguna advertencia.

Ejemplo

El siguiente ejemplo exporta el fotograma actual del documento actual a myFile.png, con la configuración de publicación actual de PNG:

```
fl.getDocumentDOM().exportPNG("file:///C|/myProject/myFile.png", true,
  true);
```

document.exportPublishProfile()

Disponibilidad

Flash MX 2004.

Uso

```
document.exportPublishProfile( fileURI )
```

Parámetros

fileURI Una cadena, expresada como archivo:/// URI, que especifica la ruta del archivo XML al que se exporta el perfil.

Valor devuelto

Ninguno.

Descripción

Método; exporta el perfil activo actualmente a un archivo XML.

Ejemplo

El ejemplo siguiente exporta el perfil activo actualmente al archivo llamado profile.xml en la carpeta /Documents and Settings/nombredeusuario/Escritorio de la unidad C :

```
fl.getDocumentDOM().exportPublishProfile('file:///C|/Documents and
  Settings/username/Desktop/profile.xml');
```

document.exportSWF()

Disponibilidad

Flash MX 2004.

Uso

```
document.exportSWF( [ fileURI [, bCurrentSettings ] ] )
```

Parámetros

fileURI Una cadena, expresada como archivo:/// URI, que especifica el nombre del archivo exportado. Si *fileURI* está vacío o no se especifica, Flash mostrará el cuadro de diálogo Exportar película. Este parámetro es opcional.

bCurrentSettings Un valor booleano que, cuando se define como `true`, hace que Flash utilice la configuración de publicación actual de SWF. En caso contrario, Flash muestra el cuadro de diálogo Exportar Flash Player. El valor predeterminado es `false`. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; exporta el documento en formato SWC de Flash.

Ejemplo

El ejemplo siguiente exporta el documento a la ubicación de archivo especificada con la configuración de publicación actual:

```
fl.getDocumentDOM().exportSWF("file:///C|/Documents and Settings/joe_user/
  Desktop/qwerty.swf");
```

El ejemplo siguiente muestra el cuadro de diálogo Exportar película y el cuadro de diálogo Exportar Flash Player y, a continuación, exporta el documento de acuerdo con la configuración especificada:

```
fl.getDocumentDOM().exportSWF("", true);
```

El ejemplo siguiente muestra el cuadro de diálogo Exportar película y, a continuación, exporta el documento según la configuración especificada:

```
fl.getDocumentDOM().exportSWF();
```

document.forceSimple

Disponibilidad

Flash MX 2004.

Uso

```
document.forceSimple
```

Descripción

Propiedad; un valor booleano que especifica si los elementos secundarios del objeto especificado son accesibles. Equivale a la lógica inversa de la opción Hacer que los objetos secundarios sean accesibles del panel Accesibilidad. Es decir, si `forceSimple` es `true`, equivale a la opción desactivada Hacer que los objetos secundarios sean accesibles. Si `forceSimple` es `false`, equivale a la opción activada Hacer que los objetos secundarios sean accesibles.

Ejemplo

El ejemplo siguiente establece la variable `areChildrenAccessible` con el valor de la propiedad `forceSimple`; un valor de `false` significa que los elementos secundarios son accesibles:

```
var areChildrenAccessible = fl.getDocumentDOM().forceSimple;
```

El ejemplo siguiente define la propiedad `forceSimple` para permitir que los elementos secundarios del documento sean accesibles:

```
fl.getDocumentDOM().forceSimple = false;
```

document.frameRate

Disponibilidad

Flash MX 2004.

Uso

```
document.frameRate
```

Descripción

Propiedad; un valor flotante que especifica el número de fotogramas mostrados por segundo cuando se reproduce el archivo SWF; el valor predeterminado es 12. Definir esta propiedad equivale a definir la velocidad de reproducción de fotogramas predeterminada en el cuadro de diálogo Propiedades del documento (Modificar > Documento) en el archivo FLA.

Ejemplo

El ejemplo siguiente establece la velocidad de reproducción en 25,5 fotogramas por segundo:

```
fl.getDocumentDOM().frameRate = 25.5;
```

document.getAlignToDocument()

Disponibilidad

Flash MX 2004.

Uso

```
document.getAlignToDocument()
```

Parámetros

Ninguno.

Valor devuelto

Un valor booleano: `true` si en las preferencias se define la alineación de objetos con el escenario, y `false` en caso contrario.

Descripción

Método; equivale a recuperar el valor del botón En escenario en el panel Alinear. Obtiene la preferencia que puede emplearse para los métodos `document.align()`, `document.distribute()`, `document.match()` y `document.space()` en el documento.

Ejemplo

El ejemplo siguiente recupera el valor del botón En escenario en el panel Alinear. Si el valor devuelto es `true`, el botón En escenario está activo; en caso contrario, no lo está.

```
var isAlignToDoc = fl.getDocumentDOM().getAlignToDocument();
fl.getDocumentDOM().align("left", isAlignToDoc);
```

Véase también

`document.setAlignToDocument()`

document.getBlendMode()

Disponibilidad

Flash 8.

Uso

```
document.getBlendMode()
```

Parámetros

Ninguno.

Valor devuelto

Una cadena que especifica el modo de mezcla para el objeto u objetos seleccionados. Si hay más de un objeto seleccionado y tienen distintos modos de mezcla, la cadena refleja el modo de mezcla del objeto con la profundidad mayor.

NOTA	El valor devuelto es impredecible si la selección contiene objetos que no admiten modos de mezcla o tienen el valor "`normal`" de modo de mezcla.

Descripción

Método; devuelve una cadena que especifica el modo de mezcla para el objeto u objetos seleccionados.

Ejemplo

El ejemplo siguiente muestra el nombre del modo de mezcla en el panel Salida:

```
fl.trace(fl.getDocumentDom().getBlendMode());
```

document.getCustomFill()

Disponibilidad

Flash MX 2004.

Uso

```
document.getCustomFill( [ objectToFill ] )
```

Parámetros

`objectToFill` Una cadena que especifica la ubicación del objeto de relleno. Los valores siguientes son válidos:

- `"toolbar"` devuelve el objeto de relleno del panel Herramientas y del inspector de propiedades.
- `"selection"` devuelve el objeto de relleno de la selección.

Si omite este parámetro, el valor predeterminado es `"selection"`. Si no hay selección, el método devuelve `undefined`. Este parámetro es opcional.

Valor devuelto

El Objeto Fill especificado por el parámetro *`objectToFill`*, si es correcto; en caso contrario, devuelve `undefined`.

Descripción

Método; recupera el objeto de relleno de la forma seleccionada o, si se especifica, del panel Herramientas y del inspector de propiedades.

Ejemplo

El ejemplo siguiente obtiene el objeto de relleno de la selección y, a continuación, cambia a blanco el color de la selección:

```
var fill = fl.getDocumentDOM().getCustomFill();
fill.color = '#FFFFFF';
fill.style = "solid";
fl.getDocumentDOM().setCustomFill(fill);
```

El ejemplo siguiente devuelve el objeto de relleno del panel Herramientas y del inspector de propiedades y, a continuación, cambia la muestra de color a un degradado lineal:

```
var fill = fl.getDocumentDOM().getCustomFill("toolbar");
fill.style = "linearGradient";
fill.colorArray = [ 0x00ff00, 0xff0000, 0x0000ff ];
fill.posArray = [0, 100, 200];
fl.getDocumentDOM().setCustomFill( fill );
```

Véase también

`document.setCustomFill()`

document.getCustomStroke()

Disponibilidad

Flash MX 2004.

Uso

```
document.getCustomStroke( [locationOfStroke] )
```

Parámetros

`locationOfStroke` Una cadena que especifica la ubicación del objeto de trazo. Los valores siguientes son válidos:

- `"toolbar"`, si se define, devuelve el objeto de trazo del panel Herramientas y del inspector de propiedades.
- `"selection"`, si se define, devuelve el objeto de trazo de la selección.

Si omite este parámetro, el valor predeterminado es `"selection"`. Si no hay selección, devuelve `undefined`. Este parámetro es opcional.

Valor devuelto

El Objeto Stroke especificado por el parámetro *`locationOfStroke`*, si es correcto; en caso contrario, devuelve `undefined`.

Descripción

Devuelve el objeto de trazo de la forma seleccionada o, si se especifica, del panel Herramientas y del inspector de propiedades.

Ejemplo

El ejemplo siguiente devuelve la configuración de trazo actual de la selección y cambia el grosor del trazo a 2:

```
var stroke = fl.getDocumentDOM().getCustomStroke("selection");
stroke.thickness = 2;
fl.getDocumentDOM().setCustomStroke(stroke);
```

El ejemplo siguiente devuelve la configuración de trazo actual del panel Herramientas y del inspector de propiedades y establece el color de trazo en rojo:

```
var stroke = fl.getDocumentDOM().getCustomStroke("toolbar");
stroke.color = "#FF0000";
fl.getDocumentDOM().setCustomStroke(stroke);
```

Véase también

`document.setCustomStroke()`

document.getDataFromDocument()

Disponibilidad

Flash MX 2004.

Uso

```
document.getDataFromDocument( name )
```

Parámetros

name Una cadena que especifica el nombre de los datos que se van a devolver.

Valor devuelto

Los datos especificados.

Descripción

Método; recupera el valor de los datos especificados. El tipo devuelto depende del tipo de datos que estaba almacenado.

Ejemplo

El ejemplo siguiente añade un valor entero de 12 al documento actual y utiliza este método para mostrar el valor en el panel Salida:

```
fl.getDocumentDOM().addDataToDocument("myData", "integer", 12);
fl.trace(fl.getDocumentDOM().getDataFromDocument("myData"));
```

Véase también

`document.addDataToDocument()`, `document.documentHasData()`, `document.removeDataFromDocument()`

document.getElementProperty()

Disponibilidad

Flash MX 2004.

Uso

```
document.getElementProperty( propertyName )
```

Parámetros

`propertyName` Una cadena que especifica el nombre de la propiedad Element para la que se va a recuperar el valor.

Valor devuelto

Valor de la propiedad especificada. Devuelve `null` si la propiedad es un estado indeterminado, como cuando se seleccionan múltiples elementos con distintos valores de propiedad. Devuelve `undefined` si la propiedad no es válida para el elemento seleccionado.

Descripción

Método; obtiene la propiedad `Element` especificada para la selección actual. Para ver una lista de los valores aceptables, consulte "Resumen de propiedades del objeto Element" en la página 198.

Ejemplo

El ejemplo siguiente obtiene el `nombre` de la propiedad Element para la selección actual:

```
//elementName = el nombre de instancia del objeto seleccionado.
var elementName = fl.getDocumentDOM().getElementProperty("name");
```

Véase también

`document.setElementProperty()`

document.getElementTextAttr()

Disponibilidad

Flash MX 2004.

Uso

```
document.getElementTextAttr( attrName [, startIndex [, endIndex]] )
```

Parámetros

attrName Una cadena que especifica el nombre de la propiedad `TextAttrs` que se va a devolver. Para ver una lista de nombres de propiedades y valores posibles, consulte "Resumen de propiedades del objeto TextAttrs" en la página 464.

startIndex Un entero que especifica el índice del primer carácter, con un 0 (cero) que especifica la primera posición. Este parámetro es opcional.

endIndex Un entero que especifica el índice del último carácter. Este parámetro es opcional.

Valor devuelto

Si hay un campo de texto seleccionado, la propiedad se devuelve si sólo se utiliza un valor en el texto. Devuelve `undefined` si se utilizan varios valores en el campo de texto. Si hay varios campos de texto seleccionados y todos los valores de alineación de texto son iguales, el método devuelve este valor. Si hay varios campos de texto seleccionados, pero no todos los valores de alineación de texto son iguales, el método devuelve `undefined`. Si no se transfieren los argumentos opcionales, estas reglas se aplican al rango de texto seleccionado actualmente o a todo el campo de texto si no se está editando el texto. Si sólo se transfiere *startIndex*, se devuelve la propiedad del carácter a la derecha del índice, si todos los objetos de texto seleccionado coinciden con los valores. Si se transfieren *startIndex* y *endIndex*, el valor devuelto refleja toda la gama de caracteres desde *startIndex* hasta *endIndex* (no incluido).

Descripción

Método; obtiene una propiedad `TextAttrs` específica de los objetos de texto seleccionados. Se ignorarán los objetos seleccionados que no sean campos de texto. Para ver una lista de nombres de propiedades y valores posibles, consulte "Resumen de propiedades del objeto TextAttrs" en la página 464. Véase también `document.setElementTextAttr()`.

Ejemplo

El ejemplo siguiente obtiene el tamaño de los campos de texto seleccionados:

```
fl.getDocumentDOM().getElementTextAttr("size");
```

El ejemplo siguiente obtiene el color del carácter en el índice 3 en los campos de texto seleccionados:

```
fl.getDocumentDOM().getElementTextAttr("fillColor", 3);
```

El ejemplo siguiente obtiene el nombre de la fuente del texto desde el índice 2 hasta el índice 10 (no incluido) de los campos de texto seleccionados:

```
fl.getDocumentDOM().getElementTextAttr("face", 2, 10);
```

document.getFilters()

Disponibilidad

Flash 8.

Uso

```
document.getFilters()
```

Parámetros

Ninguno.

Valor devuelto

Una matriz que contiene la lista de filtros aplicados al objeto u objetos seleccionados actualmente.

Descripción

Método; devuelve una matriz que contiene la lista de filtros aplicados al objeto u objetos seleccionados actualmente. Si se seleccionan varios objetos y no tienen filtros idénticos, este método devuelve la lista de filtros aplicados al primer objeto seleccionado.

Ejemplo

Véase `document.setFilters()`.

Véase también

`document.addFilter()`, `document.changeFilterOrder()`, `document.setFilters()`, Objeto Filter

document.getMetadata()

Disponibilidad

Flash 8.

Uso

```
document.getMetadata()
```

Parámetros

Ninguno.

Valor devuelto

Una cadena que contiene los metadatos XML asociados al documento, o una cadena vacía si no hay metadatos.

Descripción

Método; devuelve una cadena que contiene los metadatos XML asociados al documento, o una cadena vacía si no hay metadatos.

Ejemplo

El ejemplo siguiente muestra metadatos XML del documento actual en el panel Salida:

```
fl.trace("XML Metadata is :" + fl.getDocumentDOM().getMetadata());
```

Véase también

`document.setMetadata()`

document.getSelectionRect()

Disponibilidad

Flash MX 2004.

Uso

```
document.getSelectionRect()
```

Parámetros

Ninguno.

Valor devuelto

El rectángulo de delimitación de la selección actual o 0 si no hay nada seleccionado. Para más información sobre el formato del valor devuelto, consulte `document.addNewRectangle()`.

Descripción

Método; obtiene el rectángulo de delimitación de la selección actual. Si la selección no es rectangular, se devuelve el rectángulo más pequeño que abarque la selección completa. El rectángulo se basa en el espacio de documento o, cuando se encuentra en modo de edición, el punto de registro del símbolo que se está editando.

Ejemplo

El ejemplo siguiente obtiene el rectángulo de delimitación para la selección actual y, a continuación, muestra sus propiedades:

```
var newRect = fl.getDocumentDOM().getSelectionRect();
var outputStr = "left: " + newRect.left + " top: " + newRect.top + " right:
  " + newRect.right + " bottom: " + newRect.bottom;
alert(outputStr);
```

Véase también

`document.selection`, `document.setSelectionRect()`

document.getTextString()

Disponibilidad

Flash MX 2004.

Uso

```
document.getTextString( [startIndex [, endIndex]] )
```

Parámetros

`startIndex` Un entero que es un índice del primer carácter que se va a obtener. Este parámetro es opcional.

`endIndex` Un entero que es un índice del último carácter que se va a obtener. Este parámetro es opcional.

Valor devuelto

Una cadena que contiene el texto seleccionado.

Descripción

Método; obtiene el texto seleccionado actualmente. Si no se transfieren los parámetros opcionales, se utilizará la selección de texto actual. Si el texto no está abierto para editar, se devolverá la cadena de texto completa. Si sólo se transfiere *startIndex*, se devolverá la cadena que comienza en ese índice y que termina al final del campo. Si se transfieren *startIndex* y *endIndex*, se devolverá la cadena que comienza desde *startIndex* hasta *endIndex* (no incluido).

Si hay varios campos de texto seleccionados, se devolverá la concatenación de todas las cadenas.

Ejemplo

El ejemplo siguiente obtiene la cadena de los campos de texto seleccionados:

```
fl.getDocumentDOM().getTextString();
```

El ejemplo siguiente obtiene la cadena en el índice de carácter 5 de los campos de texto seleccionados:

```
fl.getDocumentDOM().getTextString(5);
```

El ejemplo siguiente obtiene la cadena desde el índice de carácter 2 hasta el índice de carácter 10 (no incluido):

```
fl.getDocumentDOM().getTextString(2, 10);
```

Véase también

`document.setTextString()`

document.getTimeline()

Disponibilidad

Flash MX 2004.

Uso

```
document.getTimeline()
```

Parámetros

Ninguno.

Valor devuelto

El objeto Timeline actual.

Descripción

Método; recupera el Objeto Timeline actual en el documento. El objeto de línea de tiempo actual puede ser la escena actual, el símbolo que se está editando o la pantalla actual.

Ejemplo

El ejemplo siguiente obtiene el objeto Timeline y devuelve el número de fotogramas en la capa más larga:

```
var longestLayer = fl.getDocumentDOM().getTimeline().frameCount;
fl.trace("The longest layer has" + longestLayer + "frames");
```

El ejemplo siguiente entra en el modo de edición en contexto para el símbolo seleccionado en el escenario e inserta un fotograma en la línea de tiempo del símbolo.

```
fl.getDocumentDOM().enterEditMode("inPlace");
fl.getDocumentDOM().getTimeline().insertFrames();
```

El ejemplo siguiente obtiene el objeto Timeline y muestra su nombre:

```
var timeline = fl.getDocumentDOM().getTimeline();
alert(timeline.name);
```

Véase también

`document.currentTimeline`, `document.timelines`, `symbolItem.timeline`

document.getTransformationPoint()

Disponibilidad

Flash MX 2004.

Uso

```
document.getTransformationPoint()
```

Parámetros

Ninguno.

Valor devuelto

La ubicación del punto de transformación.

Descripción

Método; obtiene la ubicación del punto de transformación de la selección actual.
Puede emplear el punto de transformación para transformaciones como rotar y sesgar.

Ejemplo

El ejemplo siguiente obtiene el punto de transformación para la selección actual. La propiedad `transPoint.x` proporciona la coordenada *x* del punto de transformación. La propiedad `transPoint.y` proporciona la coordenada *y* del punto de transformación:

```
var transPoint = fl.getDocumentDOM().getTransformationPoint();
```

Véase también

`document.setTransformationPoint()`

document.group()

Disponibilidad

Flash MX 2004.

Uso

```
document.group()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; convierte la selección actual en un grupo.

Ejemplo

El ejemplo siguiente convierte los objetos de la selección actual en un grupo:

```
fl.getDocumentDOM().group();
```

Véase también

`document.unGroup()`

document.height

Disponibilidad

Flash MX 2004.

Uso

```
document.height
```

Descripción

Propiedad; un entero que especifica el alto del documento (escenario) en píxeles.

Ejemplo

El ejemplo siguiente establece la altura del escenario en 400 píxeles:

```
fl.getDocumentDOM().height = 400;
```

Véase también

`document.width`

document.importFile()

Disponibilidad

Flash 8.

Uso

```
document.importFile(fileURI [, importToLibrary])
```

Parámetros

`fileURI` Una cadena, expresada como archivo:/// URI, que especifica la ruta del archivo que se va a importar.

`importToLibrary` Un valor booleano que especifica si se importará el archivo sólo a la biblioteca del documento (`true`) o se colocará también una copia en el escenario (`false`). El valor predeterminado es `false`.

Valor devuelto

Un valor booleano que indica si el archivo se importó correctamente.

Descripción

Método; importa un archivo a un documento. Este método realiza la misma operación que el comando del menú Importar a biblioteca o Importar a escenario. Para importar un perfil de publicación, utilice `document.importPublishProfile()`:

Ejemplo

El siguiente ejemplo permite al usuario localizar el archivo que se importará al escenario.

```
var dom = fl.getDocumentDOM();
var URI = fl.browseForFileURL("select", "Import File");
dom.importFile(URI);
```

Véase también

`document.importSWF()`, `fl.browseForFileURL()`

document.importPublishProfile()

Disponibilidad

Flash MX 2004.

Uso

```
document.importPublishProfile( fileURI )
```

Parámetros

`fileURI` Una cadena, expresada como archivo:/// URI, que especifica la ruta del archivo XML que define el perfil que se va a importar.

Valor devuelto

Un entero que es el índice del archivo importado en la lista de perfiles. Devuelve -1 si no se puede importar el perfil.

Descripción

Método; importa un perfil desde un archivo.

Ejemplo

El ejemplo siguiente importa el perfil que contiene el archivo profile.xml y muestra su índice en la lista de perfiles:

```
alert(fl.getDocumentDOM().importPublishProfile('file:///C|/Documents and
  Settings/janeUser/Desktop/profile.xml'));
```

document.importSWF()

Disponibilidad

Flash MX 2004.

Uso

```
document.importSWF( fileURI )
```

Parámetros

fileURI Una cadena, expresada como archivo:/// URI, que especifica el archivo SWF que desea importar.

Valor devuelto

Ninguno.

Descripción

Método; importa un archivo SWF en el documento. Este método equivale a utilizar el comando de menú Importar para especificar un archivo SWF. En Flash 8 y posterior, también puede utilizar `document.importFile()` para importar un archivo SWF (así como otros tipos de archivos).

Ejemplo

El ejemplo siguiente importa el archivo `"mySwf.swf"` desde la carpeta Configuration de Flash:

```
fl.getDocumentDOM().importSWF(fl.configURI+"mySwf.swf");
```

Véase también

`document.importFile()`

document.intersect()

Disponibilidad

Flash 8.

Uso

```
document.intersect();
```

Parámetros

Ninguno.

Valor devuelto

Un valor booleano: `true` si es correcto y `false` en caso contrario.

Descripción

Método; crea un objeto de dibujo de intersección a partir de todos los objetos de dibujo seleccionados. Este método devuelve `false` si no hay objetos de dibujo seleccionados o si alguno de los elementos seleccionados no es un objeto de dibujo.

Ejemplo

El ejemplo siguiente crea un objeto de dibujo de intersección a partir de todos los objetos de dibujo seleccionados.

```
fl.getDocumentDOM().intersect();
```

Véase también

`document.crop()`, `document.deleteEnvelope()`, `document.punch()`, `document.union()`, `shape.isDrawingObject`

document.library

Disponibilidad

Flash MX 2004.

Uso

```
document.library
```

Descripción

Propiedad de sólo lectura; el Objeto library para un documento.

Ejemplo

El ejemplo siguiente obtiene la biblioteca para el documento seleccionado actualmente:

```
var myCurrentLib = fl.getDocumentDOM().library;
```

Suponiendo que el documento seleccionado actualmente no es `fl.documents[1]`, el ejemplo siguiente obtiene la biblioteca para una biblioteca no seleccionada o para una biblioteca que se abrió empleando Archivo > Abrir como biblioteca externa:

```
var externalLib = fl.documents[1].library;
```

document.livePreview

Disponibilidad

Flash MX 2004.

Uso

```
document.livePreview
```

Descripción

Propiedad; un valor booleano que especifica si está activada la opción Vista previa dinámica. Si se define como `true`, los componentes aparecen en el escenario tal como aparecerán en el contenido publicado, incluido su tamaño aproximado. Si se define como `false`, los componentes sólo aparecerán como contornos. El valor predeterminado es `true`.

Ejemplo

El ejemplo siguiente define Vista previa dinámica como `false`:

```
fl.getDocumentDOM().livePreview = false;
```

document.match()

Disponibilidad

Flash MX 2004.

Uso

```
document.match( bWidth, bHeight [, bUseDocumentBounds] )
```

Parámetros

`bWidth` Un valor booleano que, cuando se define como `true`, hace que el método iguale la anchura de los elementos seleccionados.

`bHeight` Un valor booleano que, cuando se define como `true`, hace que el método iguale la altura de los elementos seleccionados.

`bUseDocumentBounds` Un valor booleano que, si se define como `true`, hace que el método iguale el tamaño de los objetos con los límites del documento. En caso contrario, el método utiliza los límites del objeto mayor. El valor predeterminado es `false`. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; iguala el tamaño de los objetos seleccionados.

Ejemplo

El ejemplo siguiente sólo iguala la anchura de los objetos seleccionados:

```
fl.getDocumentDOM().match(true,false);
```

El ejemplo siguiente sólo iguala la altura:

```
fl.getDocumentDOM().match(false,true);
```

El ejemplo siguiente sólo iguala la anchura a los límites del documento:

```
fl.getDocumentDOM().match(true,false,true);
```

Véase también

`document.getAlignToDocument()`, `document.setAlignToDocument()`

document.mouseClick()

Disponibilidad

Flash MX 2004.

Uso

```
document.mouseClick( position, bToggleSel, bShiftSel )
```

Parámetros

`position` Un par de valores de coma flotante que especifican las coordenadas *x* e *y* del clic en píxeles.

`bToggleSel` Un valor booleano que especifica el estado de la tecla Mayús: `true` para presionada; `false` para no presionada.

`bShiftSel` Un valor booleano que especifica el estado de la opción Seleccionar presionando Mayúsculas en la aplicación: `true` para activada; `false` para desactivada.

Valor devuelto

Ninguno.

Descripción

Método; ejecuta un clic de ratón desde la herramienta Flecha.

Ejemplo

El ejemplo siguiente ejecuta un clic de ratón en la ubicación especificada:

```
fl.getDocumentDOM().mouseClick({x:300, y:200}, false);
```

Véase también

`document.mouseDblClk()`

document.mouseDblClk()

Disponibilidad

Flash MX 2004.

Uso

`document.mouseDblClk( position, bAltDown, bShiftDown, bShiftSelect )`

Parámetros

`position` Un par de valores de coma flotante que especifican las coordenadas *x* e *y* del clic en píxeles.

`bAltdown` Un valor booleano que registra si la tecla Alt está presionada en el momento del evento: `true` para presionada; `false` para no presionada.

`bShiftDown` Un valor booleano que registra si la tecla Mayús estaba presionada cuando se produjo el evento: `true` para presionada; `false` para no presionada.

`bShiftSelect` Un valor booleano que indica el estado de la opción Seleccionar presionando Mayúsculas en la aplicación: `true` para activada; `false` para desactivada.

Valor devuelto

Ninguno.

Descripción

Método; ejecuta un doble clic de ratón desde la herramienta Flecha.

Ejemplo

El ejemplo siguiente ejecuta un doble clic de ratón en la ubicación especificada:

```
fl.getDocumentDOM().mouseDblClk({x:392.9, y:73}, false, false, true);
```

Véase también

`document.mouseClick()`

document.moveSelectedBezierPointsBy()

Disponibilidad

Flash MX 2004.

Uso

```
document.moveSelectedBezierPointsBy( delta )
```

Parámetros

delta Un par de valores de coma flotante que especifican las coordenadas *x* e *y* en píxeles a las que se mueven los puntos Bézier seleccionados. Por ejemplo, (`{x:1,y:2}`) especifica una ubicación que se encuentra un píxel hacia la derecha y dos píxeles hacia abajo respecto a la ubicación actual.

Valor devuelto

Ninguno.

Descripción

Método; si la selección contiene como mínimo una ruta con al menos un punto Bézier seleccionado, este método mueve todos los puntos Bézier seleccionados en todas las rutas seleccionadas con la cantidad especificada.

Ejemplo

El ejemplo siguiente mueve los puntos Bézier seleccionados 10 píxeles hacia la derecha y 5 píxeles hacia abajo:

```
fl.getDocumentDOM().moveSelectedBezierPointsBy({x:10, y:5});
```

document.moveSelectionBy()

Disponibilidad

Flash MX 2004.

Uso

```
document.moveSelectionBy( distanceToMove )
```

Parámetros

distanceToMove Un par de valores de coma flotante que especifican las coordenadas *x* e *y* a las que el método mueve la selección. Por ejemplo, (`{x:1,y:2}`) especifica una ubicación que se encuentra un píxel hacia la derecha y dos píxeles hacia abajo respecto a la ubicación actual.

Valor devuelto

Ninguno.

Descripción

Método; mueve los objetos seleccionados una distancia especificada.

NOTA	Cuando se utilizan las teclas de flecha para mover el elemento, el panel Historial combina todas las pulsaciones de la tecla de flecha como un solo paso. Cuando el usuario presiona las teclas de flecha repetidamente, en lugar de dar varios pasos en el panel Historial, este método ejecuta un paso, y los argumentos se actualizan para reflejar las teclas de flecha repetidas.

Para obtener información sobre cómo realizar una selección, consulte `document.setSelectionRect()`, `document.mouseClick()`, `document.mouseDblClk()` y Objeto Element.

Ejemplo

El ejemplo siguiente mueve el elemento seleccionado 62 píxeles hacia la derecha y 84 píxeles hacia abajo:

```
flash.getDocumentDOM().moveSelectionBy({x:62, y:84});
```

document.name

Disponibilidad

Flash MX 2004.

Uso

```
document.name
```

Descripción

Propiedad de sólo lectura; una cadena que representa el nombre de un documento (archivo FLA).

Ejemplo

El ejemplo siguiente establece la variable `fileName` con el nombre de archivo del primer documento de la matriz de documentos:

```
var fileName = flash.documents[0].name;
```

El ejemplo siguiente muestra los nombres de todos los documentos abiertos en el panel Salida:

```
var openDocs = fl.documents;
for(var i=0;i < opendocs.length; i++){
  fl.trace(i + " " + opendocs[i].name +"\n");
}
```

document.optimizeCurves()

Disponibilidad

Flash MX 2004.

Uso

```
document.optimizeCurves( smoothing, bUseMultiplePasses )
```

Parámetros

smoothing Un entero entre 0 y 100, donde 0 especifica sin suavizado y 100, suavizado máximo.

bUseMultiplePasses Un valor booleano que, cuando se define como `true`, indica que el método debe emplear varias pasadas, lo que resulta más lento aunque ofrece mejores resultados. Este parámetro equivale a hacer clic en el botón Utilizar varias pasadas en el cuadro de diálogo Optimizar curvas.

Valor devuelto

Ninguno.

Descripción

Método; optimiza el suavizado para la selección actual, lo que permite múltiples pasadas, si se especifican, para un suavizado óptimo. Este método equivale a seleccionar Modificar > Forma > Optimizar.

Ejemplo

El ejemplo siguiente optimiza la curva de la selección actual a 50º de suavizado con varias pasadas:

```
fl.getDocumentDOM().optimizeCurves(50, true);
```

document.path

Disponibilidad

Flash MX 2004.

Uso

```
document.path
```

Descripción

Propiedad de sólo lectura; una cadena que representa la ruta del documento con el formato específico de la plataforma. Si el documento no se ha guardado nunca, esta propiedad es `undefined`.

Ejemplo

El ejemplo siguiente muestra la ruta del primer documento de la matriz de documentos en el panel Salida:

```
var filePath = flash.documents[0].path;
fl.trace(filePath);
```

document.publish()

Disponibilidad

Flash MX 2004.

Uso

```
document.publish()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; publica el documento de acuerdo con la Configuración de publicación activa (Archivo > Configuración de publicación). Este método equivale a seleccionar Archivo > Publicar.

Ejemplo

El ejemplo siguiente publica el documento actual:

```
fl.getDocumentDOM().publish();
```

document.publishProfiles

Disponibilidad

Flash MX 2004.

Uso

```
document.publishProfiles
```

Descripción

Propiedad de sólo lectura; una matriz de los nombres del perfil de publicación para el documento.

Ejemplo

El ejemplo siguiente muestra los nombres de los perfiles de publicación para el documento:

```
var myPubProfiles = fl.getDocumentDOM().publishProfiles;
for (var i=0; i < myPubProfiles.length; i++){
  fl.trace(myPubProfiles[i]);
}
```

document.punch()

Disponibilidad

Flash 8.

Uso

```
document.punch()
```

Parámetros

Ninguno.

Valor devuelto

Un valor booleano: `true` si es correcto y `false` en caso contrario.

Descripción

Método; utiliza el objeto de dibujo seleccionado en la parte superior para perforar todos los objetos de dibujo seleccionados por debajo. Este método devuelve `false` si no hay objetos de dibujo seleccionados o si alguno de los elementos seleccionados no es un objeto de dibujo.

Ejemplo

El ejemplo siguiente perfora los objetos de dibujo situados por debajo del objeto de dibujo seleccionado:

```
fl.getDocumentDOM().punch();
```

Véase también

`document.crop()`, `document.deleteEnvelope()`, `document.intersect()`, `document.union()`, `shape.isDrawingObject`

document.removeDataFromDocument()

Disponibilidad

Flash MX 2004.

Uso

```
document.removeDataFromDocument( name )
```

Parámetros

name Una cadena que especifica el nombre de los datos que se van a eliminar.

Valor devuelto

Ninguno.

Descripción

Método; elimina datos persistentes con el nombre especificado que se han asociado al documento.

Ejemplo

El ejemplo siguiente elimina del documento los datos persistentes con el nombre `"myData"`:

```
fl.getDocumentDOM().removeDataFromDocument("myData");
```

Véase también

`document.addDataToDocument()`, `document.documentHasData()`, `document.getDataFromDocument()`

document.removeDataFromSelection()

Disponibilidad

Flash MX 2004.

Uso

```
document.removeDataFromSelection( name )
```

Parámetros

name Una cadena que especifica el nombre de los datos persistentes que se van a eliminar.

Valor devuelto

Ninguno.

Descripción

Método; elimina datos persistentes con el nombre especificado que se han asociado a la selección.

Ejemplo

El ejemplo siguiente elimina de la selección los datos persistentes con el nombre `"myData"`:

```
fl.getDocumentDOM().removeDataFromSelection("myData");
```

Véase también

`document.addDataToSelection()`

document.removeAllFilters()

Disponibilidad

Flash 8.

Uso

```
document.removeAllFilters()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; elimina todos los filtros del objeto u objetos seleccionados.

Ejemplo

El ejemplo siguiente elimina todos los filtros del objeto u objetos seleccionados:

```
fl.getDocumentDOM().removeAllFilters();
```

Véase también

`document.addFilter()`, `document.changeFilterOrder()`, `document.disableAllFilters()`, `document.getFilters()`, `document.removeFilter()`, Objeto Filter

document.removeFilter()

Disponibilidad

Flash 8.

Uso

```
document.removeFilter( filterIndex )
```

Parámetros

`filterIndex` Un entero que especifica el índice basado en cero del filtro que se va a eliminar del objeto u objetos seleccionados.

Valor devuelto

Ninguno.

Descripción

Método; elimina el filtro especificado de la lista Filtros del objeto u objetos seleccionados.

Ejemplo

El ejemplo siguiente elimina el primer filtro (valor de índice de 0) de la lista Filtros del objeto u objetos seleccionados:

```
fl.getDocumentDOM().removeFilter(0);
```

Véase también

`document.addFilter()`, `document.changeFilterOrder()`, `document.disableFilter()`, `document.getFilters()`, `document.removeAllFilters()`, `Objeto Filter`

document.renamePublishProfile()

Disponibilidad

Flash MX 2004.

Uso

```
document.renamePublishProfile( [profileNewName ] )
```

Parámetros

profileNewName Un parámetro opcional que especifica el nuevo nombre del perfil. El nombre nuevo debe ser único. Si no se especifica el nombre, se suministra un nombre predeterminado.

Valor devuelto

Un valor booleano: `true` si el nombre se cambia correctamente, y `false` en caso contrario.

Descripción

Método; cambia el nombre del perfil actual.

Ejemplo

El ejemplo siguiente cambia el nombre del perfil actual por un nombre predeterminado y lo muestra:

```
alert(fl.getDocumentDOM().renamePublishProfile());
```

document.renameScene()

Disponibilidad

Flash MX 2004.

Uso

```
document.renameScene( name )
```

Parámetros

name Una cadena que especifica el nuevo nombre de la escena.

Valor devuelto

Un valor booleano: `true` si el nombre se cambia correctamente, y `false` en caso contrario. Si el nuevo nombre no es único, por ejemplo, el método devuelve `false`.

Descripción

Método; cambia el nombre de la escena seleccionada actualmente en el panel Escenas. El nuevo nombre de la escena seleccionada debe ser único.

Ejemplo

El ejemplo siguiente cambia el nombre de la escena actual por `"new name"`:

```
var success = fl.getDocumentDOM().renameScene("new name");
```

document.reorderScene()

Disponibilidad

Flash MX 2004.

Uso

```
document.reorderScene( sceneToMove, sceneToPutItBefore )
```

Parámetros

`sceneToMove` Un entero que especifica qué escena se va a mover, donde 0 (cero) es la primera escena.

`sceneToPutItBefore` Un entero que especifica la escena antes de la cual desea mover la escena especificada por *`sceneToMove`*. Especifique 0 (cero) para la primera escena. Por ejemplo, si especifica 1 para *`sceneToMove`* y 0 para *`sceneToPutItBefore`*, la segunda escena se situará delante de la primera. Especifique -1 para mover la escena al final.

Valor devuelto

Ninguno.

Descripción

Método; mueve la escena especificada delante de otra escena especificada.

Ejemplo

El ejemplo siguiente mueve la segunda escena delante de la primera:

```
fl.getDocumentDOM().reorderScene(1, 0);
```

document.resetTransformation()

Disponibilidad

Flash MX 2004.

Uso

```
document.resetTransformation()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; restablece la matriz de transformación. Este método equivale a seleccionar Modificar > Transformar > Quitar transformación.

Ejemplo

El ejemplo siguiente restablece la matriz de transformación para la selección actual:

```
fl.getDocumentDOM().resetTransformation();
```

document.revert()

Disponibilidad

Flash MX 2004.

Uso

```
document.revert()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; devuelve el documento especificado a la versión guardada anteriormente. Este método equivale a seleccionar Archivo > Descartar cambios.

Ejemplo

El ejemplo siguiente devuelve el documento actual a la versión guardada anteriormente:

```
fl.getDocumentDOM().revert();
```

Véase también

`document.canRevert()`, `fl.revertDocument()`

document.rotateSelection()

Disponibilidad

Flash MX 2004.

Uso

```
document.rotateSelection( angle [, rotationPoint] )
```

Parámetros

`angle` Un valor de coma flotante que especifica el ángulo de rotación.

`rotationPoint` Una cadena que especifica qué lado del recuadro de delimitación va a rotar. Los valores aceptables son: `"top right"`, `"top left"`, `"bottom right"`, `"bottom left"`, `"top center"`, `"right center"`, `"bottom center"` y `"left center"`. Si no se especifica, el método utiliza el punto de transformación. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; gira la selección el número de grados especificado. Se obtiene el mismo efecto que con la herramienta Transformación libre para girar el objeto.

Ejemplo

El ejemplo siguiente gira la selección 45º alrededor del punto de transformación:

```
flash.getDocumentDOM().rotateSelection(45);
```

El ejemplo siguiente gira la selección 45º alrededor de la esquina inferior izquierda:

```
fl.getDocumentDOM().rotateSelection(45, "bottom left");
```

document.save()

Disponibilidad

Flash MX 2004.

Uso

```
document.save( [ bOkToSaveAs ] )
```

Parámetros

bOkToSaveAs Un parámetro opcional que especifica si se abrirá el cuadro de diálogo Guardar como.

Valor devuelto

Un valor booleano: `true` si la operación de guardado se realiza correctamente; `false` en caso contrario.

Descripción

Método; guarda el documento en su ubicación predeterminada. Este método equivale a seleccionar Archivo > Guardar.

NOTA	Si el archivo no se ha guardado nunca o no se ha modificado desde la última vez que se guardó, no se guardará y el valor devuelto será false. Para permitir que se guarde un archivo no guardado o no modificado, utilice `fl.saveDocumentAs()`.

Ejemplo

El ejemplo siguiente guarda el documento actual en su ubicación predeterminada.

```
fl.getDocumentDOM().save();
```

Véase también

`document.saveAndCompact()`, `fl.saveAll()`, `fl.saveDocument()`, `fl.saveDocumentAs()`

document.saveAndCompact()

Disponibilidad

Flash MX 2004.

Uso

```
document.saveAndCompact( [ bOkToSaveAs ] )
```

Parámetros

bOkToSaveAs Un parámetro opcional que, si se define como `true` o se omite y no se ha guardado nunca el archivo, abre el cuadro de diálogo Guardar como. Si se define como `false` y el archivo no se ha guardado nunca, el archivo no se guardará. El valor predeterminado es `true`.

Valor devuelto

Un valor booleano: `true` si la operación de guardar y compactar se realiza correctamente; `false` en caso contrario.

Descripción

Método; guarda y compacta el archivo. Este método equivale a seleccionar Archivo > Guardar y compactar.

NOTA	Si el archivo no se ha guardado nunca, este método devuelve true incluso si el usuario cancela el cuadro de diálogo Guardar como. Para permitir que se guarde un archivo no guardado, utilice `fl.saveDocumentAs()`.

Ejemplo

El ejemplo siguiente guarda y compacta el documento actual:

```
fl.getDocumentDOM().saveAndCompact();
```

Véase también

`document.save()`, `fl.saveDocumentAs()`, `fl.saveDocument()`, `fl.saveAll()`

document.scaleSelection()

Disponibilidad

Flash MX 2004.

Uso

```
document.scaleSelection( xScale, yScale [, whichCorner] )
```

Parámetros

xScale Un valor de coma flotante que especifica la cantidad de *x* que se va a escalar.

yScale Un valor de coma flotante que especifica la cantidad de *y* que se va a escalar.

whichCorner Un valor de cadena que especifica el borde sobre el que se produce la transformación. Si se omite, el escalado se produce sobre el punto de transformación. Los valores válidos son: `"bottom left"`, `"bottom right"`, `"top right"`, `"top left"`, `"top center"`, `"right center"`, `"bottom center"` y `"left center"`. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; escala la selección en la cantidad especificada. Este método equivale a utilizar la herramienta Transformación libre para escalar el objeto.

Ejemplo

El ejemplo siguiente amplía la anchura de la selección actual hasta el doble de la anchura original y reduce la altura a la mitad:

```
flash.getDocumentDOM().scaleSelection(2.0, 0.5);
```

El ejemplo siguiente voltea la selección en vertical:

```
fl.getDocumentDOM().scaleSelection(1, -1);
```

El ejemplo siguiente voltea la selección en horizontal:

```
fl.getDocumentDOM().scaleSelection(-1, 1);
```

El ejemplo siguiente escala la selección en vertical a 1,9 desde la parte superior central:

```
fl.getDocumentDOM().scaleSelection(1, 1.90, 'top center');
```

document.screenOutline

Disponibilidad

Flash MX 2004.

Uso

```
document.screenOutline
```

Descripción

Propiedad de sólo lectura; el objeto ScreenOutline actual para el documento. Antes de acceder al objeto por primera vez, asegúrese de utilizar `document.allowScreens()` para determinar si existe la propiedad.

Ejemplo

El ejemplo siguiente muestra la matriz de valores en la propiedad `screenOutline`:

```
var myArray = new Array();
for(var i in fl.getDocumentDOM().screenOutline) {
  myArray.push(" "+i+" : "+fl.getDocumentDOM().screenOutline[i]) ;
}
fl.trace("Here is the property dump for screenOutline: "+myArray);
```

Véase también

`document.allowScreens()`, Objeto ScreenOutline

document.selectAll()

Disponibilidad

Flash MX 2004.

Uso

```
document.selectAll()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; selecciona todos los elementos del escenario. Este método equivale a presionar Control+A (Windows) o Comando+A (Macintosh) o a seleccionar Edición > Seleccionar todo.

Ejemplo

El ejemplo siguiente selecciona todo lo que está visible actualmente para el usuario:

```
fl.getDocumentDOM().selectAll();
```

Véase también

`document.selection`, `document.selectNone()`

document.selection

Disponibilidad

Flash MX 2004.

Uso

```
document.selection
```

Descripción

Propiedad; una matriz de los objetos seleccionados en el documento. Si no hay nada seleccionado, devuelve una matriz de longitud cero. Si no hay ningún documento abierto, devuelve `null`.

Para añadir objetos a la matriz, deberá seleccionarlos en primer lugar de una de estas dos formas:

- Seleccione manualmente los objetos en el escenario.
- Utilice uno de los métodos de selección, como `document.setSelectionRect()`, `document.setSelectionBounds()`, `document.mouseClick()`, `document.mouseDblClk()` o `document.selectAll()`.
- Seleccione manualmente uno o varios fotogramas.
- Utilice uno de los métodos del Objeto Timeline para seleccionar uno o varios fotogramas, como `timeline.getSelectedFrames()`, `timeline.setSelectedFrames()` o `timeline.selectAllFrames()`.
- Especifique un determinado elemento en un determinado fotograma. Por ejemplo, el código siguiente especifica y selecciona un elemento:

```
fl.getDocumentDOM().selection =
  fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0];
```

Ejemplo

El ejemplo siguiente asigna todos los elementos del fotograma 11 a la selección actual (recuerde que los valores de índice son distintos de los valores de número de fotograma):

```
fl.getDocumentDOM().getTimeline().currentFrame = 10;
fl.getDocumentDOM().selection =
  fl.getDocumentDOM().getTimeline().layers[0].frames[10].elements;
```

El ejemplo siguiente crea un rectángulo en la esquina superior izquierda del escenario y una cadena de texto bajo el rectángulo. A continuación, selecciona ambos objetos empleando `document.setSelectionRect()` y los añade a la matriz `document.selection`. Por último, muestra el contenido de `document.selection` en el panel Salida.

```
fl.getDocumentDOM().addNewRectangle({left:0, top:0, right:99, bottom:99},
  0);
fl.getDocumentDOM().addNewText({left:-1, top:117.3, right:9.2,
  bottom:134.6});
fl.getDocumentDOM().setTextString('Hello World');
fl.getDocumentDOM().setSelectionRect({left:-28, top:-22, right:156.0,
  bottom:163});

var theSelectionArray = fl.getDocumentDOM().selection;

for(var i=0;i<theSelectionArray.length;i++){
   fl.trace("fl.getDocumentDOM().selection["+i+"] = " +
     theSelectionArray[i]);
}
```

A continuación, se ofrece un ejemplo avanzado. Muestra cómo se realiza un bucle a través de la matriz de capas y la matriz de elementos para localizar instancias de un determinado símbolo y seleccionarlas. Puede ampliar este ejemplo para incluir bucles para varios fotogramas o escenas. Este ejemplo asigna todas las instancias del clip de película `myMovieClip` del primer fotograma a la selección actual:

```
// Asigna la matriz de capas a la variable "theLayers".
var theLayers = fl.getDocumentDOM().getTimeline().layers;
// Crea una matriz para contener todos los elementos
// que son instancias de "myMovieClip".
var myArray = new Array();
// Variable de contador
var x = 0;
// Comienza el bucle a través de todas las capas.
for (var i = 0; i < theLayers.length; i++) {
  // Obtiene la matriz de elementos del fotograma 1
  // y la asigna a la matriz "theElems".
  var theElems = theLayers[i].frames[0].elements;
  // Comienza el bucle a través de los elementos de una capa.
  for (var c = 0; c < theElems.length; c++) {
    // Comprueba si el elemento es de tipo "instancia".
    if (theElems[c].elementType == "instance") {
      // Si el elemento es una instancia, comprueba
      // si es una instancia de "myMovieClip".
      if (theElems[c].libraryItem.name == "myMovieClip") {
        // Asigna elementos que son instancias de "myMovieClip" a
  "myArray".
        myArray[x] = theElems[c];
        // Incrementa la variable de contador.
        x++;
      }
    }
  }
}
// Ahora que ha asignado todas las instancias de "myMovieClip"
// a "myArray", hará que la matriz document.selection
// sea igual que myArray. De este modo se seleccionan los objetos del
  escenario.
fl.getDocumentDOM().selection = myArray;
```

document.selectNone()

Disponibilidad

Flash MX 2004.

Uso

```
document.selectNone()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; desactiva la selección de los elementos seleccionados.

Ejemplo

El ejemplo siguiente desactiva la selección de los elementos que estén seleccionados:

```
fl.getDocumentDOM().selectNone();
```

Véase también

`document.selectAll()`, `document.selection`

document.setAlignToDocument()

Disponibilidad

Flash MX 2004.

Uso

```
document.setAlignToDocument( bToStage )
```

Parámetros

`bToStage` Un valor booleano que, si se define como `true`, alinea los objetos con el escenario. Si se define como `false`, no los alinea.

Valor devuelto

Ninguno.

Descripción

Método; establece las preferencias de `document.align()`, `document.distribute()`, `document.match()` y `document.space()` para que actúen sobre el documento. Este método equivale a activar el botón En escenario en el panel Alinear.

Ejemplo

El ejemplo siguiente activa el botón En escenario del panel Alinear para alinear objetos con el escenario:

```
fl.getDocumentDOM().setAlignToDocument(true);
```

Véase también

`document.getAlignToDocument()`

document.setBlendMode()

Disponibilidad

Flash 8.

Uso

```
document.setBlendMode( mode )
```

Parámetros

mode Una cadena que representa el modo de mezcla deseado para los objetos seleccionados. Los valores aceptables son: `"normal"`, `"layer"`, `"multiply"`, `"screen"`, `"overlay"`, `"hardlight"`, `"lighten"`, `"darken"`, `"difference"`, `"add"`, `"subtract"`, `"invert"`, `"alpha"` y `"erase"`.

Valor devuelto

Ninguno.

Descripción

Método; establece el modo de mezcla para los objetos seleccionados.

Ejemplo

El ejemplo siguiente establece el modo de mezcla para el objeto seleccionado como `"add"`.

```
fl.getDocumentDOM().setBlendMode("add");
```

Véase también

`document.addFilter()`, `document.setFilterProperty()`, `symbolInstance.blendMode`

document.setCustomFill()

Disponibilidad

Flash MX 2004.

Uso

```
document.setCustomFill( fill )
```

Parámetros

fill Un objeto Fill que especifica la configuración de relleno que se va a utilizar. Véase Objeto Fill.

Valor devuelto

Ninguno.

Descripción

Método; establece la configuración de relleno para el panel Herramientas, el inspector de propiedades y cualquier forma seleccionada. Permite que un script establezca la configuración de relleno antes de dibujar el objeto, en lugar de dibujar el objeto, seleccionándolo y cambiando la configuración de relleno. También permite que un script cambie la configuración de relleno del panel Herramientas y del inspector de propiedades.

Ejemplo

El ejemplo siguiente cambia a blanco el color de la muestra de color de relleno en el panel Herramientas, el inspector de propiedades y las formas seleccionadas:

```
var fill = fl.getDocumentDOM().getCustomFill();
fill.color = '#FFFFFF';
fill.style = "solid";
fl.getDocumentDOM().setCustomFill(fill);
```

Véase también

`document.getCustomFill()`

document.setCustomStroke()

Disponibilidad

Flash MX 2004.

Uso

```
document.setCustomStroke( stroke )
```

Parámetros

stroke Un Objeto Stroke.

Valor devuelto

Ninguno.

Descripción

Método; establece la configuración de trazo para el panel Herramientas, el inspector de propiedades y cualquier forma seleccionada. Permite que un script establezca la configuración de trazo antes de dibujar el objeto, en lugar de dibujar el objeto, seleccionándolo y cambiando la configuración de relleno. También permite que un script cambie la configuración de trazo del panel Herramientas y del inspector de propiedades.

Ejemplo

El ejemplo siguiente cambia la configuración de grosor del trazo en el panel Herramientas, el inspector de propiedades y las formas seleccionadas:

```
var stroke = fl.getDocumentDOM().getCustomStroke();
stroke.thickness += 2;
fl.getDocumentDOM().setCustomStroke(stroke);
```

Véase también

`document.getCustomStroke()`

document.setElementProperty()

Disponibilidad

Flash MX 2004.

Uso

`document.setElementProperty( property, value )`

Parámetros

property Una cadena que especifica el nombre de la propiedad Element que se va a establecer. Para ver una lista completa de propiedades y valores, consulte "Resumen de propiedades del objeto Element" en la página 198.

NOTA	No es posible utilizar este método para establecer valores para propiedades de sólo lectura, como `element.elementType`, `element.top` y `element.left`.

value Un entero que especifica el valor que se va a establecer en la propiedad Element especificada.

Valor devuelto

Ninguno.

Descripción

Método; establece la propiedad `Element` especificada en el objeto u objetos seleccionados en el documento. Este método no tiene ningún efecto si no hay selección.

Ejemplo

El ejemplo siguiente establece la anchura de todos los objetos seleccionados en 100 y la altura en 50:

```
fl.getDocumentDOM().setElementProperty("width", 100);
fl.getDocumentDOM().setElementProperty("height", 50);
```

document.setElementTextAttr()

Disponibilidad

Flash MX 2004.

Uso

```
document.setElementTextAttr( attrName, attrValue [, startIndex [,
  endIndex]] )
```

Parámetros

`attrName` Una cadena que especifica el nombre de la propiedad `TextAttrs` que se va a cambiar.

`attrValue` El valor con el que se define la propiedad `TextAttrs`. Para ver una lista de nombres de propiedades y valores posibles, consulte "Resumen de propiedades del objeto TextAttrs" en la página 464.

`startIndex` Un valor entero que especifica el índice del primer carácter que está afectado. Este parámetro es opcional.

`endtIndex` Un valor entero que especifica el índice del último carácter que está afectado. Este parámetro es opcional.

Valor devuelto

Un valor booleano: `true` si ha cambiado al menos una propiedad de atributo de texto; `false` en caso contrario.

Descripción

Método; establece la propiedad `textAttrs` especificada de los elementos de texto seleccionados con el valor especificado. Para ver una lista de nombres de propiedades y valores permitidos, consulte "Resumen de propiedades del objeto TextAttrs" en la página 464. Si no se transfieren los parámetros opcionales, el método establece el estilo del rango de texto seleccionado actualmente o de todo el campo de texto si no hay texto seleccionado. Si sólo se transfiere *`startIndex`*, el método establece los atributos del carácter. Si se transfieren *`startIndex`* y *`endIndex`*, el método establece los atributos de los caracteres comenzando desde *`startIndex`* hasta *`endIndex`* (no incluido). Si se especifican estilos de párrafo, se verán afectados todos los que pertenezcan al rango.

Ejemplo

Los ejemplos siguientes establecen los atributos de texto `fillColor`, `italic` y `bold` para los elementos de texto seleccionados:

```
var success = fl.getDocumentDOM().setElementTextAttr("fillColor",
  "#00ff00");
var pass = fl.getDocumentDOM().setElementTextAttr("italic", true, 10);
var ok = fl.getDocumentDOM().setElementTextAttr("bold", true, 5, 15);
```

document.setFillColor()

Disponibilidad

Flash MX 2004.

Uso

```
document.setFillColor( color )
```

Parámetros

`color` El color del relleno, en uno de los formatos siguientes:

- Una cadena con el formato `"#RRGGBB"` o `"#RRGGBBAA"`
- Un número hexadecimal con el formato `0xRRGGBB`
- Un entero que representa el equivalente decimal del número hexadecimal

Si se define como `null`, no se establece ningún color de relleno, lo que equivale a definir la muestra de color de relleno en la interfaz de usuario como Sin relleno.

Valor devuelto

Ninguno.

Descripción

Método; cambia el color de relleno de la selección al especificado. Para obtener información sobre cómo cambiar el color de relleno en el panel Herramientas y el inspector de propiedades, consulte `document.setCustomFill()`.

Ejemplo

Las tres primeras sentencias del ejemplo siguiente establecen el color de relleno utilizando cada uno de los distintos formatos para especificar color. La cuarta sentencia establece el relleno como Sin relleno.

```
flash.getDocumentDOM().setFillColor("#cc00cc");
flash.getDocumentDOM().setFillColor(0xcc00cc);
flash.getDocumentDOM().setFillColor(120000);
flash.getDocumentDOM().setFillColor(null);
```

document.setFilterProperty()

Disponibilidad

Flash 8.

Uso

```
document.setFilterProperty( property, filterIndex, value )
```

Parámetros

`property` Una cadena que especifica la propiedad que se va a establecer. Los valores aceptables son: `"blurX"`, `"blurY"`, `"quality"`, `angle"`, `"distance"`, `"strength"`, `"knockout"`, `"inner"`, `"bevelType"`, `"color"`, `"shadowColor"` y `"highlightColor"`.

`filterIndex` Un entero que especifica el índice basado en cero del filtro en la lista Filtros.

`value` Un número o cadena que especifica el valor que se va a establecer para la propiedad de filtro especificada. Los valores válidos dependen de la propiedad y del filtro que se definen.

Valor devuelto

Ninguno.

Descripción

Método; establece una propiedad de filtro especificada para el objeto u objetos seleccionados actualmente que admiten la propiedad de filtro.

Ejemplo

El ejemplo siguiente define la propiedad `quality` como 2 para el segundo filtro (valor de índice de 1) de la lista Filtros de los objetos seleccionados y, a continuación, define la propiedad `shadowColor` del primer filtro de la lista Filtros en el objeto u objetos seleccionados:

```
fl.getDocumentDOM().setFilterProperty("quality", 1, 2);
fl.getDocumentDOM().setFilterProperty("shadowColor", 0, "#FF00FF");
```

Véase también

`document.addFilter()`, `document.getFilters()`, `document.setBlendMode()`, `document.setFilters()`, Objeto Filter

document.setFilters()

Disponibilidad

Flash 8.

Uso

```
document.setFilters( filterArray )
```

Parámetros

`filterArray` La matriz de filtros especificada actualmente.

Valor devuelto

Ninguno.

Descripción

Método; aplica filtros a los objetos seleccionados. Utilice este método después de llamar `document.getFilters()` y realizar los cambios deseados en los filtros.

Ejemplo

El ejemplo siguiente obtiene los filtros del objeto seleccionado y define la propiedad `blurX` de todos los filtros de desenfoque como 50.

```
var myFilters = fl.getDocumentDOM().getFilters();
for (i=0; i < myFilters.length; i++) {
  if (myFilters[i].name == "blurFilter"){
    myFilters[i].blurX = 50;
  }
}
fl.getDocumentDOM().setFilters(myFilters);
```

Véase también

`document.addFilter()`, `document.getFilters()`, `document.setFilterProperty()`, Objeto Filter

document.setInstanceAlpha()

Disponibilidad

Flash MX 2004.

Uso

```
document.setInstanceAlpha( opacity )
```

Parámetros

`opacity` Un entero entre 0 (transparente) y 100 (completamente saturado) que ajusta la transparencia de la instancia.

Valor devuelto

Ninguno.

Descripción

Método; establece la opacidad de la instancia.

Ejemplo

El ejemplo siguiente establece la opacidad de la tinta con un valor de 50:

```
fl.getDocumentDOM().setInstanceAlpha(50);
```

document.setInstanceBrightness()

Disponibilidad

Flash MX 2004.

Uso

```
document.setInstanceBrightness( brightness )
```

Parámetros

`brightness` Un entero que especifica el brillo como un valor entre -100 (negro) y 100 (blanco).

Valor devuelto

Ninguno.

Descripción

Método; establece el brillo de la instancia.

Ejemplo

El ejemplo siguiente establece el brillo de la instancia con un valor de 50:

```
fl.getDocumentDOM().setInstanceBrightness(50);
```

document.setInstanceTint()

Disponibilidad

Flash MX 2004.

Uso

```
document.setInstanceTint( color, strength )
```

Parámetros

color El color de la tinta, en uno de los formatos siguientes:

- Una cadena con el formato "#RRGGBB" o "#RRGGBBAA"
- Un número hexadecimal con el formato 0xRRGGBB
- Un entero que representa el equivalente decimal del número hexadecimal

Este parámetro equivale a seleccionar el valor Color: Tinta para un símbolo en el inspector de propiedades.

strength Un entero entre 0 y 100 que especifica la opacidad de la tinta.

Valor devuelto

Ninguno.

Descripción

Método; establece la tinta de la instancia.

Ejemplo

El ejemplo siguiente define la tinta para la instancia seleccionada como roja con un valor de opacidad de 50:

```
fl.getDocumentDOM().setInstanceTint(0xff0000, 50);
```

document.setMetadata()

Disponibilidad

Flash 8.

Uso

```
document.setMetadata( strMetadata )
```

Parámetros

strMetadata Una cadena que contiene los metadatos XML que se van a asociar al documento. Para más información, consulte la siguiente descripción.

Valor devuelto

Un valor booleano: `true` si es correcto y `false` en caso contrario.

Descripción

Método; establece los metadatos XML para el documento especificado, sobrescribiendo los metadatos existentes. El XML transferido como *strMetadata* se valida y se puede reescribir antes de almacenar. Si no se puede validar como código XML válido o infringe reglas específicas, los metadatos XML no se definen y se devuelve `false`. (Si se devuelve `false`, no hay forma de obtener información más detallada sobre el error.)

NOTA	Aunque se devuelva true, es posible que el código XML definido no sea exactamente la misma cadena que se transfirió. Para obtener el valor exacto en el que se definió XML, utilice `document.getMetadata()`.

El formato de los metadatos es RDF compatible con la especificación XMP. Para más información sobre RDF y XMP, consulte las siguientes fuentes:

- RDF Primer en www.w3.org/TR/rdf-primer/
- La especificación de RDF en www.w3.org/TR/1999/REC-rdf-syntax-19990222/
- La página de inicio de XMP en www.adobe.com/products/xmp/

Ejemplo

Los ejemplos siguientes muestran distintas formas válidas de representar los mismos datos. En todos estos casos salvo en el segundo, si los datos se enviaran a `Document.setMetadata()`, no se reescribirían (aparte de eliminar los saltos de línea).

En el primer ejemplo, los metadatos están en etiquetas, con distintos esquemas colocados en etiquetas `rdf:Description` independientes:

```
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
  <rdf:Description rdf:about='' xmlns:dc='http://purl.org/dc/1.1/'>
    <dc:title>Simple title</dc:title>
    <dc:description>Simple description</dc:description>
  </rdf:Description>
  <rdf:Description rdf:about='' xmlns:xmp='http://ns.adobe.com/xap/1.0/'>
    <xmp:CreateDate>2004-10-12T10:29-07:00</xmp:CreateDate>
    <xmp:CreatorTool>Flash Authoring WIN 8,0,0,215</xmp:CreatorTool>
  </rdf:Description>
</rdf:RDF>
```

En el segundo ejemplo, los metadatos están en etiquetas, pero con distintos esquemas en una etiqueta `rdf:Description`. Este ejemplo también incluye comentarios, que `Document.setMetadata()` ignorará y descartará:

```
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
  <!-- Esto es antes de la primera etiqueta rdf:Description -->
  <rdf:Description rdf:about='' xmlns:dc='http://purl.org/dc/1.1/'>
    <dc:title>Simple title</dc:title>
    <dc:description>Simple description</dc:description>
  </rdf:Description>
  <!-- Esto es entre las dos etiquetas rdf:Description -->
  <rdf:Description rdf:about='' xmlns:xmp='http://ns.adobe.com/xap/1.0/'>
    <xmp:CreateDate>2004-10-12T10:29-07:00</xmp:CreateDate>
    <xmp:CreatorTool>Flash Authoring WIN 8,0,0,215</xmp:CreatorTool>
  </rdf:Description>
  <!-- Esto es después de la segunda etiqueta rdf:Description -->
</rdf:RDF>
```

En el tercer ejemplo, los metadatos están en atributos y los distintos esquemas en una sola etiqueta `rdf:Description`:

```
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
  <rdf:Description rdf:about='' xmlns:dc='http://purl.org/dc/1.1/'
  dc:title='Simple title'
dc:description='Simple description' />
  <rdf:Description rdf:about='' xmlns:xmp='http://ns.adobe.com/xap/1.0/'
xmp:CreateDate='2004-10-12T10:29-07:00' xmp:CreatorTool='Flash Authoring
  WIN 8,0,0,215' />
</rdf:RDF>
```

Véase también

`document.getMetadata()`

document.setSelectionBounds()

Disponibilidad

Flash MX 2004; parámetro *bContactSensitiveSelection* añadido en Flash 8.

Uso

```
document.setSelectionBounds(boundingRectangle [,
  bContactSensitiveSelection])
```

Parámetros

boundingRectangle Un rectángulo que especifica la nueva ubicación y el tamaño de la selección. Para más información sobre el formato de *boundingRectangle*, consulte `document.addNewRectangle()`.

bContactSensitiveSelection Un valor booleano que especifica si el modo de selección Por contacto está activado (`true`) o desactivado (`false`) durante la selección de objetos. El valor predeterminado es `false`.

Valor devuelto

Ninguno.

Descripción

Método; mueve y cambia el tamaño de la selección en una única operación.

Si pasa un valor para *bContactSensitiveSelection*, sólo es válido para este método y no afecta al modo de selección Por contacto del documento (consulte `fl.contactSensitiveSelection`).

Ejemplo

El ejemplo siguiente mueve la selección actual a 10, 20 y cambia el tamaño a 100, 200:

```
var l = 10;
var t = 20;
fl.getDocumentDOM().setSelectionBounds({left:l, top:t, right:(100+l),
  bottom:(200+t)});
```

Véase también

`document.selection`, `document.setSelectionRect()`

document.setSelectionRect()

Disponibilidad

Flash MX 2004; parámetro *bContactSensitiveSelection* añadido en Flash 8.

Uso

```
document.setSelectionRect(rect [, bReplaceCurrentSelection
  [, bContactSensitiveSelection]])
```

Parámetros

rect Un objeto rectángulo para definir como seleccionado. Para más información sobre el formato de *rect*, consulte `document.addNewRectangle()`.

bReplaceCurrentSelection Un valor booleano que especifica si el método reemplaza la selección actual (`true`) o se añade a la selección actual (`false`). El valor predeterminado es `true`.

bContactSensitiveSelection Un valor booleano que especifica si el modo de selección Por contacto está activado (`true`) o desactivado (`false`) durante la selección de objetos. El valor predeterminado es `false`.

Valor devuelto

Ninguno.

Descripción

Método; dibuja un recuadro de delimitación rectangular en relación con el escenario, empleando las coordenadas especificadas. Es distinto de `document.getSelectionRect()`, en el que el rectángulo es relativo al objeto que se está editando.

Este método equivale a arrastrar un rectángulo con una herramienta Flecha. Para que se seleccione, una instancia debe estar completamente encerrada por el rectángulo.

Si pasa un valor para *bContactSensitiveSelection*, sólo es válido para este método y no afecta al modo de selección Por contacto del documento (consulte `fl.contactSensitiveSelection`).

NOTA	Si repite `setSelectionRect()` empleando el panel Historial o el elemento de menú, se repetirá el paso anterior a la operación `setSelectionRect()`.

Ejemplo

En el ejemplo siguiente, la segunda selección reemplaza a la primera:

```
fl.getDocumentDOM().setSelectionRect({left:1, top:1, right:200,
  bottom:200});
fl.getDocumentDOM().setSelectionRect({left:364.0, top:203.0, right:508.0,
  bottom:434.0}, true);
```

En el ejemplo siguiente, la segunda selección se añade a la primera. Tiene el mismo efecto que la operación manual de mantener presionada la tecla Mayús y seleccionar un segundo objeto.

```
fl.getDocumentDOM().setSelectionRect({left:1, top:1, right:200,
  bottom:200});
fl.getDocumentDOM().setSelectionRect({left:364.0, top:203.0, right:508.0,
  bottom:434.0}, false);
```

Véase también

`document.getSelectionRect()`, `document.selection`, `document.setSelectionBounds()`

document.setStroke()

Disponibilidad

Flash MX 2004.

Uso

`document.setStroke( color, size, strokeType )`

Parámetros

`color` El color del trazo, en uno de los formatos siguientes:

- Una cadena con el formato `"#RRGGBB"` o `"#RRGGBBAA"`
- Un número hexadecimal con el formato `0xRRGGBB`
- Un entero que representa el equivalente decimal del número hexadecimal

`size` Un valor de coma flotante que especifica el nuevo tamaño de trazo para la selección.

`strokeType` Una cadena que especifica el nuevo tipo de trazo para la selección. Los valores aceptables son: `"hairline"`, `"solid"`, `"dashed"`, `"dotted"`, `"ragged"`, `"stipple"` y `"hatched"`.

Valor devuelto

Ninguno.

Descripción

Método; establece el color, el ancho y el estilo de los trazos seleccionados. Para obtener información sobre cómo cambiar el trazo en el panel Herramientas y el inspector de propiedades, consulte `document.setCustomStroke()`.

Ejemplo

El ejemplo siguiente define el color del trazo como rojo, el tamaño como 3,25 y el tipo como líneas discontinuas:

```
fl.getDocumentDOM().setStroke("#ff0000", 3.25, "dashed");
```

document.setStrokeColor()

Disponibilidad

Flash MX 2004.

Uso

```
document.setStrokeColor( color )
```

Parámetros

`color` El color del trazo, en uno de los formatos siguientes:

- Una cadena con el formato `"#RRGGBB"` o `"#RRGGBBAA"`
- Un número hexadecimal con el formato `0xRRGGBB`
- Un entero que representa el equivalente decimal del número hexadecimal

Valor devuelto

Ninguno.

Descripción

Método; cambia el color de trazo de la selección al especificado. Para obtener información sobre cómo cambiar el trazo en el panel Herramientas y el inspector de propiedades, consulte `document.setCustomStroke()`.

Ejemplo

Las tres sentencias del ejemplo siguiente establecen el color de trazo utilizando cada uno de los distintos formatos para especificar color:

```
flash.getDocumentDOM().setStrokeColor("#cc00cc");
flash.getDocumentDOM().setStrokeColor(0xcc00cc);
flash.getDocumentDOM().setStrokeColor(120000);
```

document.setStrokeSize()

Disponibilidad

Flash MX 2004.

Uso

```
document.setStrokeSize( size )
```

Parámetros

size Un valor de coma flotante de 0,25 a 10 que especifica el tamaño de trazo. El método ignora las precisiones mayores que dos posiciones decimales.

Valor devuelto

Ninguno.

Descripción

Método; cambia el tamaño de trazo de la selección al especificado. Para obtener información sobre cómo cambiar el trazo en el panel Herramientas y el inspector de propiedades, consulte `document.setCustomStroke()`.

Ejemplo

El ejemplo siguiente cambia el tamaño de trazo de la selección a 5:

```
fl.getDocumentDOM().setStrokeSize(5);
```

document.setStrokeStyle()

Disponibilidad

Flash MX 2004.

Uso

```
document.setStrokeStyle( strokeType )
```

Parámetros

strokeType Una cadena que especifica el estilo de trazo para la selección actual. Los valores aceptables son: `"hairline"`, `"solid"`, `"dashed"`, `"dotted"`, `"ragged"`, `"stipple"` y `"hatched"`.

Valor devuelto

Ninguno.

Descripción

Método; cambia el estilo de trazo de la selección al especificado. Para obtener información sobre cómo cambiar el trazo en el panel Herramientas y el inspector de propiedades, consulte `document.setCustomStroke()`.

Ejemplo

El ejemplo siguiente cambia el estilo de trazo de la selección a `"dashed"`:

```
fl.getDocumentDOM().setStrokeStyle("dashed");
```

document.setTextRectangle()

Disponibilidad

Flash MX 2004.

Uso

```
document.setTextRectangle( boundingRectangle )
```

Parámetros

boundingRectangle Un objeto de rectángulo de texto que especifica el nuevo tamaño dentro del cual el elemento de texto debe fluir. Para obtener información sobre el formato de *boundingRectangle*, consulte `document.addNewRectangle()`.

Valor devuelto

Un valor booleano: `true` si ha cambiado el tamaño de al menos un campo de texto, y `false` en caso contrario.

Descripción

Método; cambia el rectángulo de delimitación para el elemento de texto seleccionado al tamaño especificado. Este método hace que el texto vuelva a fluir dentro del nuevo rectángulo. El elemento de texto no se escala ni transforma. Los valores transferidos en *boundingRectangle* se emplean de la forma siguiente:

- Si el texto es horizontal y estático, el método sólo tiene en cuenta el valor de anchura transferido en *boundingRectangle*; la altura se calcula automáticamente para que quepa todo el texto.
- Si el texto es vertical (y, por tanto, estático), el método sólo tiene en cuenta el valor de altura transferido en *boundingRectangle*; la anchura se calcula automáticamente para que quepa todo el texto.
- Si el texto es dinámico o de entrada, el método tiene en cuenta los valores de anchura y altura transferidos en *boundingRectangle* y el rectángulo resultante podría ser mayor de lo necesario para que encaje todo el texto. Sin embargo, si los parámetros especifican un tamaño de rectángulo que es demasiado pequeño para que encaje todo el texto, el método sólo tiene en cuenta el valor de anchura transferido en *boundingRectangle* (la altura se calcula automáticamente para que quepa todo el texto).

Ejemplo

El ejemplo siguiente cambia el tamaño del rectángulo de texto de delimitación a las dimensiones especificadas:

```
fl.getDocumentDOM().setTextRectangle({left:0, top:0, right:50, bottom:200})
```

document.setTextSelection()

Disponibilidad

Flash MX 2004.

Uso

```
document.setTextSelection( startIndex, endIndex )
```

Parámetros

startIndex Un entero que especifica la posición del primer carácter que se va a seleccionar. La posición del primer carácter es 0 (cero).

endIndex Un entero que especifica la posición final de la selección hasta *endIndex* (no incluido). La posición del primer carácter es 0 (cero).

Valor devuelto

Un valor booleano: `true` si el método puede establecer correctamente la selección de texto, y `false` en caso contrario.

Descripción

Método; establece la selección de texto del campo de texto seleccionado actualmente con los valores especificados por los valores *`startIndex`* y *`endIndex`*. Se activará la edición de texto, si aún no lo está.

Ejemplo

El ejemplo siguiente selecciona el texto desde el carácter 6º hasta el carácter 25º.

```
fl.document.setTextSelection(5, 25);
```

document.setTextString()

Disponibilidad

Flash MX 2004.

Uso

```
document.setTextString( text [, startIndex [, endIndex]] )
```

Parámetros

`text` Una cadena de caracteres que se va a insertar en el campo de texto.

`startIndex` Un entero que especifica el primer carácter que se va a reemplazar. La posición del primer carácter es 0 (cero). Este parámetro es opcional.

`endIndex` Un entero que especifica el último carácter que se va a reemplazar. La posición del primer carácter es 0 (cero). Este parámetro es opcional.

Valor devuelto

Un valor booleano: `true` si se ha definido el texto de al menos una cadena de texto, y `false` en caso contrario.

Descripción

Método; inserta una cadena de texto. Si no se transfieren los parámetros opcionales, se reemplazará la selección de texto existente; si el objeto de texto no se está editando actualmente, se reemplazará la cadena de texto completa. Si sólo se transfiere *`startIndex`*, la cadena transferida se insertará en esta posición. Si se transfieren *`startIndex`* y *`endIndex`*, la cadena transferida reemplazará el segmento de texto que comienza desde *`startIndex`* hasta *`endIndex`* (no incluido).

Ejemplo

El ejemplo siguiente reemplaza la selección de texto actual con “Hello World”:

```
var success = fl.getDocumentDOM().setTextString("Hello World!");
```

El ejemplo siguiente inserta “hello” en la posición 6 de la selección de texto actual:

```
var pass = fl.getDocumentDOM().setTextString("hello", 6);
```

El ejemplo siguiente inserta “Howdy” comenzando en la posición 2 hasta la posición 7 (no incluida) de la selección de texto actual:

```
var ok = fl.getDocumentDOM().setTextString("Howdy", 2, 7);
```

Véase también

`document.getTextString()`

document.setTransformationPoint()

Disponibilidad

Flash MX 2004.

Uso

```
document.setTransformationPoint( transformationPoint )
```

Parámetros

`transformationPoint` Un par de números de coma flotante que especifica valores para cada uno de los siguientes elementos:

- Formas: *`transformationPoint`* se define en relación con el documento. 0,0 es igual que el escenario (esquina superior izquierda).
- Símbolos: *`transformationPoint`* se define en relación con el punto de registro del símbolo. 0,0 se encuentra en el punto de registro.
- Texto: *`transformationPoint`* se define en relación con el campo de texto. 0,0 es la esquina superior izquierda del campo de texto.
- Mapas de bits/vídeos: *`transformationPoint`* se define en relación con el mapa de bits/vídeo. 0,0 es la esquina superior izquierda del mapa de bits o del vídeo.
- Grupos: *`transformationPoint`* se define en relación con el documento. 0,0 es igual que el escenario (esquina superior izquierda).

Valor devuelto

Ninguno.

Descripción

Método; mueve el punto de transformación de la selección actual.

Ejemplo

El ejemplo siguiente establece el punto de transformación de la selección actual en 100, 200:

```
fl.getDocumentDOM().setTransformationPoint({x:100, y:200});
```

Véase también

`document.getTransformationPoint()`

document.silent

Disponibilidad

Flash MX 2004.

Uso

```
document.silent
```

Descripción

Propiedad; un valor booleano que especifica si el objeto es accesible. Equivale a la lógica inversa de la opción Permitir acceso a la película del panel Accesibilidad. Es decir, si `document.silent` es `true`, equivale a la opción desactivada Permitir acceso a la película. Si es `false`, equivale a la opción activada Permitir acceso a la película.

Ejemplo

El ejemplo siguiente define la variable `isSilent` con el valor de la propiedad `silent`:

```
var isSilent = fl.getDocumentDOM().silent;
```

El ejemplo siguiente establece la propiedad `silent` como `false`, lo que indica que el documento es accesible:

```
fl.getDocumentDOM().silent = false;
```

document.skewSelection()

Disponibilidad

Flash MX 2004.

Uso

```
document.skewSelection( xSkew, ySkew [, whichEdge] )
```

Parámetros

xSkew Un número de coma flotante que especifica la cantidad que se va a sesgar respecto del eje *x*, medida en grados.

ySkew Un número de coma flotante que especifica la cantidad que se va a sesgar respecto del eje *y*, medida en grados.

whichEdge Una cadena que especifica el borde donde se produce la transformación. Si se omite, el sesgo se produce en el punto de transformación. Los valores aceptables son: `"top center"`, `"right center"`, `"bottom center"` y `"left center"`. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; sesga la selección en la cantidad especificada. Se obtiene el mismo efecto que con la herramienta Transformación libre para sesgar el objeto.

Ejemplo

Los ejemplos siguientes sesgan el objeto seleccionado 2,0 en vertical y 1,5 en horizontal. El segundo ejemplo transforma el objeto en el borde superior central:

```
flash.getDocumentDOM().skewSelection(2.0, 1.5);
flash.getDocumentDOM().skewSelection(2.0, 1.5, "top center");
```

document.smoothSelection()

Disponibilidad

Flash MX 2004.

Uso

```
document.smoothSelection()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; suaviza la curva de cada línea curva o contorno de relleno seleccionado. Este método realiza la misma acción que el botón Suavizar del panel Herramientas.

Ejemplo

El ejemplo siguiente suaviza la curva de la selección actual:

```
fl.getDocumentDOM().smoothSelection();
```

document.space()

Disponibilidad

Flash MX 2004.

Uso

```
document.space( direction [, bUseDocumentBounds] )
```

Parámetros

direction Una cadena que especifica la dirección en la que se distribuyen los objetos en la selección. Los valores aceptables son: `"horizontal"` o `"vertical"`.

bUseDocumentBounds Un valor booleano que, si se define como `true`, distribuye los objetos en los límites del documento. En caso contrario, el método utiliza los límites de los objetos seleccionados. El valor predeterminado es `false`. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; distribuye los objetos de la selección de manera uniforme.

Ejemplo

El ejemplo siguiente distribuye los objetos horizontalmente en relación con el escenario:

```
fl.getDocumentDOM().space("horizontal",true);
```

El ejemplo siguiente distribuye los objetos horizontalmente en relación mutua:

```
fl.getDocumentDOM().space("horizontal");
```

El ejemplo siguiente distribuye los objetos horizontalmente en relación mutua, con *bUseDcoumentBounds* definido expresamente como `false`:

```
fl.getDocumentDOM().space("horizontal",false);
```

Véase también

`document.getAlignToDocument()`, `document.setAlignToDocument()`

document.straightenSelection()

Disponibilidad

Flash MX 2004.

Uso

```
document.straightenSelection()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; endereza los trazos seleccionados actualmente. Este método equivale a utilizar el botón Enderezar del panel Herramientas.

Ejemplo

El ejemplo siguiente endereza la curva de la selección actual:

```
fl.getDocumentDOM().straightenSelection();
```

document.swapElement()

Disponibilidad

Flash MX 2004.

Uso

```
document.swapElement( name )
```

Parámetros

name Una cadena que especifica el nombre del elemento de biblioteca que se va a utilizar.

Valor devuelto

Ninguno.

Descripción

Método; cambia la selección actual por la especificada. La selección debe contener un gráfico, botón, clip de película, vídeo o mapa de bits. Este método muestra un mensaje de error si no se selecciona ningún objeto o no se encuentra el objeto dado.

Ejemplo

El ejemplo siguiente cambia la selección actual por `Symbol 1` de la biblioteca:

```
fl.getDocumentDOM().swapElement('Symbol 1');
```

document.swapStrokeAndFill()

Disponibilidad

Flash 8.

Uso

```
document.swapStrokeAndFill();
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; intercambia los colores de Trazo y Relleno.

Ejemplo

El siguiente ejemplo intercambia los colores de Trazo y Relleno en el documento actual:

```
fl.getDocumentDOM().swapStrokeAndFill();
```

document.testMovie()

Disponibilidad

Flash MX 2004.

Uso

```
document.testMovie()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; ejecuta una operación Probar película en el documento.

Ejemplo

El ejemplo siguiente prueba la película para el documento actual:

```
fl.getDocumentDOM().testMovie();
```

Véase también

`document.canTestMovie()`, `document.testScene()`

document.testScene()

Disponibilidad

Flash MX 2004.

Uso

```
document.testScene()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; ejecuta una operación Probar escena en la escena actual del documento.

Ejemplo

El ejemplo siguiente prueba la escena actual en el documento:

```
fl.getDocumentDOM().testScene();
```

Véase también

`document.canTestScene()`, `document.testMovie()`

document.timelines

Disponibilidad

Flash MX 2004.

Uso

```
document.timelines
```

Descripción

Propiedad de sólo lectura; una matriz de objetos Timeline (consulte Objeto Timeline).

Ejemplo

El ejemplo siguiente obtiene la matriz de las líneas de tiempo actuales en el documento activo y muestra sus nombres en el panel Salida:

```
var i = 0;
var curTimelines = fl.getDocumentDOM().timelines;
while(i < fl.getDocumentDOM().timelines.length){
  alert(curTimelines[i].name);
  ++i;
}
```

Véase también

`document.currentTimeline`, `document.getTimeline()`

document.traceBitmap()

Disponibilidad

Flash MX 2004.

Uso

`document.traceBitmap( threshold, minimumArea, curveFit, cornerThreshold )`

Parámetros

`threshold` Un entero que controla el número de colores del mapa de bits trazado. Los valores aceptables son enteros entre 0 y 500.

`minimumArea` Un entero que especifica el radio medido en píxeles. Los valores aceptables son enteros entre 1 y 1000.

`curveFit` Una cadena que especifica con qué suavidad se dibujan los contornos. Los valores aceptables son: `"pixels"`, `"very tight"`, `"tight"`, `"normal"`, `"smooth"` y `"very smooth"`.

cornerThreshold Una cadena que es similar a *curveFit*, aunque corresponde a las esquinas de la imagen de mapa de bits. Los valores aceptables son: `"many corners"`, `"normal"` y `"few corners"`.

Valor devuelto

Ninguno.

Descripción

Método; realiza un mapa de bits de traza en la selección actual. Este método equivale a seleccionar Modificar > Mapa de bits > Trazar mapa de bits.

Ejemplo

El ejemplo siguiente traza el mapa de bits seleccionado empleando los parámetros especificados:

```
fl.getDocumentDOM().traceBitmap(0, 500, 'normal', 'normal');
```

document.transformSelection()

Disponibilidad

Flash MX 2004.

Uso

```
document.transformSelection( a, b, c, d)
```

Parámetros

a Un número de coma flotante que especifica el elemento (0,0) de la matriz de transformación.

b Un número de coma flotante que especifica el elemento (0,1) de la matriz de transformación.

c Un número de coma flotante que especifica el elemento (1,0) de la matriz de transformación.

d Un número de coma flotante que especifica el elemento (1,1) de la matriz de transformación.

Valor devuelto

Ninguno.

Descripción

Método; realiza una transformación general en la selección actual aplicando la matriz especificada en los argumentos. Para más información, consulte la propiedad `element.matrix`.

Ejemplo

El ejemplo siguiente amplía la selección en un factor de 2 en la dirección x:

```
fl.getDocumentDOM().transformSelection(2.0, 0.0, 0.0, 1.0);
```

document.unGroup()

Disponibilidad

Flash MX 2004.

Uso

```
document.unGroup()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; desagrupa la selección actual.

Ejemplo

El ejemplo siguiente desagrupa los elementos de la selección actual:

```
fl.getDocumentDOM().unGroup();
```

Véase también

`document.group()`

document.union()

Disponibilidad

Flash 8.

Uso

```
document.union()
```

Parámetros

Ninguno.

Valor devuelto

Un valor booleano: `true` si es correcto y `false` en caso contrario.

Descripción

Método; combina todas las formas seleccionadas en un objeto de dibujo.

Ejemplo

El ejemplo siguiente combina todas las formas seleccionadas en un objeto de dibujo:

```
fl.getDocumentDOM().union();
```

Véase también

`document.crop()`, `document.deleteEnvelope()`, `document.intersect()`, `document.punch()`, `shape.isDrawingObject`

document.unlockAllElements()

Disponibilidad

Flash MX 2004.

Uso

```
document.unlockAllElements()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; desbloquea todos los elementos bloqueados en el fotograma seleccionado actualmente.

Ejemplo

El ejemplo siguiente desbloquea todos los objetos bloqueados del fotograma actual:

```
fl.getDocumentDOM().unlockAllElements();
```

Véase también

`element.locked`

document.viewMatrix

Disponibilidad

Flash MX 2004.

Uso

```
document.viewMatrix
```

Descripción

Propiedad de sólo lectura; un objeto Matrix. `viewMatrix` se emplea para cambiar del espacio de objeto al espacio de documento cuando el documento se encuentra en modo de edición. La ubicación del ratón, como la recibe una herramienta, es relativa al objeto que se está editando. Véase Objeto Matrix.

Por ejemplo, si crea un símbolo, hace doble clic en él para editarlo y dibuja con la herramienta PolyStar, el punto (0,0) estará en el punto de registro del símbolo. Sin embargo, el objeto drawingLayer espera valores en el espacio de documento, por lo que si dibuja una línea desde (0,0) empleando drawingLayer, comenzará en la esquina superior izquierda del escenario. `viewMatrix` permite cambiar del espacio del objeto que se está editando al espacio de documento.

Ejemplo

El ejemplo siguiente obtiene el valor de la propiedad `viewMatrix`:

```
var mat = fl.getDocumentDOM().viewMatrix;
```

document.width

Disponibilidad

Flash MX 2004.

Uso

```
document.width
```

Descripción

Propiedad; un entero que especifica la anchura del documento (escenario) en píxeles.

Ejemplo

El ejemplo siguiente establece la anchura del escenario en 400 píxeles:

```
fl.getDocumentDOM().width= 400;
```

Véase también

`document.height`

document.xmlPanel()

Disponibilidad

Flash MX 2004.

Uso

```
document.xmlPanel( fileURI )
```

Parámetros

`fileURI` Una cadena, expresada como archivo:/// URI, que especifica la ruta del archivo XML que define los controles del panel. Se necesita la ruta completa.

Valor devuelto

Un objeto que tiene propiedades definidas para todos los controles definidos en el archivo XML. Todas las propiedades se devuelven como cadenas. El objeto devuelto tendrá una propiedad predefinida llamada `"dismiss"` que tendrá el valor de cadena `"accept"` o `"cancel"`.

Descripción

Método; envía un cuadro de diálogo XMLUI. Véase `fl.xmlui`.

Ejemplo

El ejemplo siguiente carga el archivo Test.xml y muestra cada una de las propiedades que contiene:

```
var obj = fl.getDocumentDOM().xmlPanel(fl.configURI + "Commands/Test.xml");
for (var prop in obj) {
  fl.trace("property " + prop + " = " + obj[prop]);
}
```

document.zoomFactor

Disponibilidad

Flash 8.

Uso

```
document.zoomFactor
```

Descripción

Propiedad; especifica el porcentaje de zoom del escenario en tiempo de edición. Un valor de 1 es igual a un zoom del 100%, 8 es igual a 800%, 0,5 es igual a 50%, y así sucesivamente.

Ejemplo

El ejemplo siguiente establece el factor de zoom del escenario en un 200%.

```
fl.getDocumentDOM().zoomFactor = 2;
```

Objeto drawingLayer

Disponibilidad

Flash MX 2004.

Descripción

Se puede acceder al objeto drawingLayer desde JavaScript como elemento secundario del objeto Flash. El objeto drawingLayer se utiliza en herramientas ampliables cuando el usuario desea dibujar temporalmente mientras arrastra el ratón, por ejemplo para crear un recuadro de delimitación). Deberá llamar a `drawingLayer.beginFrame()` antes de llamar a otros métodos de drawingLayer.

Resumen de métodos del objeto drawingLayer

Los métodos siguientes están disponibles para el objeto drawingLayer:

Métodos	Descripción
`drawingLayer.beginDraw()`	Sitúa a Flash en modo de dibujo.
`drawingLayer.beginFrame()`	Borra lo que se había dibujado anteriormente utilizando drawingLayer y prepara para más comandos de dibujo.
`drawingLayer.cubicCurveTo()`	Dibuja una curva cúbica desde la ubicación actual de la pluma empleando los parámetros como coordenadas del segmento cúbico.
`drawingLayer.curveTo()`	Dibuja un segmento de curva cuadrática comenzando en la posición de dibujo actual y terminando en un punto especificado.
`drawingLayer.drawPath()`	Dibuja la ruta especificada.
`drawingLayer.endDraw()`	Sale del modo de dibujo.
`drawingLayer.endFrame()`	Señala el final de un grupo de comandos de dibujo.
`drawingLayer.lineTo()`	Dibuja una línea desde la posición de dibujo actual hasta el punto (x,y).
`drawingLayer.moveTo()`	Establece la posición de dibujo actual.
`drawingLayer.newPath()`	Devuelve un nuevo Objeto Path.
`drawingLayer.setColor()`	Establece el color de los datos dibujados a continuación.

drawingLayer.beginDraw()

Disponibilidad

Flash MX 2004.

Uso

```
drawingLayer.beginDraw([persistentDraw])
```

Parámetros

persistentDraw Un valor booleano (opcional). Si se define como `true`, indica que el dibujo del último fotograma permanece en el escenario hasta que se realice una nueva llamada `beginDraw()` o `beginFrame()`. (En este contexto, *frame* hace referencia a dónde comienza y termina el dibujo, no a fotogramas de la línea de tiempo.) Por ejemplo, cuando los usuarios dibujan un rectángulo, pueden obtener una vista previa del contorno de la forma mientras arrastran el ratón. Si desea que la vista previa de la forma permanezca cuando el usuario libere el botón del ratón, defina *persistentDraw* como `true`.

Valor devuelto

Ninguno.

Descripción

Método; sitúa a Flash en modo de dibujo. El modo de dibujo se emplea para dibujar temporalmente mientras se presiona el botón del ratón. Este método sólo suele utilizarse cuando se crean herramientas ampliables.

Ejemplo

El ejemplo siguiente sitúa a Flash en modo de dibujo:

```
fl.drawingLayer.beginDraw();
```

drawingLayer.beginFrame()

Disponibilidad

Flash MX 2004.

Uso

```
drawingLayer.beginFrame()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; borra lo que se había dibujado anteriormente utilizando drawingLayer y prepara para más comandos de dibujo. Deberá llamarse después de `drawingLayer.beginDraw()`. Todo lo que haya dibujado entre `drawingLayer.beginFrame()` y `drawingLayer.endFrame()` permanecerá en el escenario hasta que llame a los siguientes `beginFrame()` y `endFrame()`. (En este contexto, *frame* hace referencia a dónde comienza y termina el dibujo, no a fotogramas de la línea de tiempo.) Este método sólo suele utilizarse cuando se crean herramientas ampliables. Véase `drawingLayer.beginDraw()`.

drawingLayer.cubicCurveTo()

Disponibilidad

Flash MX 2004.

Uso

```
drawingLayer.cubicCurveTo(x1Ctrl, y1Ctrl, x2Ctl, y2Ctl, xEnd, yEnd)
```

Parámetros

x1Ctl Un valor de coma flotante que es la ubicación *x* del primer punto de control.

y1Ctl Un valor de coma flotante que es la ubicación *y* del primer punto de control.

x2Ctl Un valor de coma flotante que es la posición *x* del punto de control medio.

y2Ctl Un valor de coma flotante que es la posición *y* del punto de control medio.

xEnd Un valor de coma flotante que es la posición *x* del punto de control final.

yEnd Un valor de coma flotante que es la posición *y* del punto de control final.

Valor devuelto

Ninguno.

Descripción

Método; dibuja una curva cúbica desde la ubicación actual de la pluma empleando los parámetros como coordenadas del segmento cúbico. Este método sólo suele utilizarse cuando se crean herramientas ampliables.

Ejemplo

El ejemplo siguiente dibuja una curva cúbica utilizando los puntos de control especificados:

```
fl.drawingLayer.cubicCurveTo(0, 0, 1, 1, 2, 0);
```

drawingLayer.curveTo()

Disponibilidad

Flash MX 2004.

Uso

```
drawingLayer.curveTo(xCtl, yCtl, xEnd, yEnd)
```

Parámetros

xCtl Un valor de coma flotante que es la posición *x* del punto de control.

yCtl Un valor de coma flotante que es la posición *y* del punto de control.

xEnd Un valor de coma flotante que es la posición *x* del punto de control final.

yEnd Un valor de coma flotante que es la posición *y* del punto de control final.

Valor devuelto

Ninguno.

Descripción

Método; dibuja un segmento de curva cuadrática comenzando en la posición de dibujo actual y terminando en un punto especificado. Este método sólo suele utilizarse cuando se crean herramientas ampliables.

Ejemplo

El ejemplo siguiente dibuja una curva cuadrática utilizando los puntos de control especificados:

```
fl.drawingLayer.curveTo(0, 0, 2, 0);
```

drawingLayer.drawPath()

Disponibilidad

Flash MX 2004.

Uso

```
drawingLayer.drawPath(path)
```

Parámetros

path Un Objeto Path para dibujar.

Valor devuelto

Ninguno.

Descripción

Método; dibuja la ruta especificada por el parámetro *path*. Este método sólo suele utilizarse cuando se crean herramientas ampliables.

Ejemplo

El ejemplo siguiente dibuja una ruta especificada por el objeto Path llamada `gamePath`:

```
fl.drawingLayer.drawPath(gamePath);
```

drawingLayer.endDraw()

Disponibilidad

Flash MX 2004.

Uso

```
drawingLayer.endDraw()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; sale del modo de dibujo. El modo de dibujo se utiliza cuando se desea dibujar temporalmente mientras se presiona el botón del ratón. Este método sólo suele utilizarse cuando se crean herramientas ampliables.

Ejemplo

El ejemplo siguiente sale del modo de dibujo:

```
fl.drawingLayer.endDraw();
```

drawingLayer.endFrame()

Disponibilidad

Flash MX 2004.

Uso

```
drawingLayer.endFrame()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; señala el final de un grupo de comandos de dibujo. Un grupo de comandos de dibujo hace referencia a todo lo que se dibuja entre `drawingLayer.beginFrame()` y `drawingLayer.endFrame()`. La siguiente llamada a `drawingLayer.beginFrame()` borrará todo lo que se haya dibujado en este grupo de comandos de dibujo. Este método sólo suele utilizarse cuando se crean herramientas ampliables.

drawingLayer.lineTo()

Disponibilidad

Flash MX 2004.

Uso

```
drawingLayer.lineTo(x, y)
```

Parámetros

x Un valor de coma flotante que es la coordenada *x* del punto final de la línea que se va a dibujar.

y Un valor de coma flotante que es la coordenada *y* del punto final de la línea que se va a dibujar.

Valor devuelto

Ninguno.

Descripción

Método; dibuja una línea desde la posición de dibujo actual hasta el punto (*x*,*y*). Este método sólo suele utilizarse cuando se crean herramientas ampliables.

Ejemplo

El ejemplo siguiente dibuja una línea desde la posición de dibujo actual hasta el punto (20,30):

```
fl.drawingLayer.lineTo(20, 30);
```

drawingLayer.moveTo()

Disponibilidad

Flash MX 2004.

Uso

```
drawingLayer.moveTo(x, y)
```

Parámetros

x Un valor de coma flotante que especifica la coordenada *x* de la posición en la que se comienza a dibujar.

y Un valor de coma flotante que especifica la coordenada *y* de la posición en la que se comienza a dibujar.

Valor devuelto

Ninguno.

Descripción

Método; establece la posición de dibujo actual. Este método sólo suele utilizarse cuando se crean herramientas ampliables.

Ejemplo

El ejemplo siguiente establece la posición de dibujo actual en el punto (10,15):

```
fl.drawingLayer.moveTo(10, 15);
```

drawingLayer.newPath()

Disponibilidad

Flash MX 2004.

Uso

```
drawingLayer.newPath()
```

Parámetros

Ninguno.

Valor devuelto

Un objeto Path.

Descripción

Método; devuelve un nuevo objeto Path. Este método sólo suele utilizarse cuando se crean herramientas ampliables. Véase Objeto Path.

Ejemplo

El ejemplo siguiente devuelve un objeto Path nuevo:

```
fl.drawingLayer.newPath();
```

drawingLayer.setColor()

Disponibilidad

Flash MX 2004.

Uso

```
drawingLayer.setColor(color)
```

Parámetros

color El color de los datos dibujados a continuación en uno de los formatos siguientes:

- Una cadena con el formato `"#RRGGBB"` o `"#RRGGBBAA"`
- Un número hexadecimal con el formato `0xRRGGBB`
- Un entero que representa el equivalente decimal del número hexadecimal

Valor devuelto

Ninguno.

Descripción

Método; establece el color de los datos dibujados a continuación. Sólo se aplica a datos persistentes. Para utilizar este método, el parámetro transferido a `drawingLayer.beginDraw()` debe definirse como `true`. Este método sólo suele utilizarse cuando se crean herramientas ampliables. Véase `drawingLayer.beginDraw()`.

Ejemplo

El ejemplo siguiente dibuja una línea roja en el escenario:

```
fl.drawingLayer.beginDraw( true );
fl.drawingLayer.beginFrame();
fl.drawingLayer.setColor( "#ff0000" );
fl.drawingLayer.moveTo(0,0);
fl.drawingLayer.lineTo(100,100);
fl.drawingLayer.endFrame();
fl.drawingLayer.endDraw();
```

Objeto Edge

Disponibilidad

Flash MX 2004.

Descripción

El objeto Edge representa un borde de una forma en el escenario.

Resumen de métodos del objeto Edge

Los métodos siguientes están disponibles para el objeto Edge:

Método	Descripción
`edge.getControl()`	Obtiene un objeto point definido en la ubicación del punto de control especificado del borde.
`edge.getHalfEdge()`	Devuelve un Objeto HalfEdge.
`edge.setControl()`	Establece la posición del punto de control del borde.
`edge.splitEdge()`	Divide el borde en dos partes.

Resumen de propiedades del objeto Edge

Las propiedades siguientes están disponibles para el objeto Edge:

Propiedad	Descripción
`edge.id`	De sólo lectura; un entero que representa un identificador único para el borde.
`edge.isLine`	De sólo lectura; un entero con un valor de 0 o 1.

edge.getControl()

Disponibilidad

Flash MX 2004.

Uso

```
edge.getControl(i)
```

Parámetros

i Un entero que especifica qué punto de control del borde se va a devolver. Especifique 0 para el primer punto de control, 1 para el punto de control medio o 2 para el punto de control final. Si la propiedad `edge.isLine` es `true`, el punto de control medio se definirá como el punto medio del segmento que unce los puntos de control inicial y final.

Valor devuelto

El punto de control especificado.

Descripción

Método; obtiene un objeto point definido en la ubicación del punto de control especificado del borde.

Ejemplo

El ejemplo siguiente almacena el primer punto de control de la forma especificada en la variable `pt`:

```
var shape = fl.getDocumentDOM().selection[0];
var pt = shape.edges[0].getControl(0);
```

edge.getHalfEdge()

Disponibilidad

Flash MX 2004.

Uso

`edge.getHalfEdge(`*`index`*`)`

Parámetros

index Un entero que especifica qué lado dirigido se va a devolver. El valor de *index* debe ser 0 para el primer lado dirigido o 1 para el segundo.

Valor devuelto

Un objeto HalfEdge.

Descripción

Método; devuelve un Objeto HalfEdge.

Ejemplo

El ejemplo siguiente almacena los lados dirigidos del borde especificado en las variables `hEdge0` y `hEdge1`:

```
var shape = fl.getDocumentDOM().selection[0];
var edge = shape.edges[0];
var hEdge0 = edge.getHalfEdge(0);
var hEdge1 = edge.getHalfEdge(1);
```

edge.id

Disponibilidad

Flash MX 2004.

Uso

```
edge.id
```

Descripción

Propiedad de sólo lectura; un entero que representa un identificador único para el borde.

Ejemplo

El ejemplo siguiente almacena un identificador único para el borde especificado en la variable `my_shape_id`:

```
var shape = fl.getDocumentDOM().selection[0];
var my_shape_id = shape.edges[0].id;
```

edge.isLine

Disponibilidad

Flash MX 2004.

Uso

```
edge.isLine
```

Descripción

Propiedad de sólo lectura; un entero con un valor de 0 o 1. Un valor de 1 indica que el borde es una línea recta. En ese caso, el punto de control medio divide en dos partes iguales la línea que une los dos puntos finales.

Ejemplo

El ejemplo siguiente determina si el borde especificado es una línea recta y muestra un valor de 1 (es una línea recta) o 0 (no es una línea recta) en el panel Salida:

```
var shape = fl.getDocumentDOM().selection[0];
fl.trace(shape.edges[0].isLine);
```

edge.setControl()

Disponibilidad

Flash MX 2004.

Uso

```
edge.setControl( index, x, y )
```

Parámetros

index Un entero que especifica qué punto de control se va a definir. Utilice los valores 0, 1 o 2 para especificar los puntos de control inicial, medio y final respectivamente.

x Un valor de coma flotante que especifica la ubicación horizontal del punto de control. Si el escenario se encuentra en modo de edición o de edición en contexto, la coordenada del punto será relativa al objeto editado. En caso contrario, la coordenada del punto será relativa al escenario.

y Un valor de coma flotante que especifica la ubicación vertical del punto de control. Si el escenario se encuentra en modo de edición o de edición en contexto, la coordenada del punto será relativa al objeto editado. En caso contrario, la coordenada del punto será relativa al escenario.

Valor devuelto

Ninguno.

Descripción

Método; establece la posición del punto de control del borde. Deberá llamar a `shape.beginEdit()` antes de utilizar este método. Véase `shape.beginEdit()`.

Ejemplo

El ejemplo siguiente define el punto de control inicial del borde especificado en las coordenadas (0, 1):

```
x = 0; y = 1;
var shape = fl.getDocumentDOM().selection[0];
shape.beginEdit();
shape.edges[0].setControl(0, x, y);
shape.endEdit();
```

edge.splitEdge()

Disponibilidad

Flash MX 2004.

Uso

```
edge.splitEdge( t )
```

Parámetros

t Un valor de coma flotante entre 0 y 1 que especifica dónde se divide el borde. Un valor de 0 representa un punto final y 1, el otro. Por ejemplo, si se transfiere un valor de 0,5 se dividirá el borde por la mitad, lo que para una línea es exactamente el centro. Si el borde representa una curva, 0,5 representa el medio paramétrico de la curva.

Valor devuelto

Ninguno.

Descripción

Método; divide el borde en dos partes. Deberá llamar a `shape.beginEdit()` antes de utilizar este método.

Ejemplo

El ejemplo siguiente divide el borde especificado por la mitad:

```
var shape = fl.getDocumentDOM().selection[0];
shape.beginEdit()
shape.edges[0].splitEdge( 0.5 );
shape.endEdit()
```

Objeto Effect

Disponibilidad

Flash MX 2004.

Descripción

Este es un objeto descriptor de un solo efecto. Las propiedades `fl.activeEffect` y `fl.effects` contienen este tipo de objeto. El objeto Effect representa una instancia de un efecto de línea de tiempo. Véase `fl.activeEffect` y `fl.effects`.

Resumen de propiedades del objeto Effect

Además de las propiedades de la tabla siguiente, los objetos Effect también pueden tener parámetros definidos por el usuario que deben especificarse en el mismo archivo XML que especifica las propiedades `effect.effectName` y `effect.sourceFile`. Estos parámetros especifican qué elementos de la interfaz de usuario deben crearse (como campos de edición, casillas de verificación y cuadros de lista), lo que depende del tipo de efecto que esté creando. Puede especificar etiquetas que aparecerán con el control además de valores predeterminados.

Propiedad	Descripción
`effect.effectName`	De sólo lectura; una cadena que aparece en el menú Contexto para los efectos.
`effect.groupName`	De sólo lectura; una cadena que representa el nombre del grupo de efectos utilizado para el menú jerárquico Contexto para los efectos.
`effect.sourceFile`	De sólo lectura; una cadena que especifica el nombre del archivo de origen JSFL para el efecto especificado.
`effect.symbolType`	De sólo lectura; una cadena que especifica el tipo de símbolo que se va a crear durante la aplicación inicial del efecto.
`effect.useXMLToUI`	Un valor booleano que permite anular el comportamiento predeterminado del uso de XMLUI para crear un cuadro de diálogo compuesto por uno o más controles.

effect.effectName

Disponibilidad

Flash MX 2004.

Uso

```
effect.effectName
```

Descripción

Propiedad de sólo lectura; una cadena que aparece en el menú Contexto para los efectos. Cada efecto debe tener un nombre único.

Ejemplo

El ejemplo siguiente almacena el nombre del efecto actual en la variable `efName`:

```
var efName = fl.activeEffect.effectName;
```

effect.groupName

Disponibilidad

Flash MX 2004.

Uso

```
effect.groupName
```

Descripción

Propiedad de sólo lectura; una cadena que representa el nombre del grupo de efectos utilizado para el menú jerárquico Contexto de los efectos. Si este valor es una cadena vacía, el efecto aparece desagrupado en el nivel superior del menú Contexto. El nombre del grupo y el nombre del efecto se especifican en el archivo XML correspondiente al efecto.

Ejemplo

El ejemplo siguiente almacena el nombre del grupo del efecto actual en la variable `efGroupName`:

```
var efGroupName = fl.activeEffect.groupName;
```

effect.sourceFile

Disponibilidad

Flash MX 2004.

Uso

```
effect.sourceFile
```

Descripción

Propiedad de sólo lectura; una cadena que especifica el nombre del archivo de origen JSFL para el efecto especificado. Esta cadena se utiliza para vincular un parámetro XML con su implementación de efecto JSFL. Debe incluir este parámetro XML en el archivo XML correspondiente al efecto.

Ejemplo

El ejemplo siguiente almacena el nombre del archivo de origen del efecto JSFL en la variable `efSourceFile`:

```
var efSourceFile = fl.activeEffect.sourceFile;
```

effect.symbolType

Disponibilidad

Flash MX 2004.

Uso

```
effect.symbolType
```

Descripción

Propiedad de sólo lectura; una cadena que especifica el tipo de símbolo que se va a crear durante la aplicación inicial del efecto. Los tipos válidos son: `"graphic"`, `"movie clip"` y `"button"`. Si no se especificó un tipo de símbolo cuando se creó el efecto, el valor predeterminado será `"graphic"`.

Ejemplo

El ejemplo siguiente almacena el tipo de símbolo para el efecto actual en la variable `efType`:

```
var efType = fl.activeEffect.symbolType;
```

effect.useXMLToUI

Disponibilidad

Flash MX 2004.

Uso

```
effect.useXMLToUI
```

Descripción

Propiedad; un valor booleano que permite anular el comportamiento predeterminado del uso de XMLUI para crear un cuadro de diálogo compuesto por uno o más controles. El valor predeterminado es `true`. Si se define como `false`, no se publicará el cuadro de diálogo estándar de XMLUI y usted no será responsable de publicar una interfaz.

Ejemplo

El ejemplo siguiente especifica que el efecto envía su propia interfaz:

```
function configureEffect() {
  fl.activeEffect.useXMLToUI = false;
}
```

Objeto Element

Disponibilidad

Flash MX 2004.

Descripción

Todo lo que aparece en el escenario es de tipo Element. El ejemplo del código siguiente permite seleccionar un elemento:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0];
```

Resumen de métodos del objeto Element

Los métodos siguientes están disponibles para el objeto Element:

Método	Descripción
`element.getPersistentData()`	Recupera el valor de los datos especificados por el parámetro *name*.
`element.hasPersistentData()`	Determina si los datos especificados se han asociado al elemento especificado.
`element.removePersistentData()`	Elimina datos persistentes con el nombre especificado que se han asociado al objeto.
`element.setPersistentData()`	Almacena datos con un elemento.

Resumen de propiedades del objeto Element

Las propiedades siguientes están disponibles para el objeto Element:

Propiedad	Descripción
`element.depth`	De sólo lectura; un entero que tiene un valor mayor que 0 para la profundidad del objeto en la vista.
`element.elementType`	De sólo lectura; una cadena que representa el tipo de elemento especificado.
`element.height`	Un valor flotante que especifica el alto del elemento en píxeles.
`element.layer`	Sólo lectura; representa el Objeto Layer en el que se encuentra el elemento.
`element.left`	De sólo lectura; un valor flotante que representa el lado izquierdo del elemento.

Propiedad	Descripción
element.locked	Un valor booleano: true si el elemento está bloqueado; false en caso contrario.
element.matrix	Un Objeto Matrix. La matriz tiene propiedades a, b, c, d, tx y ty. a, b, c, d son valores de coma flotante; tx y ty son coordenadas.
element.name	Una cadena que especifica el nombre del elemento, que suele denominarse nombre de instancia.
element.selected	Un valor booleano que especifica si el elemento se selecciona o no.
element.top	De sólo lectura; parte superior del elemento.
element.width	Un valor flotante que especifica el ancho del elemento en píxeles.

element.depth

Disponibilidad

Flash MX 2004.

Uso

```
element.depth
```

Descripción

Propiedad de sólo lectura; un entero que tiene un valor mayor que 0 para la profundidad del objeto en la vista. El orden de dibujo de los objetos del escenario especifica cuál está sobre los demás. El orden de los objetos también se puede gestionar con el elemento de menú Modificar > Organizar.

Ejemplo

El ejemplo siguiente muestra la profundidad del elemento especificado en el panel Salida:

```
// Seleccione un objeto y ejecute este script.
fl.trace("Depth of selected object: " +
  fl.getDocumentDOM().selection[0].depth);
```

Consulte el ejemplo de element.elementType.

element.elementType

Disponibilidad

Flash MX 2004.

Uso

```
element.elementType
```

Descripción

Propiedad de sólo lectura; una cadena que representa el tipo de elemento especificado. El valor es uno de los siguientes: `"shape"`, `"text"`, `"instance"` o `"shapeObj"`. Se crea un `"shapeObj"` con una herramienta ampliable.

Ejemplo

El ejemplo siguiente almacena el tipo del primer elemento en la variable `eType`:

```
// En un nuevo archivo, sitúa un clip de película en la capa superior del
  primer fotograma y,
// a continuación, ejecuta esta línea de script.
var eType =
  fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].elemen
  tType; //eType = instancia
```

El ejemplo siguiente muestra varias propiedades para todos los elementos de la capa o el fotograma actual:

```
var tl = fl.getDocumentDOM().getTimeline()
var elts = tl.layers[tl.currentLayer].frames[tl.currentFrame].elements;
for (var x = 0; x < elts.length; x++) {
  var elt = elts[x];
  fl.trace("Element "+ x +" Name = " + elt.name + " Type = " +
  elt.elementType + " location = " + elt.left + "," + elt.top + " Depth = "
  + elt.depth);
}
```

element.getPersistentData()

Disponibilidad

Flash MX 2004.

Uso

```
element.getPersistentData( name )
```

Parámetros

name Una cadena que identifica los datos que se van a devolver.

Valor devuelto

Los datos especificados por el parámetro *name* o 0 si no existen los datos.

Descripción

Método; recupera el valor de los datos especificados por el parámetro *name*. El tipo de datos depende del tipo de datos que se han almacenado (véase `element.setPersistentData()`). Sólo los símbolos y mapas de bits admiten datos persistentes.

Ejemplo

El ejemplo siguiente define y obtiene datos para el elemento especificado, muestra su valor en el panel Salida y, a continuación, elimina los datos:

```
// Selecciona un símbolo o mapa de bits como mínimo en la primera capa del
  primer fotograma.
var elt =
  fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0];
elt.setPersistentData("myData","integer", 12);
if (elt.hasPersistentData("myData")){
   fl.trace("myData = "+ elt.getPersistentData("myData"));
   elt.removePersistentData( "myData" );
   fl.trace("myData = "+ elt.getPersistentData("myData"));
}
```

element.hasPersistentData()

Disponibilidad

Flash MX 2004.

Uso

```
element.hasPersistentData( name )
```

Parámetros

name Una cadena que especifica el nombre del elemento de datos que se va a comprobar.

Valor devuelto

Un valor booleano: `true` si los datos especificados están asociados al objeto; `false` en caso contrario.

Descripción

Método; determina si los datos especificados se han asociado al elemento especificado. Sólo los símbolos y mapas de bits admiten datos persistentes.

Ejemplo

Véase `element.getPersistentData()`.

element.height

Disponibilidad

Flash MX 2004.

Uso

```
element.height
```

Descripción

Propiedad; un valor flotante que especifica el alto del elemento en píxeles.

> **NOTA** No utilice esta propiedad para cambiar el tamaño de un campo de texto. Seleccione el campo de texto y utilice `document.setTextRectangle()`. El uso de esta propiedad con un campo de texto cambia la escala del texto.

Ejemplo

El ejemplo siguiente establece la altura del elemento especificado en 100:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].height =
  100;
```

element.layer

Disponibilidad

Flash 8.

Uso

```
element.layer
```

Descripción

Propiedad de sólo lectura; representa el Objeto Layer en el que se encuentra el elemento.

Ejemplo

El ejemplo siguiente almacena el objeto Layer que contiene el elemento de la variable `theLayer`:

```
var theLayer = element.layer;
```

element.left

Disponibilidad

Flash MX 2004.

Uso

```
element.left
```

Descripción

Propiedad de sólo lectura; un valor flotante que representa el lado izquierdo del elemento. El valor de `element.left` es relativo a la esquina superior izquierda del escenario para elementos que están en una escena, y es relativo al punto de registro del símbolo si el elemento se almacena con un símbolo. Utilice `document.setSelectionBounds()` o `document.moveSelectionBy()` para definir esta propiedad.

Ejemplo

El ejemplo siguiente ilustra cómo cambia el valor de esta propiedad cuando se mueve un elemento:

```
// Selecciona un elemento del escenario y, a continuación, ejecuta este
  script.
var sel = fl.getDocumentDOM().selection[0];
fl.trace("Left (before) = " + sel.left);
fl.getDocumentDOM().moveSelectionBy({x:100, y:0});
fl.trace("Left (after) = " + sel.left);
```

Consulte el ejemplo de `element.elementType`.

element.locked

Disponibilidad

Flash MX 2004.

Uso

```
element.locked
```

Descripción

Propiedad; un valor booleano: `true` si el elemento está bloqueado; `false` en caso contrario. Si el valor de `element.elementType` es `"shape"`, se ignorará esta propiedad.

Ejemplo

El ejemplo siguiente bloquea el primer elemento del primer fotograma de la capa superior:

```
// Es similar a Modificar > Organizar > Bloquear:
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].locked =
  true;
```

element.matrix

Disponibilidad

Flash MX 2004.

Uso

```
element.matrix
```

Descripción

Propiedad; un objeto Matrix. Una matriz tiene propiedades `a`, `b`, `c`, `d`, `tx` y `ty`. Las propiedades `a`, `b`, `c` y `d` son valores de coma flotante; las propiedades `tx` y `ty` son coordenadas. Véase Objeto Matrix.

Ejemplo

El ejemplo siguiente mueve el elemento especificado 10 píxeles en *x* y 20 píxeles en *y*:

```
var mat =
  fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].matrix
  ;
mat.tx += 10;
mat.ty += 20;
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].matrix =
  mat;
```

element.name

Disponibilidad

Flash MX 2004.

Uso

```
element.name
```

Descripción

Propiedad; una cadena que especifica el nombre del elemento, que suele denominarse nombre de instancia. Si el valor de `element.elementType` es `"shape"`, se ignorará esta propiedad. Véase `element.elementType`.

Ejemplo

El ejemplo siguiente define el nombre de instancia del primer elemento del Fotograma 1 y la capa superior como `"clip_mc"`:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].name =
  "clip_mc";
```

Consulte el ejemplo de `element.elementType`.

element.removePersistentData()

Disponibilidad

Flash MX 2004.

Uso

```
element.removePersistentData( name )
```

Parámetros

name Una cadena que especifica el nombre de los datos que se van a eliminar.

Valor devuelto

Ninguno.

Descripción

Método; elimina datos persistentes con el nombre especificado que se han asociado al objeto. Sólo los símbolos y mapas de bits admiten datos persistentes.

Ejemplo

Véase `element.getPersistentData()`.

element.selected

Disponibilidad

Flash 8.

Uso

```
element.selected
```

Descripción

Propiedad; valor booleano que especifica si el elemento está seleccionado (`true`) o no (`false`).

Ejemplo

El ejemplo siguiente selecciona el elemento:

```
element.selected = true;
```

element.setPersistentData()

Disponibilidad

Flash MX 2004.

Uso

```
element.setPersistentData( name, type, value )
```

Parámetros

name Una cadena que especifica el nombre que se va a asociar a los datos. Este nombre se utiliza para recuperar los datos.

type Una cadena que define el tipo de datos. Los valores válidos son: `"integer"`, `"integerArray"`, `"double"`, `"doubleArray"`, `"string"` y `"byteArray"`.

value Especifica el valor que se va a asociar al objeto. El tipo de datos de *value* depende del valor del parámetro *type*. El valor especificado deberá ser adecuado al tipo de datos especificado por el parámetro *type*.

Valor devuelto

Ninguno.

Descripción

Método; almacena datos con un elemento. Los datos estarán disponibles cuando se abra el archivo FLA que contiene el elemento. Sólo los símbolos y mapas de bits admiten datos persistentes.

Ejemplo

Véase `element.getPersistentData()`.

element.top

Disponibilidad

Flash MX 2004.

Uso

```
element.top
```

Descripción

Propiedad de sólo lectura; parte superior del elemento. El valor de `element.top` es relativo a la esquina superior izquierda del escenario para elementos que están en una escena, y es relativo al punto de registro del símbolo si el elemento se almacena con un símbolo. Utilice `document.setSelectionBounds()` o `document.moveSelectionBy()` para definir esta propiedad.

Ejemplo

El ejemplo siguiente muestra cómo cambia el valor de esta propiedad cuando se mueve un elemento:

```
// Selecciona un elemento del escenario y, a continuación, ejecuta este
  script.
var sel = fl.getDocumentDOM().selection[0];
fl.trace("Top (before) = " + sel.top);
fl.getDocumentDOM().moveSelectionBy({x:0, y:100});
fl.trace("Top (after) = " + sel.top);
```

Consulte el ejemplo de `element.elementType`.

element.width

Disponibilidad

Flash MX 2004.

Uso

```
element.width
```

Descripción

Propiedad; un valor flotante que especifica el ancho del elemento en píxeles.

NOTA	No utilice esta propiedad para cambiar el tamaño de un campo de texto. Seleccione el campo de texto y utilice `document.setTextRectangle()`. El uso de esta propiedad con un campo de texto cambia la escala del texto.

Ejemplo

El ejemplo siguiente establece el ancho del elemento especificado en 100:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].width=
  100;
```

Objeto Fill

Disponibilidad

Flash MX 2004.

Descripción

Este objeto contiene todas las propiedades de la configuración de Color de relleno del panel Herramientas o de una forma seleccionada. Para recuperar el objeto Fill, utilice `document.getCustomFill()`.

Resumen de propiedades del objeto Fill

Las propiedades siguientes están disponibles para el objeto Fill:

Propiedad	Descripción
`fill.color`	Una cadena, valor hexadecimal o entero que representa el color de relleno.
`fill.colorArray`	Una matriz de colores en degradado.
`fill.focalPoint`	Un entero que especifica el desplazamiento horizontal del punto focal del degradado desde el punto de transformación.
`fill.linearRGB`	Un valor booleano que especifica si se debe representar el relleno como un degradado RGB radial o lineal.
`fill.matrix`	Un Objeto Matrix que define la ubicación, la orientación y las escalas de los rellenos con degradado.
`fill.overflow`	Una cadena que especifica el comportamiento del desbordamiento de un degradado.
`fill.posArray`	Una matriz de enteros, cada uno en el rango 0 ... 255, que indica la posición del color correspondiente.
`fill.style`	Una cadena que especifica el estilo de relleno.

fill.color

Disponibilidad

Flash MX 2004.

Uso

```
fill.color
```

Descripción

Propiedad; el color del relleno, en uno de los formatos siguientes:

- Una cadena con el formato `"#RRGGBB"` o `"#RRGGBBAA"`
- Un número hexadecimal con el formato `0xRRGGBB`
- Un entero que representa el equivalente decimal del número hexadecimal

Ejemplo

El ejemplo siguiente define el color de relleno de la selección actual:

```
var fill = fl.getDocumentDOM().getCustomFill();
fill.color = '#FFFFFF';
fl.getDocumentDOM().setCustomFill( fill );
```

fill.colorArray

Disponibilidad

Flash MX 2004.

Uso

```
fill.colorArray
```

Descripción

Propiedad; una matriz de colores en degradado, expresado en enteros. Esta propiedad sólo está disponible si el valor de la propiedad `fill.style` es `"radialGradient"` o `"linearGradient"`. Véase `fill.style`.

Ejemplo

El ejemplo siguiente muestra la matriz de colores de la selección actual, si es necesario, en el panel Salida:

```
var fill = fl.getDocumentDOM().getCustomFill();
if(fill.style == "linearGradient" || fill.style == "radialGradient")
  alert(fill.colorArray);
```

fill.focalPoint

Disponibilidad

Flash 8.

Uso

```
fill.focalPoint
```

Descripción

Propiedad; un entero que especifica el desplazamiento horizontal del punto focal del degradado desde el punto de transformación. Un valor de 10, por ejemplo, situaría el punto focal a 10/255 de la distancia desde el punto de transformación hasta el borde del degradado. Un valor de -255 situaría el punto focal en el límite izquierdo del degradado. El valor predeterminado es 0.

Esta propiedad sólo está disponible si el valor de la propiedad `fill.style` es `"radialGradient"`.

Ejemplo

El ejemplo siguiente define el punto focal de un degradado radial en 10 píxeles hacia la derecha del centro de la forma.

```
var fill = fl.getDocumentDOM().getCustomFill();
fill.focalPoint = 10;
fl.getDocumentDOM().setCustomFill(fill);
```

fill.linearRGB

Disponibilidad

Flash 8.

Uso

```
fill.linearRGB
```

Descripción

Propiedad; un valor booleano que especifica si se debe representar el relleno como un degradado RGB radial o lineal. Defina esta propiedad como `true` para especificar una interpolación lineal de un degradado, o como `false` para especificar una interpolación radial de un degradado. El valor predeterminado es `false`.

Ejemplo

El ejemplo siguiente especifica que el degradado se debe representar con un valor RGB lineal.

```
var fill = fl.getDocumentDOM().getCustomFill();
fill.linearRGB = true;
fl.getDocumentDOM().setCustomFill(fill);
```

fill.matrix

Disponibilidad

Flash MX 2004.

Uso

```
fill.matrix
```

Descripción

Propiedad, un Objeto Matrix que define la ubicación, la orientación y las escalas de los rellenos con degradado.

fill.overflow

Disponibilidad

Flash 8.

Uso

```
fill.overflow
```

Descripción

Propiedad; una cadena que especifica el comportamiento del desbordamiento de un degradado. Los valores aceptables son: `"extend"`, `"repeat"` y `"reflect"`; en las cadenas no se distingue el uso de mayúsculas y minúsculas. El valor predeterminado es `"extend"`.

Ejemplo

El ejemplo siguiente especifica que el comportamiento del desbordamiento debe ser `"extend"`.

```
var fill = fl.getDocumentDOM().getCustomFill();
fill.overflow = "extend";
fl.getDocumentDOM().setCustomFill(fill);
```

fill.posArray

Disponibilidad

Flash MX 2004.

Uso

```
fill.posArray
```

Descripción

Propiedad; una matriz de enteros, cada uno en el rango 0 ... 255, que indica la posición del color correspondiente. Esta propiedad sólo está disponible si el valor de la propiedad `fill.style` es `"radialGradient"` o `"linearGradient"`.

Ejemplo

El ejemplo siguiente especifica los colores que se van a utilizar en un degradado lineal para la selección actual:

```
var fill = fl.getDocumentDOM().getCustomFill();
fill.style = "linearGradient";
fill.colorArray = [ 0x00ff00, 0xff0000, 0x0000ff ];
fill.posArray  = [0,  100, 200];
fl.getDocumentDOM().setCustomFill( fill );
```

fill.style

Disponibilidad

Flash MX 2004.

Uso

```
fill.style
```

Descripción

Propiedad; una cadena que especifica el estilo de relleno. Los valores aceptables son: `"solid"`, `"linearGradient"`, `"radialGradient"` y `"noFill"`. Si un objeto no tiene relleno, esta propiedad tiene un valor de `"noFill"`.

Si este valor es `"linearGradient"` o `"radialGradient"`, también estarán disponibles las propiedades `fill.colorArray` y `fill.posArray`.

Ejemplo

El ejemplo siguiente especifica los colores que se van a utilizar en un degradado lineal para la selección actual:

```
var fill = fl.getDocumentDOM().getCustomFill();
fill.style= "linearGradient";
fill.colorArray = [ 0x00ff00, 0xff0000, 0x0000ff ];
fill.posArray  = [0,  100, 200];
fl.getDocumentDOM().setCustomFill( fill );
```

Objeto Filter

Disponibilidad

Flash 8.

Descripción

Este objeto contiene todas las propiedades para todos los filtros. La propiedad `filter.name` especifica el tipo de filtro y determina qué propiedades se pueden aplicar a cada filtro. Véase `filter.name`.

Para devolver la lista de filtros de un objeto u objetos, utilice `document.getFilters()`. Para aplicar los filtros a un objeto u objetos, utilice `document.setFilters()`. Véase `document.getFilters()` y `document.setFilters()`.

Resumen de propiedades del objeto Filter

Pueden emplearse las propiedades siguientes con el objeto Filter.

Propiedad	Descripción
`filter.angle`	Un valor flotante que especifica el ángulo del color de la sombra o de resaltado, en grados.
`filter.blurX`	Un valor flotante que especifica la cantidad de desenfoque en la dirección *x*, en píxeles.
`filter.blurY`	Un valor flotante que especifica la cantidad de desenfoque en la dirección *y*.
`filter.brightness`	Un valor flotante que especifica el brillo del filtro.
`filter.color`	Una cadena, valor hexadecimal o entero que representa el color del filtro.
`filter.contrast`	Un valor flotante que especifica el valor de contraste del filtro.
`filter.distance`	Un valor flotante que especifica la distancia entre el efecto del filtro y un objeto, en píxeles.
`filter.hideObject`	Valor booleano que especifica si la imagen de origen está oculta (`true`) o se muestra (`false`).
`filter.highlightColor`	Una cadena, valor hexadecimal o entero que representa el color de resaltado.
`filter.hue`	Un valor flotante que especifica el matiz del filtro.
`filter.inner`	Valor booleano que especifica si la sombra es interior (`true`) o no (`false`).

Propiedad	Descripción
`filter.knockout`	Valor booleano que especifica si el filtro es extractor (`true`) o no (`false`).
`filter.name`	Una cadena que especifica el tipo de filtro (propiedad de sólo lectura).
`filter.quality`	Una cadena que especifica la calidad del desenfoque.
`filter.saturation`	Un valor flotante que especifica el valor de saturación del filtro.
`filter.shadowColor`	Una cadena, valor hexadecimal o entero que representa el color de sombra.
`filter.strength`	Un entero que especifica el porcentaje de intensidad del filtro.
`filter.type`	Una cadena que especifica el tipo de bisel o iluminado.

filter.angle

Disponibilidad

Flash 8.

Uso

```
filter.angle
```

Descripción

Propiedad; un valor flotante que especifica el ángulo del color de la sombra o de resaltado, en grados. Los valores aceptables están entre 0 y 360. Esta propiedad se define para los objetos Filter con un valor de `"bevelFilter"`, `"dropShadowFilter"`, `"gradientBevelFilter"` o `"gradientGlowFilter"` para la propiedad `filter.name`.

Ejemplo

El ejemplo siguiente establece el ángulo en 120 para los filtros de bisel en el objeto u objetos seleccionados:

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++) {
  if(myFilters[i].name == 'bevelFilter'){
    myFilters[i].angle = 120;
  }
}
fl.getDocumentDOM().setFilters(myFilters);
```

Véase también

`document.setFilterProperty()`

filter.blurX

Disponibilidad

Flash 8.

Uso

```
filter.blurX
```

Descripción

Propiedad; un valor flotante que especifica la cantidad de desenfoque en la dirección *x*, en píxeles. Los valores aceptables están entre 0 y 255. Esta propiedad se define para los objetos Filter con un valor de `"bevelFilter"`, `"blurFilter"`, `"dropShadowFilter"`, `"glowFilter"`, `"gradientBevelFilter"` o `"gradientGlowFilter"` para la propiedad `filter.name`.

Ejemplo

El ejemplo siguiente establece el valor `blurX` en 30 y el valor `blurY` en 20 para los filtros de desenfoque en el objeto u objetos seleccionados:

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
  if(myFilters[i].name == 'blurFilter'){
    myFilters[i].blurX = 30;
    myFilters[i].blurY = 20;
  }
}
fl.getDocumentDOM().setFilters(myFilters);
```

Véase también

`document.setFilterProperty()`, `filter.blurY`

filter.blurY

Disponibilidad

Flash 8.

Uso

```
filter.blurY
```

Descripción

Propiedad; un valor flotante que especifica la cantidad de desenfoque en la dirección *y*, en píxeles. Los valores aceptables están entre 0 y 255. Esta propiedad se define para los objetos Filter con un valor de `"bevelFilter"`, `"blurFilter"`, `"dropShadowFilter"`, `"glowFilter"`, `"gradientBevelFilter"` o `"gradientGlowFilter"` para la propiedad `filter.name`.

Ejemplo

Véase `filter.blurX`.

Véase también

`document.setFilterProperty()`, `filter.blurX`

filter.brightness

Disponibilidad

Flash 8.

Uso

```
filter.brightness
```

Descripción

Propiedad; un valor flotante que especifica el brillo del filtro. Los valores aceptables están entre -100 y 100. Esta propiedad se define para los objetos Filter con un valor de `"adjustColorFilter"` para la propiedad `filter.name`.

Ejemplo

El ejemplo siguiente establece el brillo en 30,5 para los filtros de ajustar color en el objeto u objetos seleccionados:

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
  if(myFilters[i].name == 'adjustColorFilter'){
    myFilters[i].brightness = 30.5;
  }
}
fl.getDocumentDOM().setFilters(myFilters);
```

filter.color

Disponibilidad

Flash 8.

Uso

```
filter.color
```

Descripción

Propiedad; el color del filtro, en uno de los formatos siguientes:

- Una cadena con el formato `"#RRGGBB"` o `"#RRGGBBAA"`
- Un número hexadecimal con el formato `0xRRGGBB`
- Un entero que representa el equivalente decimal del número hexadecimal

Esta propiedad se define para los objetos Filter con un valor de `"dropShadowFilter"` o `"glowFilter"` para la propiedad `filter.name`.

Ejemplo

El ejemplo siguiente establece el color en `"#ff00003e"` para los filtros de sombra en el objeto u objetos seleccionados:

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
  if(myFilters[i].name == 'dropShadowFilter'){
    myFilters[i].color = '#ff00003e';
  }
}
fl.getDocumentDOM().setFilters(myFilters);
```

Véase también

`document.setFilterProperty()`

filter.contrast

Disponibilidad

Flash 8.

Uso

```
filter.contrast
```

Descripción

Propiedad; un valor flotante que especifica el valor de contraste del filtro. Los valores aceptables están entre -100 y 100. Esta propiedad se define para los objetos Filter con un valor de `"adjustColorFilter"` para la propiedad `filter.name`.

Ejemplo

El ejemplo siguiente establece el valor de contraste en -15,5 para los filtros de ajustar color en el objeto u objetos seleccionados:

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
  if(myFilters[i].name == 'adjustColorFilter'){
    myFilters[i].contrast = -15.5;
  }
}
fl.getDocumentDOM().setFilters(myFilters);
```

filter.distance

Disponibilidad

Flash 8.

Uso

```
filter.distance
```

Descripción

Propiedad; un valor flotante que especifica la distancia entre el efecto del filtro y un objeto, en píxeles. Los valores aceptables están entre-255 y 255. Esta propiedad se define para los objetos Filter con un valor de `"bevelFilter"`, `"dropShadowFilter"`, `"gradientBevelFilter"` o `"gradientGlowFilter"` para la propiedad `filter.name`.

Ejemplo

El ejemplo siguiente establece la distancia en 10 píxeles para los filtros de sombra en el objeto u objetos seleccionados:

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
  if(myFilters[i].name == 'dropShadowFilter'){
    myFilters[i].distance = 10;
  }
}
fl.getDocumentDOM().setFilters(myFilters);
```

Véase también

`document.setFilterProperty()`

filter.hideObject

Disponibilidad

Flash 8.

Uso

```
filter.hideObject
```

Descripción

Propiedad; valor booleano que especifica si la imagen de origen está oculta (`true`) o se muestra (`false`). Esta propiedad se define para los objetos Filter con un valor de `"dropShadowFilter"` para la propiedad `filter.name`.

Ejemplo

El ejemplo siguiente establece el valor de `hideObject` como `true` para los filtros de sombra en el objeto u objetos seleccionados:

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
  if(myFilters[i].name == 'dropShadowFilter'){
    myFilters[i].hideObject = true;
  }
}
fl.getDocumentDOM().setFilters(myFilters);
```

filter.highlightColor

Disponibilidad

Flash 8.

Uso

```
filter.highlightColor
```

Descripción

Propiedad; el color del resaltado, en uno de los formatos siguientes:

- Una cadena con el formato `"#RRGGBB"` o `"#RRGGBBAA"`
- Un número hexadecimal con el formato `0xRRGGBB`
- Un entero que representa el equivalente decimal del número hexadecimal

Esta propiedad se define para los objetos Filter con un valor de `"bevelFilter"` para la propiedad `filter.name`.

Ejemplo

El ejemplo siguiente establece el color de resaltado en "#ff00003e" para los filtros de bisel en el objeto u objetos seleccionados:

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
  if(myFilters[i].name == 'bevelFilter'){
    myFilters[i].highlightColor = '#ff00003e';
  }
}
fl.getDocumentDOM().setFilters(myFilters);
```

filter.hue

Disponibilidad

Flash 8.

Uso

```
filter.hue
```

Descripción

Propiedad; un valor flotante que especifica el matiz del filtro. Los valores aceptables están entre -180 y 180. Esta propiedad se define para los objetos Filter con un valor de "adjustColorFilter" para la propiedad filter.name.

Ejemplo

El ejemplo siguiente establece el matiz en 120 para los filtros de ajustar color en el objeto u objetos seleccionados:

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
  if(myFilters[i].name == 'adjustColorFilter'){
    myFilters[i].hue = 120;
  }
}
fl.getDocumentDOM().setFilters(myFilters);
```

filter.inner

Disponibilidad

Flash 8.

Uso

```
filter.inner
```

Descripción

Propiedad; valor booleano que especifica si la sombra es interior (`true`) o no (`false`). Esta propiedad se define para los objetos Filter con un valor de `"dropShadowFilter"` o `"glowFilter"` para la propiedad `filter.name`.

Ejemplo

El ejemplo siguiente establece el valor de la propiedad `inner` como `true` para los filtros de iluminado en el objeto u objetos seleccionados:

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
  if(myFilters[i].name == 'glowFilter'){
    myFilters[i].inner = true;
  }
}
fl.getDocumentDOM().setFilters(myFilters);
```

Véase también

`document.setFilterProperty()`

filter.knockout

Disponibilidad

Flash 8.

Uso

```
filter.knockout
```

Descripción

Propiedad; valor booleano que especifica si el filtro es extractor (`true`) o no (`false`). Esta propiedad se define para los objetos Filter con un valor de `"bevelFilter"`, `"dropShadowFilter"`, `"glowFilter"`, `"gradientBevelFilter"` o `"gradientGlowFilter"` para la propiedad `filter.name`.

Ejemplo

El ejemplo siguiente establece la propiedad `knockout` como `true` para los filtros de iluminado en el objeto u objetos seleccionados:

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
  if(myFilters[i].name == 'glowFilter'){
    myFilters[i].knockout = true;
  }
}
fl.getDocumentDOM().setFilters(myFilters);
```

Véase también

`document.setFilterProperty()`

filter.name

Disponibilidad

Flash 8.

Uso

```
filter.name
```

Descripción

Propiedad de sólo lectura; una cadena que especifica el tipo de filtro. El valor de esta propiedad determina las demás propiedades del objeto Filter que están disponibles. El valor es uno de los siguientes: `"adjustColorFilter"`, `"bevelFilter"`, `"blurFilter"`, `"dropShadowFilter"`, `"glowFilter"`, `"gradientBevelFilter"` o `"gradientGlowFilter"`.

Ejemplo

El ejemplo siguiente muestra los nombres del filtro y las posiciones de índice en el panel Salida:

```
var myFilters = fl.getDocumentDOM().getFilters();
var traceStr = "";
for(i=0; i < myFilters.length; i++){
  traceStr = traceStr + " At index " + i + ": " + myFilters[i].name;
}
fl.trace(traceStr);
```

Véase también

`document.getFilters()`, `document.setFilterProperty()`

filter.quality

Disponibilidad

Flash 8.

Uso

```
filter.quality
```

Descripción

Propiedad; una cadena que especifica la calidad del desenfoque. Los valores aceptables son: `"low"`, `"medium"` y `"high"` (`"high"` es similar a un desenfoque gaussiano). Esta propiedad se define para los objetos Filter con un valor de `"bevelFilter"`, `"blurFilter"`, `"dropShadowFilter"`, "`glowFilter`", `"gradientGlowFilter"` o `"gradientBevelFilter"` para la propiedad `filter.name`.

Ejemplo

El ejemplo siguiente establece la calidad del desenfoque en `"medium"` para los filtros de iluminado en el objeto u objetos seleccionados:

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
  if(myFilters[i].name == 'glowFilter'){
    myFilters[i].quality = 'medium';
  }
}
fl.getDocumentDOM().setFilters(myFilters);
```

Véase también

`document.setFilterProperty()`

filter.saturation

Disponibilidad

Flash 8.

Uso

```
filter.saturation
```

Descripción

Propiedad; un valor flotante que especifica el valor de saturación del filtro. Los valores aceptables están entre -100 y 100. Esta propiedad se define para los objetos Filter con un valor de `"adjustColorFilter"` para la propiedad `filter.name`.

Ejemplo

El ejemplo siguiente establece el valor de saturación en 0 (escala de grises) para los filtros de ajustar color en el objeto u objetos seleccionados:

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
  if(myFilters[i].name == 'adjustColorFilter'){
    myFilters[i].saturation = 0;
  }
}
fl.getDocumentDOM().setFilters(myFilters);
```

Véase también

`document.setFilterProperty()`

filter.shadowColor

Disponibilidad

Flash 8.

Uso

```
filter.shadowColor
```

Descripción

Propiedad; el color de la sombra, en uno de los formatos siguientes:

- Una cadena con el formato `"#RRGGBB"` o `"#RRGGBBAA"`
- Un número hexadecimal con el formato `0xRRGGBB`
- Un entero que representa el equivalente decimal del número hexadecimal

Esta propiedad se define para los objetos Filter con un valor de `"bevelFilter"` para la propiedad `filter.name`.

Ejemplo

El ejemplo siguiente establece el color de sombra en `"#ff00003e"` para los filtros de bisel en el objeto u objetos seleccionados:

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
  if(myFilters[i].name == 'bevelFilter'){
    myFilters[i].shadowColor = '#ff00003e';
  }
}
fl.getDocumentDOM().setFilters(myFilters);
```

Véase también

`document.setFilterProperty()`

filter.strength

Disponibilidad

Flash 8.

Uso

```
filter.strength
```

Descripción

Propiedad; un entero que especifica el porcentaje de intensidad del filtro. Los valores aceptables están entre 0 y 25.500. Esta propiedad se define para los objetos Filter con un valor de `"bevelFilter"`, `"dropShadowFilter"`, `"glowFilter"`, `"gradientGlowFilter"` o `"gradientBevelFilter"` para la propiedad `filter.name`.

Ejemplo

El ejemplo siguiente establece la intensidad en 50 para los filtros de iluminado en el objeto u objetos seleccionados:

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
  if(myFilters[i].name == 'glowFilter'){
    myFilters[i].strength = 50;
  }
}
fl.getDocumentDOM().setFilters(myFilters);
```

Véase también

`document.setFilterProperty()`

filter.type

Disponibilidad

Flash 8.

Uso

```
filter.type
```

Descripción

Propiedad; una cadena que especifica el tipo de bisel o iluminado. Los valores válidos son: `"inner"`, `"outer"` y `"full"`. Esta propiedad se define para los objetos Filter con un valor de `"bevelFilter"`, `"gradientGlowFilter"` o `"gradientBevelFilter"` para la propiedad `filter.name`.

Ejemplo

El ejemplo siguiente establece el tipo en en`"full"` para los filtros de bisel en el objeto u objetos seleccionados:

```
var myFilters = fl.getDocumentDOM().getFilters();
for(i=0; i < myFilters.length; i++){
  if(myFilters[i].name == 'bevelFilter'){
    myFilters[i].type = 'full';
  }
}
fl.getDocumentDOM().setFilters(myFilters);
```

Véase también

`document.setFilterProperty()`

Objeto Flash (fl)

Disponibilidad

Flash MX 2004.

Descripción

El objeto Flash representa la aplicación Flash. Puede utilizar `flash` o `fl` para hacer referencia a este objeto. Esta documentación utiliza `fl`.

Resumen de métodos del objeto Flash

Pueden emplearse los métodos siguientes con el objeto Flash.

Método	Descripción
`fl.browseForFileURL()`	Abre el cuadro de diálogo del sistema Abrir archivo o Guardar archivo y permite que el usuario especifique un archivo para abrir o guardar.
`fl.browseForFolderURL()`	Muestra el cuadro de diálogo Buscar carpeta y permite que el usuario seleccione una carpeta.
`fl.closeAll()`	Cierra todos los documentos abiertos y muestra el cuadro de diálogo Guardar como para los documentos que no se han guardado previamente.
`fl.closeDocument()`	Cierra el documento especificado.
`fl.closeProject()`	Cierra el archivo de proyecto de Flash (FLP) que está abierto.
`fl.createDocument()`	Abre un documento nuevo y lo selecciona.
`fl.createProject()`	Crea un archivo de proyecto de Flash (FLP) con el nombre especificado.
`fl.enableImmediateUpdates()`	Permite que el desarrollador de scripts active actualizaciones visuales inmediatas de la línea de tiempo cuando ejecute efectos.
`fl.fileExists()`	Comprueba si el archivo ya existe en el disco.
`fl.findDocumentIndex()`	Devuelve una matriz de enteros que representan la posición de un documento en la matriz `fl.documents`.
`fl.getAppMemoryInfo()`	Devuelve un entero que representa el número de bytes que se están utilizando en un área especificada de memoria Flash.exe.
`fl.getDocumentDOM()`	Recupera el DOM (Objeto Document) del documento activo actualmente.

Método	Descripción
fl.getProject()	Devuelve un objeto Project que representa el proyecto abierto actualmente.
fl.mapPlayerURL()	Asigna una URL Unicode de escape a una URL UTF-8 o MBCS.
fl.openDocument()	Abre un documento de Flash (FLA) para editarlo en una nueva ventana de documento de Flash y lo selecciona.
fl.openProject()	Abre un archivo de proyecto de Flash (FLP) en la herramienta de edición de Flash para editarlo.
fl.openScript()	Abre un archivo de script (JSFL, AS, ASC) o de otro tipo (XML, TXT) en el editor de texto de Flash.
fl.quit()	Sale de Flash y pregunta al usuario si desea guardar los documentos modificados.
fl.reloadEffects()	Vuelve a cargar todos los descriptores de efectos definidos en la carpeta Configuration Effects del usuario.
fl.reloadTools()	Recrea el panel Herramientas a partir del archivo toolconfig.xml. Sólo se utiliza para crear herramientas ampliables.
fl.revertDocument()	Devuelve el documento FLA especificado a la última versión guardada.
fl.runScript()	Ejecuta un archivo JavaScript.
fl.saveAll()	Guarda todos los documentos abiertos y muestra el cuadro de diálogo Guardar como para los documentos que no se han guardado previamente.
fl.saveDocument()	Guarda el documento especificado como documento FLA.
fl.saveDocumentAs()	Muestra el cuadro de diálogo Guardar como para el documento especificado.
fl.setActiveWindow()	Establece el documento especificado como ventana activa.
fl.showIdleMessage()	Permite desactivar la advertencia sobre la ejecución demasiado larga de un script.
fl.trace()	Envía una cadena de texto al panel Salida.

Resumen de propiedades del objeto Flash

Pueden emplearse las propiedades siguientes con el objeto Flash.

Propiedades	Descripción
`fl.activeEffect`	De sólo lectura; el Objeto Effect para el efecto que se está aplicando.
`fl.componentsPanel`	De sólo lectura; un Objeto componentsPanel que representa el panel Componentes.
`fl.configDirectory`	De sólo lectura; una cadena que especifica la ruta completa de la carpeta Configuration del usuario local con el formato específico de la plataforma.
`fl.configURI`	De sólo lectura; una cadena que especifica la ruta completa de la carpeta Configuration del usuario local como archivo:/// URI.
`fl.contactSensitiveSelection`	Un valor booleano que especifica si el modo de selección Por contacto está activado.
`fl.createNewDocList`	De sólo lectura; una matriz de cadenas que representa los distintos tipos de documentos que se pueden crear.
`fl.createNewDocListType`	De sólo lectura; una matriz de cadenas que representa las extensiones de archivo de los tipos de documentos que se pueden crear.
`fl.createNewTemplateList`	De sólo lectura; una matriz de cadenas que representa los distintos tipos de plantillas que se pueden crear.
`fl.documents`	De sólo lectura; una matriz de objetos Document (véase Objeto Document) que representa los documentos (archivos FLA) que están abiertos para editar.
`fl.drawingLayer`	De sólo lectura; el Objeto drawingLayer que una herramienta ampliable debe utilizar cuando el usuario desea dibujar temporalmente mientras arrastra el ratón.
`fl.effects`	De sólo lectura; una matriz de objetos Effect (véase Objeto Effect) basada en el archivo de parámetros XML.
`fl.Math`	De sólo lectura; el Objeto Math que proporciona métodos para operaciones de matrices y puntos.
`fl.mruRecentFileList`	De sólo lectura; una matriz de los nombres completos de archivo de la lista Usados recientemente (MRU) que gestiona la herramienta de edición de Flash.
`fl.mruRecentFileListType`	De sólo lectura; una matriz de los tipos de archivo de la lista MRU que gestiona la herramienta de edición de Flash.

Propiedades	Descripción
`fl.objectDrawingMode`	Un valor booleano que especifica si el modelo de dibujo de objeto está activado.
`fl.outputPanel`	De sólo lectura; referencia al Objeto outputPanel.
`fl.tools`	De sólo lectura; una matriz de objetos Tools.
`fl.version`	De sólo lectura; la versión de cadena larga de la herramienta de edición de Flash, incluida la plataforma.
`fl.xmlui`	De sólo lectura; un Objeto XMLUI.

fl.activeEffect

Disponibilidad

Flash MX 2004.

Uso

```
fl.activeEffect
```

Descripción

Propiedad de sólo lectura; el Objeto Effect para el efecto que se está aplicando. Para ver una lista de propiedades disponibles para `fl.activeEffect`, consulte "Resumen de propiedades del objeto Effect" en la página 194.

Ejemplo

El ejemplo siguiente almacena un objeto que representa el efecto actual de la variable `ef`.

```
var ef = fl.activeEffect;
```

fl.browseForFileURL()

Disponibilidad

Flash MX 2004.

Uso

```
fl.browseForFileURL( browseType [, title [, previewArea ] ])
```

Parámetros

browseType Una cadena que especifica el tipo de operación de búsqueda de archivo. Los valores aceptables son: `"open"`, `"select"` y `"save"`. Los valores `"open"` y `"select"` abren el cuadro de diálogo del sistema Abrir archivo. Cada valor se suministra por compatibilidad con Dreamweaver. El valor `"save"` abre un cuadro de diálogo del sistema Guardar archivo.

title Una cadena que especifica el título del cuadro de diálogo Abrir archivo o Guardar archivo. Si se omite este parámetro se empleará un valor predeterminado. Este parámetro es opcional.

previewArea Un parámetro opcional que ignoran Flash y Fireworks y sólo está presente por compatibilidad con Dreamweaver.

Valor devuelto

La URL del archivo, expresada como archivo:/// URI; devuelve `null` si el usuario cancela el cuadro de diálogo.

Descripción

Método; abre el cuadro de diálogo del sistema Abrir archivo o Guardar archivo y permite que el usuario especifique un archivo para abrir o guardar.

Ejemplo

El ejemplo siguiente permite que el usuario elija un archivo FLA para abrir y, a continuación, abre el archivo. (El método `fl.browseForFileURL()` puede buscar cualquier tipo de archivo, pero `fl.openDocument()` sólo puede abrir archivos FLA.)

```
var fileURL = fl.browseForFileURL("open", "Select file");
var doc = fl.openDocument(fileURL);
```

Véase también

`fl.browseForFolderURL()`

fl.browseForFolderURL()

Disponibilidad

Flash 8.

Uso

```
fl.browseForFolderURL( [ description ] )
```

Parámetros

description Una cadena opcional que especifica la descripción del cuadro de diálogo Buscar carpeta. Si se omite este parámetro, no se mostrará nada en el área de descripción.

Valor devuelto

La URL de la carpeta, expresada como archivo:/// URI; devuelve `null` si el usuario cancela el cuadro de diálogo.

Descripción

Método; muestra el cuadro de diálogo Buscar carpeta y permite que el usuario seleccione una carpeta.

NOTA	El título del cuadro de diálogo es siempre “Buscar carpeta”. Utilice el parámetro description para añadir más detalle en el área de descripción debajo del título, como "Seleccione una carpeta" o "Seleccione la ruta que contiene los archivos .as de definición de clase que desea importar".

Ejemplo

El ejemplo siguiente permite que el usuario seleccione una carpeta y, a continuación, muestra una lista de archivos de esa carpeta.

```
var folderURI = fl.browseForFolderURL("Select a folder.");
var folderContents = FLfile.listFolder(folderURI);
```

Véase también

`fl.browseForFileURL()`, Objeto FLfile

fl.closeAll()

Disponibilidad

Flash MX 2004.

Uso

```
fl.closeAll()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; cierra todos los documentos abiertos y muestra el cuadro de diálogo Guardar como para los documentos que no se han guardado previamente. El método muestra un mensaje al usuario, si es necesario, pero no cierra la aplicación. Véase también `fl.closeDocument()`.

Ejemplo

El código siguiente cierra todos los documentos abiertos.

```
fl.closeAll();
```

fl.closeDocument()

Disponibilidad

Flash MX 2004.

Uso

```
fl.closeDocument( documentObject [, bPromptToSaveChanges] )
```

Parámetros

documentObject, [*bPromptToSaveChanges*]

documentObject Un Objeto Document. Si *documentObject* hace referencia al documento activo, es posible que la ventana Documento no se cierre hasta que termine de ejecutarse el script que llama a este método.

bPromptToSaveChanges Un valor booleano. Si es `false`, no se mostrará un mensaje al usuario si el documento contiene cambios no guardados; es decir, el archivo se cierra y los cambios se descartan. Si el valor es `true` y el documento contiene cambios no guardados, el usuario recibirá un mensaje con el cuadro de diálogo estándar con Sí y No. El valor predeterminado es `true`. Este parámetro es opcional.

Valor devuelto

Un valor booleano: `true` si es correcto, y `false` en caso contrario.

Descripción

Método; cierra el documento especificado. Véase también `fl.closeAll()`.

Ejemplo

El ejemplo siguiente ilustra dos formas de cerrar un documento.

```
// Cierra el documento especificado y pregunta al usuario si desea guardar
  los cambios.
fl.closeDocument(fl.documents[0]);
fl.closeDocument(fl.documents[0] , true); // El uso de true es opcional.
// Cierra el documento especificado sin preguntar al usuario si desea
  guardar los cambios.
fl.closeDocument(fl.documents[0], false);
```

fl.closeProject()

Disponibilidad

Flash 8.

Uso

```
fl.closeProject()
```

Parámetros

Ninguno.

Valor devuelto

Un valor booleano de `true` si el proyecto se cerró correctamente; `false` si no hay ningún archivo de proyecto abierto.

Descripción

Método; cierra el archivo de proyecto de Flash (FLP) que está abierto.

El ejemplo siguiente intenta cerrar un archivo de proyecto y muestra un mensaje que indica si el archivo se ha cerrado correctamente.

```
fl.trace("The project was" + (fl.closeProject() ? "closed" : "not
  closed"));
```

Véase también

`fl.getProject()`, `fl.openProject()`, Objeto Project

fl.componentsPanel

Disponibilidad

Flash MX 2004.

Uso

```
fl.componentsPanel
```

Descripción

Propiedad de sólo lectura; un Objeto componentsPanel que representa el panel Componentes.

Ejemplo

El ejemplo siguiente almacena un objeto componentsPanel en la variable `comPanel`.

```
var comPanel = fl.componentsPanel;
```

fl.configDirectory

Disponibilidad

Flash MX 2004.

Uso

```
fl.configDirectory
```

Descripción

Propiedad de sólo lectura; una cadena que especifica la ruta completa de la carpeta Configuration del usuario local con el formato específico de la plataforma. Para especificar esta ruta como un archivo:/// URI, que no es específico de la plataforma, utilice `fl.configURI`.

Ejemplo

El ejemplo siguiente muestra el directorio Configuration en el panel Salida.

```
fl.trace( "My local configuration directory is " + fl.configDirectory );
```

fl.configURI

Disponibilidad

Flash MX 2004.

Uso

```
fl.configURI
```

Descripción

Propiedad de sólo lectura; una cadena que especifica la ruta completa de la carpeta Configuration del usuario local como archivo:/// URI. Véase también `fl.configDirectory`.

Ejemplo

El ejemplo siguiente ejecuta un script especificado. El uso de `fl.configURI` permite especificar la ubicación del script sin necesidad de saber en qué plataforma se está ejecutando el script.

```
// Para ejecutar un comando en el menú de comandos, cambie "Test.Jsfl"
// al comando que desea ejecutar en la línea siguiente.
fl.runScript( fl.configURI + "Commands/Test.jsfl" );
```

fl.contactSensitiveSelection

Disponibilidad

Flash 8.

Uso

```
fl.contactSensitiveSelection
```

Descripción

Valor booleano que especifica si el modo de selección Por contacto está activado (`true`) o no (`false`).

Ejemplo

El siguiente ejemplo muestra cómo desactivar el modo de selección Por contacto antes de realizar una selección y luego cómo restablecerlo a su valor original después de realizar la selección.

```
var contact = fl.contactSensitiveSelection;
fl.contactSensitiveSelection = false;
// Introducir aquí el código de selección.
fl.contactSensitiveSelection = contact;
```

fl.createDocument()

Disponibilidad

Flash MX 2004.

Uso

```
fl.createDocument( [docType] )
```

Parámetros

`docType` Una cadena que especifica el tipo de documento que se va a crear. Los valores aceptables son: `"timeline"`, `"presentation"` y `"application"`. El valor predeterminado es `"timeline"`. Este parámetro es opcional.

Valor devuelto

El objeto Document para el documento recién creado, si el método es correcto. Si se produce un error, el valor es `undefined`.

Descripción

Método; abre un documento nuevo y lo selecciona. Los valores de tamaño, resolución y color son los predeterminados.

Ejemplo

El ejemplo siguiente crea distintos tipos de documentos.

```
// Crea un documento Flash basado en la línea de tiempo.
fl.createDocument();
fl.createDocument("timeline");
// Crea un documento de presentación de diapositivas.
fl.createDocument("presentation");
// Crea un documento de aplicación de formularios.
fl.createDocument("application");
```

fl.createNewDocList

Disponibilidad

Flash MX 2004.

Uso

```
fl.createNewDocList
```

Descripción

Propiedad de sólo lectura; una matriz de cadenas que representa los distintos tipos de documentos que se pueden crear.

Ejemplo

El ejemplo siguiente muestra los tipos de documentos que se pueden crear en el panel Salida.

```
fl.trace("Number of choices " + fl.createNewDocList.length);
for (i = 0; i < fl.createNewDocList.length; i++)
  fl.trace("choice: " + fl.createNewDocList[i]);
```

fl.createNewDocListType

Disponibilidad

Flash MX 2004.

Uso

```
fl.createNewDocListType
```

Descripción

Propiedad de sólo lectura; una matriz de cadenas que representa las extensiones de archivo de los tipos de documentos que se pueden crear. Las entradas de la matriz corresponden directamente (por índice) a las entradas de la matriz `fl.createNewDocList`.

Ejemplo

El ejemplo siguiente muestra las extensiones de los tipos de documentos que se pueden crear en el panel Salida.

```
fl.trace("Number of types " + fl.createNewDocListType.length);
for (i = 0; i < fl.createNewDocListType.length; i++) fl.trace("type: " +
  fl.createNewDocListType[i]);
```

fl.createNewTemplateList

Disponibilidad

Flash MX 2004.

Uso

```
fl.createNewTemplateList
```

Descripción

Propiedad de sólo lectura; una matriz de cadenas que representa los distintos tipos de plantillas que se pueden crear.

Ejemplo

El ejemplo siguiente muestra los tipos de plantillas que se pueden crear en el panel Salida.

```
fl.trace("Number of template types: " + fl.createNewTemplateList.length);
  for (i = 0; i < fl.createNewTemplateList.length; i++) fl.trace("type: " +
  fl.createNewTemplateList[i]);
```

fl.createProject()

Disponibilidad

Flash 8.

Uso

```
fl.createProject( fileURI [ , name ] )
```

Parámetros

`fileURI` Una cadena, expresada como archivo:/// URI, que especifica el nombre de proyecto Flash (FLP) que se va a crear.

`name` Una cadena opcional que se muestra como el nombre del proyecto en el panel Proyecto. Si se omite *`name`*, se mostrará el nombre del archivo FLP (excluida la ruta o la extensión) en el panel Proyecto.

Valor devuelto

Un Objeto Project si el método es correcto; `undefined` si no se puede crear el archivo (por ejemplo, *`fileURI`* contiene un directorio que no existe).

Descripción

Método; crea un archivo de proyecto de Flash (FLP) con el nombre especificado. Si no se puede crear el archivo, se muestra un cuadro de diálogo informativo. Si ya existe el archivo, se muestra un cuadro de diálogo en el que se le pregunta si desea sobrescribir el archivo.

Ejemplo

El ejemplo siguiente crea un archivo de proyecto en el directorio especificado (si existe) y especifica un nombre para mostrar en el panel Proyecto.

```
var myProject = fl.createProject("file:///C|/Projects/
  MasterProject_2005.flp", "Master Project");
```

Véase también

`fl.getProject()`, `fl.openProject()`, Objeto Project

fl.documents

Disponibilidad

Flash MX 2004.

Uso

```
fl.documents
```

Descripción

Propiedad de sólo lectura; una matriz de objetos Document (véase Objeto Document) que representa los documentos (archivos FLA) que están abiertos para editar.

Ejemplo

El ejemplo siguiente almacena una matriz de documentos abiertos en la variable `docs`.

```
var docs = fl.documents;
```

El ejemplo siguiente muestra los nombres de los documentos abiertos en el panel Salida.

```
for (doc in fl.documents) {
  fl.trace(fl.documents[doc].name);
}
```

fl.drawingLayer

Disponibilidad

Flash MX 2004.

Uso

```
fl.drawingLayer
```

Descripción

Propiedad de sólo lectura; el Objeto drawingLayer que una herramienta ampliable debe utilizar cuando el usuario desea dibujar temporalmente mientras arrastra el ratón (por ejemplo, para crear un recuadro de delimitación).

Ejemplo

Véase `drawingLayer.setColor()`.

fl.effects

Disponibilidad

Flash MX 2004.

Uso

```
fl.effects
```

Descripción

Propiedad de sólo lectura; una matriz de objetos Effect (véase Objeto Effect) basada en el archivo de parámetros XML. No son efectos, sino una descripción de los efectos. La longitud de la matriz corresponde al número de efectos (basado en los archivos de definición de parámetros XML, no en el número de archivos de implementación JSFL) registrados cuando se abre el programa.

Ejemplo

El siguiente ejemplo devuelve el primer efecto registrado:

```
ef = fl.effects[0]
```

fl.enableImmediateUpdates()

Disponibilidad

Flash MX 2004.

Uso

```
fl.enableImmediateUpdates(bEnableUpdates)
```

Parámetros

bEnableUpdates Un valor booleano que especifica si se activan (`true`) o desactivan (`false`) actualizaciones visuales inmediatas de la línea de tiempo cuando se ejecutan efectos.

Valor devuelto

Ninguno.

Descripción

Método; permite que el desarrollador de scripts active actualizaciones visuales inmediatas de la línea de tiempo cuando ejecute efectos. Las actualizaciones inmediatas suelen suprimirse para que el usuario no vea pasos intermedios que puedan distraerle visualmente y puedan dar la sensación de que el efecto tarda más de lo necesario. Este método sirve fundamentalmente para depurar y no deberá utilizarse en efectos que se despliegan sobre el terreno. Cuando termina el efecto, el estado interno se restablece para suprimir las actualizaciones inmediatas.

Ejemplo

El ejemplo siguiente activa las actualizaciones inmediatas.

```
fl.enableImmediateUpdates(true) ;
fl.trace("Immediate updates are enabled");
```

fl.fileExists()

Disponibilidad

Flash MX 2004.

Uso

```
fl.fileExists( fileURI )
```

Parámetros

fileURI Una cadena, expresada como archivo:/// URI, que contiene la ruta al archivo.

Valor devuelto

Un valor booleano: `true` si el archivo se encuentra en el disco; `false` en caso contrario.

Descripción

Método; comprueba si el archivo ya existe en el disco.

Ejemplo

El ejemplo siguiente muestra `true` o `false` en el panel Salida para cada archivo especificado, en función de si existe el archivo.

```
alert(fl.fileExists("file:///C|/example.fla"));
alert(fl.fileExists("file:///C|/example.jsfl"));
alert(fl.fileExists(""));
```

fl.findDocumentIndex()

Disponibilidad

Flash MX 2004.

Uso

```
fl.findDocumentIndex( name )
```

Parámetros

`name` El nombre del documento para el que desea buscar el índice. El documento debe estar abierto.

Valor devuelto

Una matriz de enteros que representan la posición del *`nombre`* del documento en la matriz `fl.documents`.

Descripción

Método; devuelve una matriz de enteros que representan la posición del *`nombre`* del documento en la matriz `fl.documents`. Se pueden abrir varios documentos con el mismo nombre (si están ubicados en distintas carpetas).

Ejemplo

El ejemplo siguiente muestra información sobre la posición del índice de todos los archivos abiertos denominados test.fla en el panel Salida:

```
var filename = "test.fla"
var docIndex = fl.findDocumentIndex(filename);
for (var index in docIndex)
  fl.trace(filename + " is open at index " + docIndex[index]);
```

Véase también

`fl.documents`

fl.getAppMemoryInfo()

Disponibilidad

Flash 8 (sólo Windows).

Uso

```
fl.getAppMemoryInfo( memType )
```

Parámetros

memType Un entero que especifica el área de uso de la memoria que se va a consultar. Para ver una lista de los valores aceptables, consulte la descripción siguiente.

Valor devuelto

Un entero que representa el número de bytes que se están utilizando en un área especificada de memoria Flash.exe.

Descripción

Método (sólo Windows); devuelve un entero que representa el número de bytes que se están utilizando en un área especificada de memoria Flash.exe. Utilice la tabla siguiente para determinar qué valor desea pasar como *memType*.

memType	Datos de recursos
0	PAGEFAULTCOUNT
1	PEAKWORKINGSETSIZE
2	WORKINGSETSIZE
3	QUOTAPEAKPAGEDPOOLUSAGE
4	QUOTAPAGEDPOOLUSAGE
5	QUOTAPEAKNONPAGEDPOOLUSAGE
6	QUOTANONPAGEDPOOLUSAGE
7	PAGEFILEUSAGE
8	PEAKPAGEFILEUSAGE

Ejemplo

El ejemplo siguiente muestra el consumo actual de memoria de trabajo.

```
var memsize = fl.getAppMemoryInfo(2);
fl.trace("Flash current memory consumption is " + memsize + " bytes or " +
  memsize/1024 + " KB");
```

fl.getDocumentDOM()

Disponibilidad

Flash MX 2004.

Uso

```
fl.getDocumentDOM()
```

Parámetros

Ninguno.

Valor devuelto

Un objeto Document o `null` si no hay documentos abiertos.

Descripción

Método; recupera el DOM (Objeto Document) del documento activo actualmente (archivo FLA). Si hay uno o varios documentos abiertos, pero no hay uno seleccionado (por ejemplo, hay un archivo JSFL seleccionado), recupera el DOM del documento que más recientemente ha estado activo.

Ejemplo

El ejemplo siguiente muestra el nombre del documento activo actualmente o que más recientemente ha estado activo en el panel Salida:

```
var currentDoc = fl.getDocumentDOM();
fl.trace(currentDoc.name);
```

fl.getProject()

Disponibilidad

Flash 8.

Uso

```
fl.getProject()
```

Parámetros

Ninguno.

Valor devuelto

Un Objeto Project que representa el proyecto abierto actualmente. Si no hay ningún proyecto abierto, devuelve `undefined`.

Descripción

Método; devuelve un Objeto Project que representa el proyecto abierto actualmente.

Ejemplo

El ejemplo siguiente muestra el nombre del proyecto abierto actualmente en el panel Salida.

```
fl.trace("Current project: " + fl.getProject().name);
```

Véase también

`fl.createProject()`, `fl.openProject()`, Objeto Project

fl.mapPlayerURL()

Disponibilidad

Flash MX 2004.

Uso

```
fl.mapPlayerURL( URI [, returnMBCS] )
```

Parámetros

`URI` Una cadena que contiene la URL Unicode de escape que se va a asignar.

`returnMBCS` Un valor booleano que debe definir como `true` si desea que se devuelva una ruta MBCS de escape. En caso contrario, el método devuelve UTF-8. El valor predeterminado es `false`. Este parámetro es opcional.

Valor devuelto

Una cadena que es la URL convertida.

Descripción

Método; asigna una URL Unicode de escape a una URL UTF-8 o MBCS. Utilice este método cuando la cadena se va a utilizar en ActionScript para acceder a un recurso externo. Deberá emplear este método si necesita gestionar caracteres de múltiples bytes.

Ejemplo

El ejemplo siguiente convierte una URL a UTF-8 para que el reproductor pueda cargarla.

```
var url = MMExecute( "fl.mapPlayerURL(" + myURL + ", false);" );
mc.loadMovie( url);
```

fl.Math

Disponibilidad

Flash MX 2004.

Uso

```
fl.Math
```

Descripción

Propiedad de sólo lectura; el Objeto Math proporciona métodos para operaciones de matrices y puntos.

Ejemplo

A continuación, se muestra la matriz de transformación del objeto seleccionado y su inversa.

```
// Selecciona un elemento del escenario y, a continuación, ejecuta este
  script.
var mat =fl.getDocumentDOM().selection[0].matrix;
for(var prop in mat){
  fl.trace("mat."+prop+" = " + mat[prop]);
}
var invMat = fl.Math.invertMatrix( mat );
for(var prop in invMat) {
fl.trace("invMat."+prop+" = " + invMat[prop]);
}
```

fl.mruRecentFileList

Disponibilidad

Flash MX 2004.

Uso

```
fl.mruRecentFileList
```

Descripción

Propiedad de sólo lectura; una matriz de los nombres completos de archivo de la lista Usados recientemente (MRU) que gestiona la herramienta de edición de Flash.

Ejemplo

El ejemplo siguiente muestra el número de archivos abiertos recientemente y el nombre de cada uno en el panel Salida.

```
fl.trace("Number of recently opened files: " +
  fl.mruRecentFileList.length);
for (i = 0; i < fl.mruRecentFileList.length; i++) fl.trace("file: " +
  fl.mruRecentFileList[i]);
```

fl.mruRecentFileListType

Disponibilidad

Flash MX 2004.

Uso

```
fl.mruRecentFileListType
```

Descripción

Propiedad de sólo lectura; una matriz de los tipos de archivo de la lista MRU que gestiona la herramienta de edición de Flash. Esta matriz corresponde a la matriz de la propiedad `fl.mruRecentFileList`.

Ejemplo

El ejemplo siguiente muestra el número de archivos abiertos recientemente y el tipo de cada uno en el panel Salida.

```
fl.trace("Number of recently opened files: " +
  fl.mruRecentFileListType.length);
for (i = 0; i < fl.mruRecentFileListType.length; i++) fl.trace("type: " +
  fl.mruRecentFileListType[i]);
```

fl.objectDrawingMode

Disponibilidad

Flash 8.

Uso

```
fl.objectDrawingMode
```

Descripción

Propiedad; un valor booleano que especifica si el modo de dibujo de objeto está activado (`true`) o lo está el modo de dibujo de fusión (`false`).

Ejemplo

El siguiente ejemplo cambia el estado del modo de dibujo de objeto:

```
var toggleMode = fl.objectDrawingMode;
if (toggleMode) {
  fl.objectDrawingMode = false;
} else {
  fl.objectDrawingMode = true;
}
```

fl.openDocument()

Disponibilidad

Flash MX 2004.

Uso

```
fl.openDocument( fileURI )
```

Parámetros

fileURI Una cadena, expresada como archivo:/// URI, que especifica el nombre del archivo que se va a abrir.

Valor devuelto

El Objeto Document para el documento recién abierto, si el método es correcto Si no se encuentra el archivo o no es un archivo FLA válido, se presenta un error y se cancela el script.

Descripción

Método; abre un documento de Flash (archivo FLA) para editarlo en una nueva ventana de documento de Flash y lo selecciona. Para un usuario, el efecto equivale a seleccionar Archivo > Abrir y, a continuación, seleccionar un archivo. Si el archivo especificado ya está abierto, la ventana que contiene el documento se coloca en primer plano. La ventana que contiene el archivo especificado se convierte en el documento seleccionado actualmente.

Ejemplo

El ejemplo siguiente abre un archivo llamado Document.fla que se almacena en el directorio raíz de la unidad C , almacena un objeto Document que representa ese documento en la variable `doc` y define el documento como el documento seleccionado actualmente. Es decir, hasta que cambie el enfoque, `fl.getDocumentDOM()` hace referencia a este documento.

```
var doc = fl.openDocument("file:///c|/Document.fla");
```

fl.openProject()

Disponibilidad

Flash MX 2004; valor devuelto cambiado en Flash 8.

Uso

```
fl.openProject( fileURI )
```

Parámetros

fileURI Una cadena, expresada como archivo:/// URI, que especifica la ruta del archivo de proyecto de Flash (FLP) que se va a abrir.

Valor devuelto

Nada en Flash MX 2004, un Objeto Project en Flash 8.

Descripción

Método; abre un archivo de proyecto de Flash (FLP) en la herramienta de edición de Flash para editarlo.

Ejemplo

El ejemplo siguiente abre un archivo de proyecto llamado myProjectFile.flp que se almacena en el directorio raíz de la unidad C .

```
fl.openProject("file:///c|/myProjectFile.flp");
```

Véase también

`fl.closeProject()`, `fl.createProject()`, `fl.getProject()`, Objeto Project

fl.openScript()

Disponibilidad

Flash MX 2004.

Uso

```
fl.openScript( fileURI )
```

Parámetros

fileURI Una cadena, expresada como archivo:/// URI, que especifica la ruta del archivo JSFL, AS, ASC, XML, TXT o de otro tipo que se debe cargar en el editor de texto de Flash.

Valor devuelto

Ninguno.

Descripción

Método; abre un archivo de script (JSFL, AS, ASC) o de otro tipo (XML, TXT) en el editor de texto de Flash.

Ejemplo

El ejemplo siguiente abre un archivo llamado my_test.jsfl que se almacena en el directorio / temp de la unidad C .

```
fl.openScript("file:///c|/temp/my_test.jsfl");
```

fl.outputPanel

Disponibilidad

Flash MX 2004.

Uso

```
fl.outputPanel
```

Descripción

Propiedad de sólo lectura; referencia al Objeto outputPanel.

Ejemplo

Véase Objeto outputPanel.

fl.quit()

Disponibilidad

Flash MX 2004.

Uso

```
fl.quit( [bPromptIfNeeded] )
```

Parámetros

bPromptIfNeeded Un valor booleano que es `true` (predeterminado) si desea que el usuario reciba un mensaje sobre si desea guardar los documentos modificados. Defina este parámetro como `false` si no desea que el usuario reciba un mensaje sobre si desea guardar los documentos modificados. En este último caso se descartarán las modificaciones realizadas en los documentos abiertos y la aplicación se cerrará inmediatamente. Aunque resulta útil para el procesamiento por lotes, utilice este método con precaución. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; sale de Flash y pregunta al usuario si desea guardar los documentos modificados.

Ejemplo

El ejemplo siguiente ilustra el cierre con la pregunta sobre si desea guardar los documentos modificados y sin ella.

```
// Cierra sin preguntar si desea guardar los documentos modificados.
fl.quit();
fl.quit(true); // True es opcional.
// Cierra sin guardar ningún archivo.
fl.quit(false);
```

fl.reloadEffects()

Disponibilidad

Flash MX 2004.

Uso

```
fl.reloadEffects()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; vuelve a cargar todos los descriptores de efectos definidos en la carpeta Configuration Effects del usuario. Permite cambiar rápidamente los script durante el desarrollo y proporciona un mecanismo para mejorar los efectos sin volver a iniciar la aplicación. Este método funciona mejor si se utiliza en un comando situado en la carpeta Commands.

Ejemplo

El ejemplo siguiente es un script de una línea que puede situar en la carpeta Commands. Cuando necesite volver a cargar los efectos, acceda al menú Comandos y ejecute el script.

```
fl.reloadEffects();
```

fl.reloadTools()

Disponibilidad

Flash MX 2004.

Uso

```
fl.reloadTools()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; recrea el panel Herramientas a partir del archivo toolconfig.xml. Este método sólo se utiliza para crear herramientas ampliables. Utilice este método cuando necesite volver a cargar el panel Herramientas, por ejemplo, después de modificar el archivo JSFL que define una herramienta que ya se encuentra en el panel.

Ejemplo

El ejemplo siguiente es un script de una línea que puede situar en la carpeta Commands. Cuando necesite volver a cargar el panel Herramientas, ejecute el script desde el menú Comandos.

```
fl.reloadTools();
```

fl.revertDocument()

Disponibilidad

Flash MX 2004.

Uso

```
fl.revertDocument( documentObject )
```

Parámetros

documentObject Un Objeto Document. Si *documentObject* hace referencia al documento activo, es posible que no se descarten los cambios en la ventana Documento hasta que termine de ejecutarse el script que llama a este método.

Valor devuelto

Un valor booleano: `true` si la operación Descartar cambios se realiza correctamente; `false` en caso contrario.

Descripción

Método; devuelve el documento FLA especificado a la última versión guardada. A diferencia de la opción del menú Archivo > Descartar cambios, este método no muestra una ventana de advertencia para preguntar al usuario si confirma la operación. Véase también `document.revert()` y `document.canRevert()`.

Ejemplo

El ejemplo siguiente devuelve el documento FLA actual a la última versión guardada; se perderán los cambios realizados desde la última vez que lo guardó.

```
fl.revertDocument(fl.getDocumentDOM());
```

fl.runScript()

Disponibilidad

Flash MX 2004.

Uso

```
fl.runScript( fileURI [, funcName [, arg1, arg2, ...] ])
```

Parámetros

`fileURI` Una cadena, expresada como archivo:/// URI, que especifica el nombre del archivo de script que se va a ejecutar.

`funcName` Una cadena que identifica una función que se va a ejecutar en el archivo JSFL que se especifica en *`fileURI`*. Este parámetro es opcional.

`arg` Un parámetro opcional que especifica uno o más argumentos que se van a transferir a *`funcname`*.

Valor devuelto

El resultado de la función como cadena, si se especifica *`funcName`*; de lo contrario, no devuelve nada.

Descripción

Método; ejecuta un archivo JavaScript. Si se especifica una función como uno de los argumentos, ejecuta la función y el código del script que no está dentro de la función. El resto del código del script se ejecuta antes de que se ejecute la función.

Ejemplo

Supongamos que hay un archivo de script llamado testScript.jsfl en el directorio raíz de la unidad C y su contenido es el siguiente:

```
function testFunct(num, minNum) {
  fl.trace("in testFunct: 1st arg: " + num + " 2nd arg: " + minNum);
}
for (i=0; i<2; i++) {
  fl.trace("in for loop i=" + i);
}
fl.trace("end of for loop");
// Final de testScript.jsfl
```

Si ejecuta el comando siguiente:

```
fl.runScript("file:///C|/testScript.jsfl", "testFunct", 10, 1);
```

Aparecerá la información siguiente en el panel Salida:

```
in for loop i=0
in for loop i=1
end of for loop
in testFunct: 1st arg: 10 2nd arg: 1
```

También puede llamar a testScript.jsfl sin ejecutar una función:

```
fl.runScript("file:///C|/testScript.jsfl");
```

lo que produce lo siguiente en el panel Salida:

```
in for loop i=0
in for loop i=1
end of for loop
```

fl.saveAll()

Disponibilidad

Flash MX 2004.

Uso

```
fl.saveAll()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; guarda todos los documentos abiertos.

NOTA	Si un archivo no se ha guardado nunca o no se ha modificado desde la última vez que se guardó, no se guardará. Para permitir que se guarde un archivo no guardado o no modificado, utilice `fl.saveDocumentAs()`.

Ejemplo

El ejemplo siguiente guarda todos los documentos abiertos.

```
fl.saveAll();
```

Véase también

`document.save()`, `document.saveAndCompact()`, `fl.saveDocument()`, `fl.saveDocumentAs()`

fl.saveDocument()

Disponibilidad

Flash MX 2004.

Uso

```
fl.saveDocument( document [, fileURI] )
```

Parámetros

`document` Un Objeto Document que especifica el documento que se va a guardar. Si *`document`* es `null`, se guardará el documento activo.

`fileURI` Una cadena, expresada como archivo:/// URI, que especifica el nombre del documento guardado. Si el parámetro *`fileURI`* es `null` o se omite, el documento se guardará con su nombre actual. Este parámetro es opcional.

Valor devuelto

Un valor booleano: `true` si la operación de guardado se realiza correctamente; `false` en caso contrario.

NOTA	Si el archivo no se ha guardado nunca o no se ha modificado desde la última vez que se guardó, no se guardará y el valor devuelto será false. Para permitir que se guarde un archivo no guardado o no modificado, utilice `fl.saveDocumentAs()`.

Descripción

Método; guarda el documento especificado como documento FLA.

Ejemplo

El ejemplo siguiente guarda el documento actual y dos documentos especificados.

```
// Guarda el documento actual.
alert(fl.saveDocument(fl.getDocumentDOM()));
// Guarda los documentos especificados.
alert(fl.saveDocument(fl.documents[0], "file:///C|/example1.fla"));
alert(fl.saveDocument(fl.documents[1],"file:///C|/example2.fla"));
```

Véase también

`document.save()`, `document.saveAndCompact()`, `fl.saveAll()`, `fl.saveDocumentAs()`

fl.saveDocumentAs()

Disponibilidad

Flash MX 2004.

Uso

`fl.saveDocumentAs(` *`document`* `)`

Parámetros

`document` Un Objeto Document que especifica el documento que se va a guardar. Si *`document`* es `null`, se guardará el documento activo.

Valor devuelto

Un valor booleano: `true` si la operación Guardar como se realiza correctamente; `false` en caso contrario.

Descripción

Método; muestra el cuadro de diálogo Guardar como para el documento especificado.

Ejemplo

El ejemplo siguiente pregunta al usuario si desea guardar el documento especificado y, a continuación, muestra un mensaje de alerta en el que se indica si el documento estaba guardado.

```
alert(fl.saveDocumentAs(fl.documents[1]));
```

Véase también

`document.save()`, `document.saveAndCompact()`, `fl.saveAll()`, `fl.saveDocument()`

fl.setActiveWindow()

Disponibilidad

Flash MX 2004.

Uso

```
fl.setActiveWindow( document [, bActivateFrame] )
```

Parámetros

document Un Objeto Document que especifica el documento que se va a seleccionar en la ventana activa.

bActivateFrame Un parámetro opcional que ignoran Flash y Fireworks y sólo está presente por compatibilidad con Dreamweaver.

Valor devuelto

Ninguno.

Descripción

Método; establece el documento especificado como ventana activa. Este método también se admite en Dreamweaver y Fireworks. Si el documento tiene múltiples vistas (creadas por Editar en nueva ventana), se seleccionará la primera vista.

Ejemplo

El ejemplo siguiente muestra dos formas de guardar un documento especificado.

```
fl.setActiveWindow(fl.documents[0]);

var theIndex = fl.findDocumentIndex("myFile.fla");
fl.setActiveWindow(fl.documents[theIndex]);
```

fl.showIdleMessage()

Disponibilidad

Flash 8.

Uso

```
fl.showIdleMessage( show )
```

Parámetros

show Un valor booleano que especifica si se activa o desactiva la advertencia sobre un script con ejecución demasiado larga.

Valor devuelto

Ninguno.

Descripción

Método; permite desactivar la advertencia sobre un script con ejecución demasiado larga (transfiere `false` para *`show`*). Puede utilizarlo cuando procese operaciones por lotes que tarden demasiado tiempo en terminar. Para volver a activar la alerta, ejecute de nuevo el comando, transfiriendo esta vez `true` para *`show`*.

Ejemplo

El ejemplo siguiente ilustra cómo se desactiva y vuelve a activar la advertencia sobre un script con ejecución demasiado larga.

```
fl.showIdleMessage(false);
var result = timeConsumingFunction();
fl.showIdleMessage(true);
```

fl.tools

Disponibilidad

Flash MX 2004.

Uso

```
fl.tools
```

Descripción

Propiedad de sólo lectura; una matriz de objetos Tools (véase Objeto Tools). Esta propiedad sólo se utiliza para crear herramientas ampliables.

fl.trace()

Disponibilidad

Flash MX 2004.

Uso

```
fl.trace( message )
```

Parámetros

`message` Una cadena que aparece en el panel Salida.

Valor devuelto

Ninguno.

Descripción

Método; envía una cadena de texto al panel Salida, terminada con una línea nueva y muestra el panel Salida si aún no está visible. Este método es idéntico a `outputPanel.trace()` y funciona igual que la sentencia `trace()` en ActionScript.

Para enviar una línea en blanco, utilice `fl.trace("")` o `fl.trace("\n")`. Puede utilizar estos comandos en línea, convirtiendo `\n` en parte de la cadena *`message`*.

Ejemplo

El ejemplo siguiente muestra varias líneas de texto en el panel Salida:

```
fl.outputPanel.clear();
fl.trace("Hello World!!!");
var myPet = "cat";
fl.trace("\nI have a " + myPet);
fl.trace("");
fl.trace("I love my " + myPet);
fl.trace("Do you have a " + myPet +"?");
```

fl.version

Disponibilidad

Flash MX 2004.

Uso

```
fl.version
```

Descripción

Propiedad de sólo lectura; la versión de cadena larga de la herramienta de edición de Flash, incluida la plataforma.

Ejemplo

El ejemplo siguiente muestra la versión de la herramienta de edición de Flash en el panel Salida.

```
alert( fl.version ); // Por ejemplo, WIN 7,0,0,380
```

fl.xmlui

Disponibilidad

Flash MX 2004.

Uso

```
fl.xmlui
```

Descripción

Propiedad de sólo lectura; un Objeto XMLUI. Esta propiedad permite obtener y establecer propiedades XMLUI en un cuadro de diálogo XMLUI y permite aceptar o cancelar el cuadro de diálogo de forma programada.

Ejemplo

Véase Objeto XMLUI.

Objeto FLfile

Disponibilidad

Flash MX 2004 7.2.

Descripción

El objeto FLfile permite escribir extensiones de Flash que pueden acceder, modificar y eliminar archivos y carpetas en el sistema de archivos local. La API de FLfile se suministra en forma de una extensión de la API JavaScript. Esta extensión se denomina *biblioteca compartida* y se encuentra en la carpeta siguiente:

- Windows 2000 o Windows XP:

  unidad de inicio\Documents and Settings*usuario*\Configuración local\Datos de programa\Macromedia\ Flash 8*idioma*\Configuration\External Libraries\FLfile.dll

- Mac OS X:

  Macintosh HD/Users/*nombreUsuario*/Library/Application Support/Macromedia/Flash 8/*idioma*/Configuration/External Libraries/FLfile.dll

NOTA	No confunda las bibliotecas compartidas que contienen símbolos en los documentos de Flash con las bibliotecas compartidas de la API JavaScript. Son dos tipos distintos de bibliotecas.

Los métodos FLfile funcionan con archivos o carpetas (directorios) del disco. Por tanto, cada método toma uno o varios parámetros que especifican la ubicación de un archivo o carpeta. La ubicación del archivo o la carpeta se expresa como una cadena de forma muy similar a la URL de un sitio Web. Se llama *URI (Id. uniforme de recursos) de archivo* y tiene el formato que se indica a continuación (incluidas las comillas):

```
"file:///drive|/folder 1/folder 2/.../filename"
```

Por ejemplo, si desea crear una carpeta en la unidad C llamada config y colocarla en la carpeta Archivos de programa/MyApp, utilice el comando siguiente:

```
FLfile.createFolder("file:///C|/Program Files/MyApp/config");
```

Si a continuación desea situar un archivo llamado config.ini en esa carpeta, utilice el comando siguiente:

```
FLfile.write("file:///C|/Program Files/MyApp/config/config.ini", "");
```

Para crear una carpeta en Macintosh, podría utilizar el comando siguiente:

```
FLfile.createFolder("file:///Macintosh/MyApp/config");
```

Resumen de métodos para el objeto FLfile

Pueden emplearse los métodos siguientes con el objeto FLfile.

Método	Descripción
`FLfile.copy()`	Copia un archivo.
`FLfile.createFolder()`	Crea una o varias carpetas.
`FLfile.exists()`	Determina la existencia de un archivo o una carpeta.
`FLfile.getAttributes()`	Detecta si un archivo es de escritura, de sólo lectura, oculto, visible o una carpeta del sistema.
`FLfile.getCreationDate()`	Especifica cuántos segundos han transcurrido entre el 1 de enero de 1970 y la hora de creación del archivo o carpeta.
`FLfile.getCreationDateObj()`	Obtiene la fecha de creación de un archivo o carpeta.
`FLfile.getModificationDate()`	Especifica cuántos segundos han transcurrido entre el 1 de enero de 1970 y la hora de modificación del archivo o carpeta.
`FLfile.getModificationDateObj()`	Obtiene la fecha de la última modificación de un archivo o carpeta.
`FLfile.getSize()`	Obtiene el tamaño de un archivo.
`FLfile.listFolder()`	Muestra el contenido de una carpeta.
`FLfile.read()`	Lee el contenido de un archivo.
`FLfile.remove()`	Elimina un archivo o carpeta.
`FLfile.setAttributes()`	Convierte un archivo o carpeta en de sólo lectura, de escritura, oculto o visible.
`FLfile.write()`	Crea, escribe o añade a un archivo.

FLfile.copy()

Disponibilidad

Flash MX 2004 7.2.

Uso

```
FLfile.copy( fileURI, copyURI )
```

Parámetros

fileURI Una cadena, expresada como archivo:/// URI, que especifica el archivo que desea copiar.

copyURI Una cadena, expresada como archivo:/// URI, que especifica la ubicación y el nombre del archivo copiado.

Valor devuelto

Un valor booleano de `true` si es correcto, y de `false` en caso contrario.

Descripción

Método; copia un archivo de una ubicación a otra. Este método devuelve `false` si ya existe *copyURI*.

Ejemplo

El ejemplo siguiente realiza una copia de seguridad de un archivo de configuración llamado config.ini y lo sitúa en la misma carpeta en la que se encuentra, pero con otro nombre.

```
var originalFileURI="file:///C|/Program Files/MyApp/config.ini";
var newFileURI="file:///C|/Program Files/MyApp/config_backup.ini";
Flfile.copy(originalFileURI, newFileURI);
```

Si lo prefiere, puede realizar la misma tarea con un solo comando:

```
Flfile.copy("file:///C|:/Program Files/MyApp/config.ini",
  file:///C|/Program Files/MyApp/config_backup.ini");
```

FLfile.createFolder()

Disponibilidad

Flash MX 2004 7.2.

Uso

```
FLfile.createFolder( folderURI )
```

Parámetros

folderURI Una URI de carpeta que especifica la carpeta que desea crear.

Valor devuelto

Un valor booleano de `true` si es correcto y de `false` si *folderURI* ya existe.

Descripción

Método; crea una o varias carpetas en la ubicación especificada.

Puede crear varias carpetas de una vez. Por ejemplo, el comando siguiente crea las carpetas MyData y TempData si aún no existen:

```
FLfile.createFolder("file:///c|/MyData/TempData")
```

Ejemplo

El ejemplo siguiente crea dos subcarpetas en la carpeta de configuración (fl.configURI).

```
fl.trace(FLfile.createFolder(fl.configURI+"folder01/subfolder01"));
```

El ejemplo siguiente intenta crear una carpeta llamada tempFolder en el nivel de la raíz de la unidad C y muestra un cuadro de alerta que indica si la operación ha tenido éxito.

```
var folderURI = "file:///c|/tempFolder";
if (FLfile.createFolder(folderURI)) {
  alert("Created " + folderURI);
}
else {
  alert(folderURI + " already exists");
}
```

Véase también

FLfile.remove(), FLfile.write()

FLfile.exists()

Disponibilidad

Flash MX 2004 7.2.

Uso

```
FLfile.exists( fileURI )
```

Parámetros

fileURI Una cadena, expresada como archivo:/// URI, que especifica el archivo que desea verificar.

Valor devuelto

Un valor booleano de `true` si es correcto, y de false en caso contrario.

Descripción

Método; determina si existe un archivo especificado.

Ejemplos

El ejemplo siguiente comprueba si hay un archivo llamado mydata.txt y muestra un cuadro de alerta que indica si el archivo existe.

```
var fileURI = "file:///c|/temp/mydata.txt";
if (FLfile.exists(fileURI)) {
  alert( fileURI + " exists!");
}
else {
  alert( fileURI + " does not exist.");
}
```

El ejemplo siguiente comprueba si existe un archivo de configuración necesario. Si el archivo no existe, lo crea:

```
var configFile = "file:///C|/MyApplication/config.ini";
if (!FLfile.exists(configFile)) {
     FLfile.write(configFile,"")
}
```

Véase también

`FLfile.write()`

FLfile.getAttributes()

Disponibilidad

Flash MX 2004 7.2.

Uso

`FLfile.getAttributes( fileOrFolderURI )`

Parámetros

`fileOrFolderURI` Una cadena, expresada como archivo:/// URI, que especifica el archivo o la carpeta cuyos atributos desea recuperar.

Valor devuelto

Una cadena que representa los atributos del archivo o carpeta que se ha especificado.

NOTA	Los resultados son impredecibles si el archivo o carpeta no existen. Deberá utilizar `FLfile.exists()` antes de emplear este método.

Descripción

Método; devuelve una cadena que representa los atributos del archivo o la carpeta que se ha especificado, o bien una cadena vacía si el archivo no tiene atributos específicos (es decir, no es de sólo lectura, no está oculto, etc.). Debe utilizar siempre `FLfile.exists()` para probar la existencia de un archivo o carpeta antes de emplear este método.

Los caracteres de la cadena representan los atributos siguientes:

- `R` — *`fileOrFolderURI`* es de sólo lectura
- `D` — *`fileOrFolderURI`* es una carpeta (directorio)
- `H` — *`fileOrFolderURI`* está oculto (sólo Windows)
- `S` — *`fileOrFolderURI`* es un archivo o carpeta del sistema (sólo Windows)
- `A` — *`fileOrFolderURI`* está listo para archivar (sólo Windows)

Por ejemplo, si *`fileOrFolderURI`* es una carpeta oculta, la cadena devuelta es `"DH"`.

Ejemplo

El ejemplo siguiente obtiene los atributos del archivo mydata.txt y muestra un cuadro de alerta si el archivo es de sólo lectura.

```
var URI = "file:///c|/temp/mydata.txt";
if (FLfile.exists(URI)){
  var attr = FLfile.getAttributes(URI);
    if (attr && (attr.indexOf("R") != -1)) { // La cadena devuelta
  contieneR
      alert(URI + " is read only!");
  }
}
```

Véase también.

`FLfile.setAttributes()`

FLfile.getCreationDate()

Disponibilidad

Flash MX 2004 7.2.

Uso

`FLfile.getCreationDate(`*`fileOrFolderURI`*`)`

Parámetros

`fileOrFolderURI` Una cadena, expresada como archivo:/// URI, que especifica el archivo o carpeta cuya fecha y hora de creación desea recuperar como cadena hexadecimal.

Valor devuelto

Una cadena que contiene un número hexadecimal que representa el número de segundos que han transcurrido entre el 1 de enero de 1970 y la hora de creación del archivo o carpeta, o bien `"00000000"` si no existe el archivo o carpeta.

Descripción

Método; especifica cuántos segundos han transcurrido entre el 1 de enero de 1970 y la hora de creación del archivo o carpeta. Este método se utiliza principalmente para comparar las fechas de creación o modificación de los archivos o carpetas.

Ejemplo

El ejemplo siguiente determina si un archivo se ha modificado desde que se creó.

```
// Asegúrese de que existe el archivo especificado.
var fileURI = "file:///C|/MyApplication/MyApp.fla";
var creationTime = FLfile.getCreationDate(fileURI)
var modificationTime = FLfile.getModificationDate(fileURI)
if ( modificationTime > creationTime ) {
     alert("The file has been modified since it was created")
}
else {
     alert("The file has not been modified since it was created")
}
```

Véase también

`FLfile.getCreationDateObj()`, `FLfile.getModificationDate()`

FLfile.getCreationDateObj()

Disponibilidad

Flash MX 2004 7.2.

Uso

`FLfile.getCreationDateObj(`*`fileOrFolderURI`*`)`

Parámetros

`fileOrFolderURI` Una cadena, expresada como archivo:/// URI, que especifica el archivo o carpeta cuya fecha y hora de creación desea recuperar como objeto Date de JavaScript.

Valor devuelto

Un objeto Date de JavaScript que representa la fecha y hora de creación de un archivo o carpeta especificada. Si el archivo no existe, el objeto contiene información que indica que el archivo o carpeta se creó a medianoche GMT el 31 de diciembre de 1969.

Descripción

Método; devuelve un objeto Date de JavaScript que representa la fecha y hora de creación de un archivo o carpeta especificada.

Ejemplo

El ejemplo siguiente muestra (con formato legible para el usuario) la fecha de creación de un archivo en el panel Salida:

```
// Asegúrese de que existe el archivo especificado.
var file1Date = FLfile.getCreationDateObj("file:///c|/temp/file1.txt");
fl.trace(file1Date);
```

Véase también

`FLfile.getCreationDate()`, `FLfile.getModificationDateObj()`

FLfile.getModificationDate()

Disponibilidad

Flash MX 2004 7.2.

Uso

`FLfile.getModificationDate(`*`fileOrFolderURI`*`)`

Parámetros

`fileOrFolderURI` Una cadena, expresada como archivo:/// URI, que especifica el archivo cuya fecha y hora de modificación desea recuperar como cadena hexadecimal.

Valor devuelto

Una cadena que contiene un número hexadecimal que representa el número de segundos que han transcurrido entre el 1 de enero de 1970 y la hora de modificación del archivo o carpeta, o bien `"00000000"` si no existe el archivo.

Descripción

Método; especifica cuántos segundos han transcurrido entre el 1 de enero de 1970 y la hora de la última modificación del archivo o carpeta. Este método se utiliza principalmente para comparar las fechas de creación o modificación de los archivos o carpetas.

Ejemplo

El ejemplo siguiente compara las fechas de modificación de dos archivos y determina cuál de los dos se modificó más recientemente:

```
// Asegúrese de que existe el archivo especificado.
file1 = "file:///C|/MyApplication/MyApp.fla"
file2 = "file:///C|/MyApplication/MyApp.as"
modificationTime1 = FLfile.getModificationDate(file1)
modificationTime2 = FLfile.getModificationDate(file2)

if(modificationTime1 > modificationTime2) {
     alert("File 2 is older than File 1")
}
else if(modificationTime1 < modificationTime2) {
     alert("File 1 is older than File 2")
}
else {
     alert("File 1 and File 2 were saved at the same time")
}
```

Véase también

`FLfile.getCreationDate()`, `FLfile.getModificationDateObj()`

FLfile.getModificationDateObj()

Disponibilidad

Flash MX 2004 7.2.

Uso

`FLfile.getModificationDateObj(`*`fileOrFolderURI`*`)`

Parámetros

`fileOrFolderURI` Una cadena, expresada como archivo:/// URI, que especifica el archivo o carpeta cuya fecha y hora de modificación desea recuperar como objeto Date de JavaScript.

Valor devuelto

Un objeto Date de JavaScript que representa la fecha y hora de la última modificación de un archivo o carpeta especificada. Si el archivo o carpeta no existe, el objeto contiene información que indica que el archivo o carpeta se creó a medianoche GMT el 31 de diciembre de 1969.

Descripción

Método; devuelve un objeto Date de JavaScript que representa la fecha y hora de la última modificación de un archivo o carpeta especificada.

Ejemplo

El ejemplo siguiente muestra (con formato legible para el usuario) la fecha de la última modificación de un archivo en el panel Salida:

```
// Asegúrese de que existe el archivo especificado.
var file1Date = FLfile.getModificationDateObj("file:///c|/temp/file1.txt");
trace(file1Date);
```

Véase también

`FLfile.getCreationDateObj()`, `FLfile.getModificationDate()`

FLfile.getSize()

Disponibilidad

Flash MX 2004 7.2.

Uso

`FLfile.getSize( fileURI )`

Parámetros

`fileURI` Una cadena, expresada como archivo:/// URI, que especifica el archivo cuyo tamaño desea recuperar.

Valor devuelto

Un entero que representa el tamaño del archivo especificado, en bytes, o bien 0 si no existe el archivo.

Descripción

Método; devuelve un entero que representa el tamaño del archivo especificado, en bytes, o bien 0 si no existe el archivo. Si el valor devuelto es 0, puede utilizar `FLfile.exists()` para determinar si el archivo es de byte cero o si no existe.

Ejemplo

El ejemplo siguiente almacena el tamaño del archivo mydata.txt en la variable `fileSize`:

```
var URL = "file:///c|/temp/mydata.txt";
var fileSize = FLfile.getSize(URL);
```

FLfile.listFolder()

Disponibilidad

Flash MX 2004 7.2.

Uso

```
FLfile.listFolder( folderURI [, filesOrDirectories ] )
```

Parámetros

folderURI Una cadena, expresada como archivo:/// URI, que especifica la carpeta cuyo contenido desea recuperar. Puede incluir una máscara de comodín como parte de *folderURI*. Los comodines válidos son: * (equivale a uno o varios caracteres) y ? (equivale a un solo carácter).

filesOrDirectories Una cadena opcional que especifica si sólo se devuelven nombres de archivo o sólo nombres de carpeta (directorio). Si se omite, se devuelven nombres de archivo y de carpeta. Los valores aceptables son: `"files"` y `"directories"`.

Valor devuelto

Una matriz de cadenas que representa el contenido de la carpeta o `false` si no existe la carpeta.

Descripción

Método; devuelve una matriz de cadenas que representa el contenido de la carpeta o una matriz vacía si no existe la carpeta.

Ejemplos

El ejemplo siguiente devuelve una matriz que representa los archivos, carpeta o archivos y carpetas del directorio Archivos de programa.

```
var folderURI = "file:///C|/WINDOWS/Program Files" ;
var fileList = FLfile.listFolder(folderURI, "files") // archivos
var fileList = FLfile.listFolder("folderURI", "directories") //carpetas
var fileList = FLfile.listFolder(folderURI) //archivos y carpetas
```

El ejemplo siguiente devuelve una matriz de todos los archivos de texto (.txt) de la carpeta temp y muestra la lista en un cuadro de alerta.

```
var folderURI = "file:///c|/temp";
var fileMask = "*.txt";
var list = FLfile.listFolder(folderURI + "/" + fileMask, "files");
if (list) {
  alert(folderURI + " contains: " + list.join(" "));
}
```

El ejemplo siguiente utiliza una máscara de archivo en la *folderURI* especificada para devolver los nombres de todos los archivos ejecutables de la carpeta de aplicación de Windows:

```
var executables = FLfile.listFolder("file:///C|/WINDOWS/*.exe","files")
alert(executables.join("\n"))
```

FLfile.read()

Disponibilidad

Flash MX 2004 7.2.

Uso

```
FFLfile.read()
```

Parámetros

fileOrFolderURI Una cadena, expresada como archivo:/// URI, que especifica el archivo o la carpeta cuyos atributos desea recuperar.

Valor devuelto

El contenido del archivo especificado como una cadena o `null` si se produce un error de lectura.

Descripción

Método; devuelve el contenido del archivo especificado como una cadena o `null` si se produce un error de lectura.

Ejemplos

El ejemplo siguiente lee el archivo mydata.txt y, si es correcto, muestra un cuadro de alerta con el contenido del archivo.

```
var fileURI = "file:///c|/temp/mydata.txt";
var str = FLfile.read( fileURI);
if (str) {
  alert( fileURL + " contains: " + str);
}
```

El ejemplo siguiente lee el código de ActionScript de un archivo de clase y lo almacena en la variable `code`:

```
var classFileURI = "file:///C|/MyApplication/TextCarousel.as";
var code = Flfile.read(classFileURI);
```

FLfile.remove()

Disponibilidad

Flash MX 2004 7.2.

Uso

```
FLfile.remove( fileOrFolderURI )
```

Parámetros

fileOrFolderURI Una cadena, expresada como archivo:/// URI, que especifica el archivo o la carpeta que desea eliminar (quitar).

Valor devuelto

Un valor booleano de `true` si es correcto, y de `false` en caso contrario.

Descripción

Método; elimina el archivo o la carpeta especificada. Si la carpeta contiene archivos, también se eliminarán esos archivos. No se pueden eliminar los archivos con el atributo R (de sólo lectura).

Ejemplos

El ejemplo siguiente advierte a un usuario si existe un archivo y, a continuación, lo elimina si el usuario lo desea.

```
var fileURI = prompt ("Enter file/folder to be deleted: ", "file:///c|/temp/
  delete.txt");
if (FLfile.exists(fileURI)) {
  var confirm = prompt("File exists. Delete it? (y/n)", "y");
  if (confirm == "y" || confirm == "Y") {
    if(FLfile.remove(fileURI)) {
      alert(fileURI + " is deleted.");
    }
    else {
      alert("fail to delete " + fileURI);
    }
  }
}
else {
  alert(fileURI + " does not exist");
}
```

El ejemplo siguiente elimina un archivo de configuración creado por una aplicación:

```
if(FLfile.remove("file:///C|/MyApplication/config.ini")) {
     alert("Configuration file deleted")
}
```

El ejemplo siguiente elimina la carpeta Configuration y su contenido:

```
FLfile.remove("file:///C|/MyApplication/Configuration/")
```

Véase también

FLfile.createFolder(), FLfile.getAttributes()

FLfile.setAttributes()

Disponibilidad

Flash MX 2004 7.2.

Uso

```
FLfile.setAttributes( fileURI, strAttrs )
```

Parámetros

fileURI Una cadena, expresada como archivo:/// URI, que especifica el archivo cuyos atributos desea definir.

strAttrs Una cadena que especifica valores para los atributos que desea definir. Para obtener valores aceptables de *strAttrs*, consulte la descripción siguiente.

Valor devuelto

Un valor booleano de `true` si es correcto.

NOTA	Los resultados son impredecibles si el archivo o carpeta no existen. Deberá utilizar FLfile.exists() antes de emplear este método.

Descripción

Método; especifica atributos de nivel del sistema para el archivo especificado.

Los valores siguientes son válidos para *strAttrs*:

- `N` — Sin atributos específicos (no es de sólo lectura, no está oculto, etc.)
- `A` — Listo para archivar (sólo Windows)
- `R` — De sólo lectura (en Macintosh, de sólo lectura significa “bloqueado”)
- `W` — De escritura (anula `R`)
- `H` — Oculto (sólo Windows)
- `V` — Visible (anula `H`, sólo Windows)

Si incluye `R` y `W` en *strAttrs*, se ignorará `R` y el archivo se definirá como de escritura. De forma similar, si transfiere `H` y `V`, se ignorará `H` y el archivo se definirá como visible.

Si desea asegurarse de que el atributo de archivo no está definido, utilice este comando con el parámetro `N` antes de establecer los atributos. Es decir, no hay contrapartida directa para `A` que desactive el atributo de archivo.

Ejemplos

El ejemplo siguiente define el archivo mydata.text como de sólo lectura y oculto. No tiene efecto en el atributo de archivo.

```
var URI = "file:///c|/temp/mydata.txt";
if (FLfile.exists(URI)) {
  FLfile.setAttributes(URI, "RH");
}
```

El ejemplo siguiente define el archivo mydata.text como de sólo lectura y oculto. También garantiza que no se define el atributo de archivo.

```
var URI = "file:///c|/temp/mydata.txt";

if (FLfile.exists(URI)) {
  FLfile.setAttributes(URI, "N");
  FLfile.setAttributes(URI, "RH");
}
```

Véase también

`FLfile.getAttributes()`

FLfile.write()

Disponibilidad

Flash MX 2004 7.2.

Uso

`FLfile.write( fileURI, textToWrite, [ , strAppendMode ] )`

Parámetros

`fileURI` Una cadena, expresada como archivo:/// URI, que especifica el archivo en el que desea escribir.

`textToWrite` Una cadena que representa el texto que desea situar en el archivo.

`strAppendMode` Una cadena opcional con el valor `"append"`, que especifica que desea añadir *`textToWrite`* al archivo existente. Si se omite, *`fileURI`* se sobrescribe con *`textToWrite`*.

Valor devuelto

Un valor booleano de `true` si es correcto, y de `false` en caso contrario.

Descripción

Método; escribe la cadena especificada en el archivo especificado (como UTF-8). Si no existe el archivo especificado se crea. Sin embargo, debe existir la carpeta en la que está ubicando el archivo antes de utilizar este método. Para crear carpetas, utilice `FLfile.createFolder()`.

Ejemplo

El ejemplo siguiente intenta escribir la cadena "xxx" en el archivo mydata.txt y muestra un mensaje de alerta si la escritura se realiza correctamente. A continuación, intenta añadir la cadena "aaa" al archivo y muestra un segundo mensaje de alerta si la escritura se realiza correctamente. Después de ejecutar este script, el archivo mydata.txt sólo contendrá el texto "xxxaaa".

```
var URI = "file:///c|/temp/mydata.txt";
if (FLfile.write(URI, "xxx")) {
  alert("Wrote xxx to " + URI);
}
if (FLfile.write(URI, "aaa", "append")) {
  alert("Appended aaa to " + fileURI);
}
```

Véase también

`FLfile.createFolder()`, `FLfile.exists()`

Objeto folderItem

Herencia Objeto Item > Objeto folderItem

Disponibilidad

Flash MX 2004.

Descripción

El objeto folderItem es una subclase del objeto Item. folderItem no tiene métodos ni propiedades exclusivos. Véase Objeto Item.

Objeto fontItem

Herencia Objeto Item > Objeto fontItem

Disponibilidad

Flash MX 2004.

Descripción

El objeto fontItem es una subclase del objeto Item. fontItem no tiene métodos ni propiedades exclusivos. Véase Objeto Item.

Objeto Frame

Disponibilidad

Flash MX 2004.

Descripción

El objeto Frame representa fotogramas en la capa.

Resumen de métodos del objeto Frame

Pueden emplearse los métodos siguientes con el objeto Frame.

Método	Descripción
`frame.getCustomEase()`	Devuelve una matriz de objetos JavaScript, cada uno de los cuales tiene una propiedad *x* e *y*.
`frame.setCustomEase()`	Especifica una curva cúbica de Bézier que se va a utilizar como curva de suavizado personalizada.

Resumen de propiedades del objeto Frame

Pueden emplearse las propiedades siguientes con el objeto Frame:

Propiedad	Descripción
`frame.actionScript`	Una cadena que representa código ActionScript.
`frame.duration`	De sólo lectura; un entero que representa el número de fotogramas en una secuencia de fotogramas.
`frame.elements`	De sólo lectura; una matriz de objetos Element (véase Objeto Element).
`frame.hasCustomEase`	Un valor booleano que especifica si el fotograma obtiene la información de suavizado de la curva de suavizado personalizada.
`frame.labelType`	Una cadena que especifica el tipo de nombre de fotograma.
`frame.motionTweenOrientToPath`	Un valor booleano que especifica si el elemento interpolado gira el elemento cuando se mueve a lo largo de una ruta para mantener su ángulo con respecto a cada punto de la ruta.

Propiedad	Descripción
`frame.motionTweenRotate`	Una cadena que especifica cómo gira el elemento interpolado.
`frame.motionTweenRotateTimes`	Un entero que especifica el número de veces que el elemento interpolado gira entre el fotograma clave inicial y el siguiente fotograma clave.
`frame.motionTweenScale`	Un valor booleano; especifica si el elemento interpolado se escala hasta el tamaño del objeto del siguiente fotograma clave, incrementando su tamaño con cada fotograma de la interpolación (`true`) o si no se escala (`false`).
`frame.motionTweenSnap`	Un valor booleano; especifica si el elemento interpolado se ajusta automáticamente al punto más próximo de la capa de guía de movimiento asociada a la capa de este fotograma (`true`) o si no se ajusta (`false`).
`frame.motionTweenSync`	Un valor booleano; si se define como `true`, sincroniza la animación del objeto interpolado con la línea de tiempo principal.
`frame.name`	Una cadena que especifica el nombre del fotograma.
`frame.shapeTweenBlend`	Una cadena que especifica cómo se mezcla una interpolación de forma entre la forma del fotograma clave al comienzo de la interpolación y la forma del siguiente fotograma clave.
`frame.soundEffect`	Una cadena que especifica efectos para un sonido que está asociado directamente a un fotograma (`frame.soundLibraryItem`).
`frame.soundLibraryItem`	Un elemento de biblioteca (véase Objeto SoundItem) empleado para crear un sonido.
`frame.soundLoop`	Un valor entero que especifica el número de veces que se reproduce un sonido asociado directamente a un fotograma (`frame.soundLibraryItem`).
`frame.soundLoopMode`	Una cadena que especifica si un sonido asociado directamente a un fotograma (`frame.soundLibraryItem`) debe reproducirse un número específico de veces o realizar un bucle indefinido.
`frame.soundName`	Una cadena que especifica el nombre de un sonido que está asociado directamente a un fotograma (`frame.soundLibraryItem`), tal como está almacenado en la biblioteca.

Propiedad	Descripción
`frame.soundSync`	Una cadena que especifica el comportamiento de sincronización de un sonido que está asociado directamente a un fotograma (`frame.soundLibraryItem`).
`frame.startFrame`	De sólo lectura; el índice del primer fotograma de una secuencia.
`frame.tweenEasing`	Un entero que especifica la cantidad de suavizado que debe aplicarse al objeto interpolado.
`frame.tweenType`	Una cadena que especifica el tipo de interpolado.
`frame.useSingleEaseCurve`	Un valor booleano que especifica si se utilizará una curva de suavizado personalizada para la información de suavizado de todas las propiedades.

frame.actionScript

Disponibilidad

Flash MX 2004.

Uso

```
frame.actionScript
```

Descripción

Propiedad; una cadena que representa código ActionScript. Para insertar una nueva línea de carácter, utilice `"\\n"`.

Ejemplo

El ejemplo siguiente asigna `stop()` a la acción primer fotograma de la capa superior:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].actionScript =
  'stop();';
```

frame.duration

Disponibilidad

Flash MX 2004.

Uso

```
frame.duration
```

Descripción

Propiedad de sólo lectura; un entero que representa el número de fotogramas en una secuencia de fotogramas.

Ejemplo

El ejemplo siguiente almacena el número de fotogramas de una secuencia de fotogramas que comienza en el primer fotograma de la capa superior en la variable `frameSpan`:

```
var frameSpan =
  fl.getDocumentDOM().getTimeline().layers[0].frames[0].duration;
```

frame.elements

Disponibilidad

Flash MX 2004.

Uso

```
frame.elements
```

Descripción

Propiedad de sólo lectura; una matriz de objetos Element (consulte Objeto Element). Los elementos aparecen en el orden en que están almacenados en el archivo FLA. Si hay múltiples formas en el escenario y todas están desagrupadas, Flash las tratará como un elemento. Si todas las formas están agrupadas, de modo que haya múltiples grupos en el escenario, Flash las considerará como elementos independientes. Es decir, Flash trata las formas no agrupadas y en bruto como un solo elemento, independientemente del número de formas independientes haya en el escenario. Si un fotograma se compone de tres formas no agrupadas y en bruto, por ejemplo, `elements.length` de ese fotograma devolverá un valor de 1. Seleccione cada forma individualmente y agrúpela para solucionar este problema.

Ejemplo

El ejemplo siguiente almacena una matriz de elementos actuales en la capa superior del primer fotograma en la variable `myElements`:

```
var myElements =
  fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements;
```

frame.getCustomEase()

Disponibilidad

Flash 8.

Uso

```
Frame.getCustomEase( [ property ] )
```

Parámetros

property Una cadena opcional que especifica la propiedad para la que desea devolver el valor de suavizado personalizado. Los valores aceptables son: `"all"`, `"position"`, `"rotation"`, `"scale"`, `"color"` y `"filters"`. El valor predeterminado es `"all"`.

Valor devuelto

Devuelve una matriz de objetos JavaScript, cada uno de los cuales tiene una propiedad *x* e *y*.

Descripción

Método; devuelve una matriz de objetos que representa los puntos de control de la curva cúbica de Béizier que define la curva de suavizado.

Ejemplo

El ejemplo siguiente devuelve el valor de suavizado personalizado de la propiedad `position` para el primer fotograma de la capa superior:

```
var theFrame = fl.getDocumentDOM().getTimeline().layers[0].frames[0]
var easeArray = theFrame.getCustomEase( "position" );
```

Véase también

`frame.hasCustomEase`, `frame.setCustomEase()`, `frame.useSingleEaseCurve`

frame.hasCustomEase

Disponibilidad

Flash 8.

Uso

```
frame.hasCustomEase
```

Descripción

Propiedad; un valor booleano. Si es `true`, el fotograma obtiene su información de suavizado de la curva de suavizado personalizada. Si es `false`, el fotograma obtiene la información de suavizado del valor de suavizado.

Ejemplo

El ejemplo siguiente especifica que el primer fotograma de la capa superior debe obtener su información de suavizado del valor de suavizado en lugar de tomarlo de la curva de suavizado personalizado:

```
var theFrame = fl.getDocumentDOM().getTimeline().layers[0].frames[0]
theFrame.hasCustomEase = false;
```

Véase también

`frame.getCustomEase()`, `frame.setCustomEase()`, `frame.useSingleEaseCurve`

frame.labelType

Disponibilidad

Flash MX 2004.

Uso

```
frame.labelType
```

Descripción

Propiedad; una cadena que especifica el tipo de nombre de fotograma. Los valores aceptables son: `"none"`, `"name"`, `"comment"` y `"anchor"`. Si define una etiqueta como `"none"` se borrará la propiedad `frame.name`.

Ejemplo

El ejemplo siguiente define el nombre del primer fotograma de la capa superior como `"First Frame"` y, a continuación, define su etiqueta como `"comment"`:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].name = 'First Frame';
fl.getDocumentDOM().getTimeline().layers[0].frames[0].labelType =
  'comment';
```

frame.motionTweenOrientToPath

Disponibilidad

Flash MX 2004.

Uso

```
frame.motionTweenOrientToPath
```

Descripción

Propiedad; un valor booleano; especifica si el elemento interpolado gira el elemento cuando se mueve a lo largo de una ruta para mantener su ángulo con respecto a cada punto de la ruta (`true`) o si no rota (`false`).

Si desea especificar un valor para esta propiedad, deberá definir `frame.motionTweenRotate` como `"none"`.

frame.motionTweenRotate

Disponibilidad

Flash MX 2004.

Uso

```
frame.motionTweenRotate
```

Descripción

Propiedad; una cadena que especifica cómo gira el elemento interpolado. Los valores aceptables son: `"none"`, `"auto"`, `"clockwise"` y `"counter-clockwise"`. Un valor de `"auto"` significa que el objeto girará en la dirección que requiera menos movimiento para coincidir con la rotación del objeto en el fotograma clave siguiente.

Si desea especificar un valor para `frame.motionTweenOrientToPath`, defina esta propiedad como `"none"`.

Ejemplo

Véase `frame.motionTweenRotateTimes`.

frame.motionTweenRotateTimes

Disponibilidad

Flash MX 2004.

Uso

```
frame.motionTweenRotateTimes
```

Descripción

Propiedad; un entero que especifica el número de veces que el elemento interpolado gira entre el fotograma clave inicial y el siguiente fotograma clave.

Ejemplo

El ejemplo siguiente gira el elemento de este fotograma en el sentido contrario al de las agujas del reloj tres veces cuando llega al siguiente fotograma clave:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].motionTweenRotate =
  "counter-clockwise";
fl.getDocumentDOM().getTimeline().layers[0].frames[0].motionTweenRotateTime
  s = 3;
```

frame.motionTweenScale

Disponibilidad

Flash MX 2004.

Uso

```
frame.motionTweenScale
```

Descripción

Propiedad; un valor booleano; especifica si el elemento interpolado se escala hasta el tamaño del objeto del siguiente fotograma clave, incrementando su tamaño con cada fotograma de la interpolación (`true`) o si no se escala (`false`).

Ejemplo

El siguiente ejemplo especifica que el elemento interpolado se debe escalar al tamaño del objeto del fotograma clave siguiente, aumentando el tamaño con cada fotograma de la interpolación.

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].motionTweenScale =
  true;
```

frame.motionTweenSnap

Disponibilidad

Flash MX 2004.

Uso

```
frame.motionTweenSnap
```

Descripción

Propiedad; un valor booleano; especifica si el elemento interpolado se ajusta automáticamente al punto más próximo de la capa de guía de movimiento asociada a la capa de este fotograma (`true`) o si no se ajusta (`false`).

frame.motionTweenSync

Disponibilidad

Flash MX 2004.

Uso

```
frame.motionTweenSync
```

Descripción

Propiedad; un valor booleano; si se define como `true`, sincroniza la animación del objeto interpolado con la línea de tiempo principal.

Ejemplo

El ejemplo siguiente especifica que el objeto interpolado debe sincronizarse con la línea de tiempo:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].motionTweenSync =
  true;
```

frame.name

Disponibilidad

Flash MX 2004.

Uso

```
frame.name
```

Descripción

Propiedad; una cadena que especifica el nombre del fotograma.

Ejemplo

El ejemplo siguiente define el nombre del primer fotograma de la capa superior como `"First Frame"` y, a continuación, almacena el valor `name` en la variable `frameLabel`:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].name = 'First Frame';
var frameLabel =
  fl.getDocumentDOM().getTimeline().layers[0].frames[0].name;
```

frame.setCustomEase()

Disponibilidad

Flash 8.

Uso

```
frame.setCustomEase( property, easeCurve )
```

Parámetros

property Una cadena que especifica para qué propiedad debe emplearse la curva de suavizado. Los valores aceptables son: `"all"`, `"position"`, `"rotation"`, `"scale"`, `"color"` y `"filters"`.

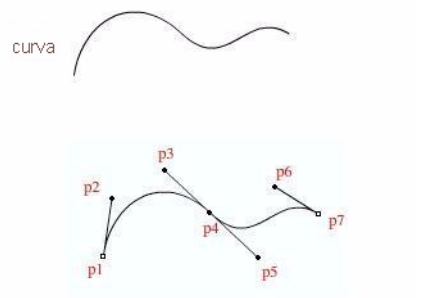
easeCurve Una matriz de objetos que define la curva de suavizado. Cada elemento de la matriz debe ser un objeto JavaScript con propiedades *x* e *y*.

Valor devuelto

Ninguno.

Descripción

Método; especifica una matriz de coordenadas de puntos de control y puntos finales tangentes que describen una curva cúbica Bézier que se utilizará como curva de suavizado personalizada. Esta matriz se crea con la posición horizontal (ordinal: de izquierda a derecha) de los puntos de control y los puntos finales tangentes. Por ejemplo, en la siguiente ilustración se muestra una curva de suavizado que se crearía si la matriz *easeCurve* incluyera valores para los siete puntos mostrados del *p1* al *p7*:



Ejemplo

El ejemplo siguiente establece la curva de suavizado para todas las propiedades del primer fotograma de la primera capa en la curva Bézier especificada por los puntos de control y los puntos finales tangentes almacenados en la matriz `myCurve`:

```
var theFrame = fl.getDocumentDOM().getTimeline().layers[0].frames[0]
var myCurve = [{x:100, y:200},{x:200, y:100}, {x:10, y:0}]
theFrame.setCustomEase("all", myCurve);
```

Véase también

`frame.getCustomEase()`, `frame.hasCustomEase`, `frame.useSingleEaseCurve`

frame.shapeTweenBlend

Disponibilidad

Flash MX 2004.

Uso

```
frame.shapeTweenBlend
```

Descripción

Propiedad; una cadena que especifica cómo se mezcla una interpolación de forma entre la forma del fotograma clave al comienzo de la interpolación y la forma del siguiente fotograma clave. Los valores aceptables son: `"distributive"` y `"angular"`.

frame.soundEffect

Disponibilidad

Flash MX 2004.

Uso

```
frame.soundEffect
```

Descripción

Propiedad; una cadena que especifica efectos para un sonido que está asociado directamente a un fotograma (`frame.soundLibraryItem`). Los valores aceptables son: `"none"`, `"left channel"`, `"right channel"`, `"fade left to right"`, `"fade right to left"`, `"fade in"`, `"fade out"` y `"custom"`.

Ejemplo

El ejemplo siguiente especifica que el sonido asociado al primer fotograma debe desaparecer de forma paulatina:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].soundEffect = "fade
  in";
```

frame.soundLibraryItem

Disponibilidad

Flash MX 2004.

Uso

```
frame.soundLibraryItem
```

Descripción

Propiedad; un elemento de biblioteca (consulte Objeto SoundItem) empleado para crear un sonido. El sonido se asocia directamente al fotograma.

Ejemplo

El siguiente ejemplo asigna el primer elemento de la biblioteca a la propiedad `soundLibraryItem` del primer fotograma:

```
// El primer elemento de la biblioteca debe ser un objeto de sonido
fl.getDocumentDOM().getTimeline().layers[0].frames[0].soundLibraryItem
  =fl.getDocumentDOM().library.items[0];
```

frame.soundLoop

Disponibilidad

Flash MX 2004.

Uso

```
frame.soundLoop
```

Descripción

Propiedad; un valor entero que especifica el número de veces que se reproduce un sonido asociado directamente a un fotograma (`frame.soundLibraryItem`). Si desea especificar un valor para esta propiedad, defina `frame.soundLoopMode` como `"repeat"`.

Ejemplo

Véase `frame.soundLoopMode`.

frame.soundLoopMode

Disponibilidad

Flash MX 2004.

Uso

```
frame.soundLoopMode
```

Descripción

Propiedad; una cadena que especifica si un sonido asociado directamente a un fotograma (`frame.soundLibraryItem`) debe reproducirse un número específico de veces o realizar un bucle indefinido. Los valores aceptables son: `"repeat"` y `"loop"`. Para especificar el número de veces que el sonido debe reproducirse, defina un valor para `frame.soundLoop`.

Ejemplo

El ejemplo siguiente especifica que un sonido debe reproducirse dos veces:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].soundLoopMode =
  "repeat";
fl.getDocumentDOM().getTimeline().layers[0].frames[0].soundLoop = 2;
```

frame.soundName

Disponibilidad

Flash MX 2004.

Uso

```
frame.soundName
```

Descripción

Propiedad; una cadena que especifica el nombre de un sonido que está asociado directamente a un fotograma (`frame.soundLibraryItem`), tal como está almacenado en la biblioteca.

Ejemplo

El ejemplo siguiente cambia la propiedad `soundName` del primer fotograma a `"song1.mp3"`; song1.mp3 debe estar presente en la biblioteca:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].soundName =
  "song1.mp3";
```

frame.soundSync

Disponibilidad

Flash MX 2004.

Uso

```
frame.soundSync
```

Descripción

Propiedad; una cadena que especifica el comportamiento de sincronización de un sonido que está asociado directamente a un fotograma (`frame.soundLibraryItem`). Los valores aceptables son: `"event"`, `"stop"`, `"start"` y `"stream"`.

Ejemplo

El ejemplo siguiente especifica que un sonido debe reproducirse sin interrupción:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].soundSync = 'stream';
```

frame.startFrame

Disponibilidad

Flash MX 2004.

Uso

```
frame.startFrame
```

Descripción

Propiedad de sólo lectura; el índice del primer fotograma de una secuencia.

Ejemplo

En el ejemplo siguiente, `stFrame` es el índice del primer fotograma de la secuencia de fotogramas. En este ejemplo, una secuencia de fotogramas abarca los seis fotogramas desde el Fotograma 5 hasta el Fotograma 10. Por tanto, el valor de `stFrame` en cualquier fotograma entre Fotograma 5 y Fotograma 10 es 4 (recuerde que los valores de índice son distintos de los valores de número de fotograma).

```
var stFrame =
  fl.getDocumentDOM().getTimeline().layers[0].frames[4].startFrame;
fl.trace(stFrame); // 4
var stFrame =
  fl.getDocumentDOM().getTimeline().layers[0].frames[9].startFrame;
fl.trace(stFrame); // 4
```

frame.tweenEasing

Disponibilidad

Flash MX 2004.

Uso

```
frame.tweenEasing
```

Descripción

Propiedad; un entero que especifica la cantidad de suavizado que debe aplicarse al objeto interpolado. Los valores aceptables van de -100 a 100. Para comenzar la interpolación de movimiento lentamente y acelerar hacia el final de la animación, utilice un valor entre -1 y -100. Para comenzar la interpolación de movimiento rápidamente y desacelerar hacia el final de la animación, utilice un valor positivo entre 1 y 100.

Ejemplo

El ejemplo siguiente especifica que el movimiento del objeto interpolado debe comenzar con bastante rapidez y desacelerarse hacia el final de la animación:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].tweenEasing = 50;
```

frame.tweenType

Disponibilidad

Flash MX 2004.

Uso

```
frame.tweenType
```

Descripción

Propiedad; una cadena que especifica el tipo de interpolación; los valores aceptables son: `"motion"`, `"shape"` y `"none"`. El valor `"none"` elimina la interpolación de movimiento. Utilice el método `timeline.createMotionTween()` para crear una interpolación.

Si especifica `"motion"`, el objeto del fotograma deberá ser un símbolo, un campo de texto o un objeto agrupado. Se interpolará desde su ubicación en el fotograma clave actual hasta la ubicación del siguiente fotograma clave.

Si especifica `"shape"`, el objeto del fotograma deberá ser un objeto. Se mezclará desde su forma en el fotograma clave actual hasta la forma del siguiente fotograma clave.

Ejemplo

El ejemplo siguiente especifica que el objeto tiene una interpolación de movimiento y, por tanto, debe interpolarse desde su ubicación en el fotograma clave actual hasta la ubicación del siguiente fotograma clave:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].tweenType = "motion";
```

frame.useSingleEaseCurve

Disponibilidad

Flash 8.

Uso

```
frame.useSingleEaseCurve
```

Descripción

Propiedad; un valor booleano. Si es `true`, se utilizará una curva de suavizado personalizada para la información de suavizado de todas las propiedades. Si es `false`, cada propiedad tendrá su propia curva.

Esta propiedad se ignora si no se ha aplicado al fotograma el suavizado personalizado.

Ejemplo

El ejemplo siguiente especifica que una curva de suavizado personalizada debe emplearse para todas las propiedades del primer fotograma de la primera capa:

```
var theFrame = fl.getDocumentDOM().getTimeline().layers[0].frames[0]
theFrame.useSingleEaseCurve = true;
```

Véase también

`frame.getCustomEase()`, `frame.hasCustomEase`, `frame.setCustomEase()`

Objeto HalfEdge

Disponibilidad

Flash MX 2004.

Descripción

El objeto HalfEdge es el lado dirigido del borde de un Objeto Shape. Un borde tiene dos lados dirigidos. Puede atravesar los contornos de una forma rodeando estos lados dirigidos. Por ejemplo, comenzando desde un lado dirigido, puede llevar un seguimiento de todos los lados dirigidos alrededor de un contorno de una forma y volver al lado dirigido original.

Los lados dirigidos son ordenados. Un lado dirigido representa un lado del borde; el otro lado dirigido representa el otro lado.

Resumen de métodos del objeto HalfEdge

Los métodos siguientes están disponibles para el objeto HalfEdge:

Método	Descripción
`halfEdge.getEdge()`	Obtiene el Objeto Edge para el objeto HalfEdge.
`halfEdge.getNext()`	Obtiene el siguiente lado dirigido del contorno actual.
`halfEdge.getOppositeHalfEdge()`	Obtiene el objeto HalfEdge del otro lado del borde.
`halfEdge.getPrev()`	Obtiene el objeto HalfEdge anterior del contorno actual.
`halfEdge.getVertex()`	Obtiene el Objeto Vertex en la cabeza del objeto HalfEdge.

Resumen de propiedades del objeto HalfEdge

Las propiedades siguientes están disponibles para el objeto HalfEdge:

Propiedad	Descripción
`halfEdge.id`	De sólo lectura; un identificador único de entero para el objeto HalfEdge.
`halfEdge.index`	

halfEdge.getEdge()

Disponibilidad

Flash MX 2004.

Uso

```
halfEdge.getEdge()
```

Parámetros

Ninguno.

Valor devuelto

Un Objeto Edge.

Descripción

Método; obtiene el objeto Edge para el objeto HalfEdge. Véase Objeto Edge.

Ejemplo

El ejemplo siguiente ilustra la obtención de un borde y de un lado dirigido para la forma especificada.

```
var shape = fl.getDocumentDOM().selection[0];
var hEdge = shape.edges[0].getHalfEdge(0);
var edge = hEdge.getEdge();
```

halfEdge.getNext()

Disponibilidad

Flash MX 2004.

Uso

```
halfEdge.getNext()
```

Parámetros

Ninguno.

Valor devuelto

Un objeto HalfEdge.

Descripción

Método; obtiene el siguiente lado dirigido del contorno actual.

NOTA	Los lados dirigidos tienen una dirección y un orden de secuencia, a diferencia de los bordes.

Ejemplo

El ejemplo siguiente almacena el siguiente lado dirigido del contorno especificado en la variable `nextHalfEdge`:

```
var shape = fl.getDocumentDOM().selection[0];
var hEdge = shape.edges[0].getHalfEdge( 0 );
var nextHalfEdge = hEdge.getNext();
```

halfEdge.getOppositeHalfEdge()

Disponibilidad

Flash MX 2004.

Uso

```
halfEdge.getOppositeHalfEdge()
```

Parámetros

Ninguno.

Valor devuelto

Un objeto HalfEdge.

Descripción

Método; obtiene el objeto HalfEdge del otro lado del borde.

Ejemplo

El ejemplo siguiente almacena el lado dirigido situado frente a `hEdge` en la variable `otherHalfEdge`:

```
var shape = fl.getDocumentDOM().selection[0];
var hEdge = shape.edges[0].getHalfEdge(0);
var otherHalfEdge = hEdge.getOppositeHalfEdge();
```

halfEdge.getPrev()

Disponibilidad

Flash MX 2004.

Uso

```
halfEdge.getPrev()
```

Parámetros

Ninguno.

Valor devuelto

Un objeto HalfEdge.

Descripción

Método; obtiene el objeto HalfEdge anterior del contorno actual.

NOTA	Los lados dirigidos tienen una dirección y un orden de secuencia, a diferencia de los bordes.

Ejemplo

El ejemplo siguiente almacena el lado dirigido anterior del contorno especificado en la variable `prevHalfEdge`:

```
var shape = fl.getDocumentDOM().selection[0];
var hEdge = shape.edges[0].getHalfEdge( 0 );
var prevHalfEdge = hEdge.getPrev();
```

halfEdge.getVertex()

Disponibilidad

Flash MX 2004.

Uso

```
halfEdge.getVertex()
```

Parámetros

Ninguno.

Valor devuelto

Un Objeto Vertex.

Descripción

Método; obtiene el objeto Vertex en la cabeza del objeto HalfEdge. Véase Objeto Vertex.

Ejemplo

El ejemplo siguiente almacena el objeto Vertex en la cabeza de `hEdge` en la variable `vertex`:

```
var shape = fl.getDocumentDOM().selection[0];
var edge = shape.edges[0];
var hEdge = edge.getHalfEdge(0);
var vertex = hEdge.getVertex();
```

halfEdge.id

Disponibilidad

Flash MX 2004.

Uso

```
halfEdge.id
```

Descripción

Propiedad de sólo lectura; un identificador único de entero para el objeto HalfEdge.

Ejemplo

El ejemplo siguiente muestra un identificador único para el lado dirigido especificado en el panel Salida:

```
var shape = fl.getDocumentDOM().selection[0];
alert(shape.contours[0].getHalfEdge().id);
```

halfEdge.index

Disponibilidad

Flash MX 2004.

Uso

```
halfEdge.index
```

Descripción

Propiedad de sólo lectura; un entero con un valor de 0 ó 1 que especifica el índice para este objeto HalfEdge en el borde principal.

Ejemplo

El ejemplo siguiente muestra el valor de índice para el lado dirigido especificado en el panel Salida:

```
var shape = fl.getDocumentDOM().selection[0];
var hEdge = shape.edges[0].getHalfEdge(0);
var heIndex = hEdge.index;
```

Objeto Instance

Herencia Objeto Element > Objeto Instance

Disponibilidad

Flash MX 2004.

Descripción

Instance es una subclase del Objeto Element.

Resumen de propiedades del objeto Instance

Además de todas las propiedades del Objeto Element, Instance tiene las siguientes propiedades:

Propiedad	Descripción
`instance.instanceType`	De sólo lectura; una cadena que representa el tipo de instancia.
`instance.libraryItem`	Elemento de biblioteca empleado para crear una instancia de esta instancia.

instance.instanceType

Disponibilidad

Flash MX 2004; el valor aceptable de `"video"` añadido en Flash 8.

Uso

```
instance.instanceType
```

Descripción

Propiedad de sólo lectura; una cadena que representa el tipo de instancia. Los valores aceptables son: `"symbol"`, `"bitmap"`, `"embedded video"`, `"linked video"`, `"video"` y `"compiled clip"`.

NOTA

En Flash MX 2004, el valor de `instance.instanceType` para un elemento añadido a la biblioteca mediante `library.addNewItem("video")` es `"embedded_video"`. En Flash 8, el valor es `"video"`. Véase `library.addNewItem()`.

Ejemplo

El ejemplo siguiente muestra que el tipo de instancia de un clip de película es "symbol":

```
// Selecciona un clip de película y, a continuación, ejecuta este script.
var type = fl.getDocumentDOM().selection[0].instanceType;
fl.trace("This instance type is " + type);
```

instance.libraryItem

Disponibilidad

Flash MX 2004.

Uso

```
instance.libraryItem
```

Descripción

Propiedad; un elemento de biblioteca empleado para crear una instancia de esta instancia. Sólo puede cambiar esta propiedad a otro elemento de biblioteca del mismo tipo (es decir, no puede definir una instancia `symbol` para hacer referencia a un mapa de bits). Véase Objeto library.

Ejemplo

El ejemplo siguiente cambia el símbolo seleccionado para hacer referencia al primer elemento de la biblioteca:

```
fl.getDocumentDOM().selection[0].libraryItem =
  fl.getDocumentDOM().library.items[0];
```

Objeto Item

Disponibilidad

Flash MX 2004.

Descripción

El objeto Item es una clase base abstracta. Todos los contenidos de la biblioteca se derivan de Item. Véase también Objeto library.

Resumen de métodos del objeto Item

Los métodos siguientes están disponibles para el objeto Item.

Método	Descripción
`item.addData()`	Añade datos especificados a un elemento de biblioteca.
`item.getData()`	Recupera el valor de los datos especificados.
`item.hasData()`	Determina si el elemento de biblioteca tiene los datos con nombre.
`item.removeData()`	Elimina datos persistentes del elemento de biblioteca.

Resumen de propiedades del objeto Item

Las propiedades siguientes están disponibles para el objeto Item.

Propiedad	Descripción
`item.itemType`	De sólo lectura; una cadena que especifica el tipo de elemento.
`item.linkageClassName`	Una cadena que especifica la clase de ActionScript 2.0 que se asociará al símbolo.
`item.linkageExportForAS`	Valor booleano. Si es `true`, el elemento se exportará para ActionScript.
`item.linkageExportForRS`	Valor booleano. Si es `true`, el elemento se exportará para compartir tiempo de ejecución.
`item.linkageExportInFirstFrame`	Valor booleano. Si es `true`, el elemento se exportará en el primer fotograma.
`item.linkageIdentifier`	Una cadena que especifica el nombre que utilizará Flash para identificar el activo cuando cree un vínculo con el archivo SWF de destino.

Propiedad	Descripción
item.linkageImportForRS	Valor booleano. Si es true, el elemento se importará para compartir en tiempo de ejecución.
item.linkageURL	Una cadena que especifica la URL donde se encuentra el archivo SWF que contiene el activo compartido.
item.name	Una cadena que especifica el nombre del elemento de biblioteca, que incluye la estructura de carpetas.

item.addData()

Disponibilidad

Flash MX 2004.

Uso

```
item.addData( name, type, data )
```

Parámetros

name Una cadena que especifica el nombre de los datos.

type Una cadena que especifica el tipo de datos. Los tipos válidos son: "integer", "integerArray", "double", "doubleArray", "string" y "byteArray".

data Los datos que se van a añadir al elemento de biblioteca especificado. El tipo de datos depende del valor del parámetro type. Por ejemplo si type es "integer", el valor de los datos deberá ser un entero, etc.

Valor devuelto

Ninguno.

Descripción

Método; añade datos especificados a un elemento de biblioteca.

Ejemplo

El ejemplo siguiente añade datos con el nombre myData con un valor entero de 12 al primer elemento de la biblioteca:

```
fl.getDocumentDOM().library.items[0].addData("myData", "integer", 12);
```

item.getData()

Disponibilidad

Flash MX 2004.

Uso

```
item.getData( name )
```

Parámetros

name Una cadena que especifica el nombre de los datos que se van a recuperar.

Valor devuelto

Los datos especificados por el parámetro *name*. El tipo de datos devueltos depende del tipo de datos almacenados.

Descripción

Método; recupera el valor de los datos especificados.

Ejemplo

El ejemplo siguiente obtiene el valor de los datos con nombre myData del primer elemento de la biblioteca y lo almacena en la variable `libData`.

```
var libData = fl.getDocumentDOM().library.items[0].getData( "myData" );
```

item.hasData()

Disponibilidad

Flash MX 2004.

Uso

```
item.hasData( name )
```

Parámetros

name Una cadena que especifica el nombre de los datos que se van a comprobar en el elemento de biblioteca.

Valor devuelto

Un valor booleano: `true` si existen los datos especificados; `false` en caso contrario.

Descripción

Método; determina si el elemento de biblioteca tiene los datos con nombre.

Ejemplo

El ejemplo siguiente muestra un mensaje en el panel Salida si el primer elemento de la biblioteca contiene un punto de datos con el nombre myData:

```
if ( fl.getDocumentDOM().library.items[0].hasData( "myData" ) ){
  fl.trace("Yep, it's there!");
}
```

item.itemType

Disponibilidad

Flash MX 2004.

Uso

```
item.itemType
```

Descripción

Propiedad de sólo lectura; una cadena que especifica el tipo de elemento. El valor es uno de los siguientes: `"undefined"`, `"component"`, `"movie clip"`, `"graphic"`, `"button"`, `"folder"`, `"font"`, `"sound"`, `"bitmap"`, `"compiled clip"`, `"screen"` y `"video"`. Si esta propiedad es `"video"`, puede determinar el tipo de vídeo; consulte `videoItem.videoType`.

Ejemplo

El ejemplo siguiente muestra el tipo de la biblioteca especificada en el panel Salida:

```
fl.trace(fl.getDocumentDOM().library.items[0].itemType);
```

item.linkageClassName

Disponibilidad

Flash MX 2004.

Uso

```
item.linkageClassName
```

Descripción

Propiedad; una cadena que especifica la clase de ActionScript 2.0 que se asociará al símbolo. Para que se defina esta propiedad, las propiedades `item.linkageExportForAS` y/o `item.linkageExportForRS` deben definirse como `true` y la propiedad `item.linkageImportForRS` debe definirse como `false`.

Ejemplo

El ejemplo siguiente especifica que el nombre de clase de ActionScript 2.0 asociado al primer elemento de la biblioteca es myClass:

```
fl.getDocumentDOM().library.items[0].linkageClassName = "myClass";
```

item.linkageExportForAS

Disponibilidad

Flash MX 2004.

Uso

```
item.linkageExportForAS
```

Descripción

Propiedad; valor booleano. Si esta propiedad es `true`, el elemento se exportará para ActionScript. También puede definir las propiedades `item.linkageExportForRS` y `item.linkageExportInFirstFrame` como `true`.

Si define esta propiedad como true, la propiedad `item.linkageImportForRS` debe definirse como `false`. Asimismo, deberá especificar un identificador (`item.linkageIdentifier`) y una URL (`item.linkageURL`).

Ejemplo

El ejemplo siguiente define esta propiedad para el elemento de biblioteca especificado:

```
fl.getDocumentDOM().library.items[0].linkageExportForAS = true;
```

item.linkageExportForRS

Disponibilidad

Flash MX 2004.

Uso

```
item.linkageExportForRS
```

Descripción

Propiedad; valor booleano. Si esta propiedad es `true`, el elemento se exportará para compartir en tiempo de ejecución. También puede definir las propiedades `item.linkageExportForAS` y `item.linkageExportInFirstFrame` como `true`.

Si define esta propiedad como true, la propiedad `item.linkageImportForRS` debe definirse como `false`. Asimismo, deberá especificar un identificador (`item.linkageIdentifier`) y una URL (`item.linkageURL`).

Ejemplo

El ejemplo siguiente define esta propiedad para el elemento de biblioteca especificado:

```
fl.getDocumentDOM().library.items[0].linkageExportForRS = true;
```

item.linkageExportInFirstFrame

Disponibilidad

Flash MX 2004.

Uso

```
item.linkageExportInFirstFrame
```

Descripción

Propiedad; un valor booleano. Si es `true`, el elemento se exportará en el primer fotograma; si es `false`, el elemento se exportará en el fotograma de la primera instancia. Si el elemento no aparece en el escenario, no se exportará.

El valor de esta propiedad sólo se puede definir como `true` cuando `item.linkageExportForAS` y/o `item.linkageExportForRS` se definen como `true`.

Ejemplo

El ejemplo siguiente especifica que el elemento de biblioteca especificado se exporta en el primer fotograma:

```
fl.getDocumentDOM().library.items[0].linkageExportInFirstFrame = true;
```

item.linkageIdentifier

Disponibilidad

Flash MX 2004.

Uso

```
item.linkageIdentifier
```

Descripción

Propiedad; una cadena que especifica el nombre que utilizará Flash para identificar el activo cuando cree un vínculo con el archivo SWF de destino. Flash ignora esta propiedad si `item.linkageImportForRS`, `item.linkageExportForAS` y `item.linkageExportForRS` se definen como `false`. De manera inversa, se debe definir esta propiedad cuando cualquiera de dichas propiedades estén definidas como `true`.

Ejemplo

El ejemplo siguiente especifica que la cadena `my_mc` se utilizará para identificar el elemento de biblioteca cuando se vincule al archivo SWF de destino al que se está exportando:

```
fl.getDocumentDOM().library.items[0].linkageIdentifier = "my_mc";
```

Véase también

`item.linkageURL`

item.linkageImportForRS

Disponibilidad

Flash MX 2004.

Uso

```
item.linkageImportForRS
```

Descripción

Propiedad; un valor booleano: si es `true`, el elemento se importará para compartir en tiempo de ejecución. Si esta propiedad se define como `true`, tanto `item.linkageExportForAS` como `item.linkageExportForRS` deberán definirse como `false`. Asimismo, deberá especificar un identificador (`item.linkageIdentifier`) y una URL (`item.linkageURL`).

Ejemplo

El ejemplo siguiente define esta propiedad `true` para el elemento de biblioteca especificado:

```
fl.getDocumentDOM().library.items[0].linkageImportForRS = true;
```

item.linkageURL

Disponibilidad

Flash MX 2004.

Uso

```
item.linkageURL
```

Descripción

Propiedad; una cadena que especifica la URL donde se encuentra el archivo SWF que contiene el activo compartido. Flash ignora esta propiedad si `item.linkageImportForRS`, `item.linkageExportForAS` y `item.linkageExportForRS` se definen como `false`. De manera inversa, se debe definir esta propiedad cuando cualquiera de dichas propiedades estén definidas como `true`. Puede especificar una URL Web o un nombre de archivo con formato dependiente de la plataforma (es decir, barras diagonales [/] o barras invertidas [\], según la plataforma).

Ejemplo

El ejemplo siguiente especifica una URL de vinculación para el elemento de biblioteca especificado:

```
fl.getDocumentDOM().library.items[0].linkageURL = "theShareSWF.swf";
```

Véase también

`item.linkageIdentifier`

item.name

Disponibilidad

Flash MX 2004.

Uso

```
item.name
```

Descripción

Método; una cadena que especifica el nombre del elemento de biblioteca, que incluye la estructura de carpetas. Por ejemplo, si Symbol_1 se encuentra en una carpeta llamada Folder_1, la propiedad `name` de Symbol_1 será`"Folder_1/Symbol_1"`.

Ejemplo

El ejemplo siguiente muestra el nombre de la biblioteca especificada en el panel Salida:

```
fl.trace(fl.getDocumentDOM().library.items[0].name);
```

item.removeData()

Disponibilidad

Flash MX 2004.

Uso

```
item.removeData( name )
```

Parámetros

name Especifica el nombre de los datos que se van a eliminar del elemento de biblioteca.

Valor devuelto

Ninguno.

Descripción

Propiedad; elimina datos persistentes del elemento de biblioteca.

Ejemplo

El ejemplo siguiente elimina los datos con el nombre myData del primer elemento de la biblioteca:

```
fl.getDocumentDOM().library.items[0].removeData( "myData" );
```

Objeto Layer

Disponibilidad

Flash MX 2004.

Descripción

El objeto Layer representa una capa en la línea de tiempo. La propiedad `timeline.layers` contiene una matriz de objetos Layer a los que puede acceder `fl.getDocumentDOM().getTimeline().layers`.

Resumen de propiedades del objeto Layer

Las propiedades siguientes están disponibles para el objeto Layer:

Propiedad	Descripción
`layer.color`	Una cadena, valor hexadecimal o entero que especifica el color asignado al contorno de la capa.
`layer.frameCount`	De sólo lectura; un entero que especifica el número de fotogramas de la capa.
`layer.frames`	De sólo lectura; una matriz de objetos Frame.
`layer.height`	Un entero que especifica la altura de la capa en porcentaje; equivale al valor de valor de altura de Capa del cuadro de diálogo Propiedades de Capa.
`layer.layerType`	Una cadena que especifica el uso actual de la capa; equivale a la opción Tipo del cuadro de diálogo Propiedades de Capa.
`layer.locked`	Un valor booleano que especifica el estado bloqueado de la capa.
`layer.name`	Una cadena que especifica el nombre de la capa.
`layer.outline`	Un valor booleano que especifica el estado de los contornos para todos los objetos de la capa.
`layer.parentLayer`	Un objeto Layer que representa la carpeta o capa de guía o enmascaramiento que contiene la capa.
`layer.visible`	Un valor booleano que especifica si se muestran u ocultan los objetos de la capa en el escenario.

layer.color

Disponibilidad

Flash MX 2004.

Uso

```
layer.color
```

Descripción

Propiedad; el color asignado al contorno de la capa, en uno de los formatos siguientes:

- Una cadena con el formato `"#RRGGBB"` o `"#RRGGBBAA"`
- Un número hexadecimal con el formato `0xRRGGBB`
- Un entero que representa el equivalente decimal del número hexadecimal

Esta propiedad equivale a la opción Color de contorno del cuadro de diálogo Propiedades de Capa.

Ejemplo

El ejemplo siguiente almacena el valor de la primera capa en la variable `colorValue`:

```
var colorValue = fl.getDocumentDOM().getTimeline().layers[0].color;
```

El ejemplo siguiente muestra tres formas de definir el color de la primera capa como rojo:

```
fl.getDocumentDOM().getTimeline().layers[0].color=16711680;
fl.getDocumentDOM().getTimeline().layers[0].color="#ff0000";
fl.getDocumentDOM().getTimeline().layers[0].color=0xFF0000;
```

layer.frameCount

Disponibilidad

Flash MX 2004.

Uso

```
layer.frameCount
```

Descripción

Propiedad de sólo lectura; un entero que especifica el número de fotogramas de la capa.

Ejemplo

El ejemplo siguiente almacena el número de fotogramas de la primera capa en la variable `fcNum`:

```
var fcNum = fl.getDocumentDOM().getTimeline().layers[0].frameCount;
```

layer.frames

Disponibilidad

Flash MX 2004.

Uso

```
layer.frames
```

Descripción

Propiedad de sólo lectura; una matriz de objetos Frame (véase Objeto Frame).

Ejemplo

El ejemplo siguiente define la variable `frameArray` como la matriz de objetos Frame para los fotogramas del documento actual:

```
var frameArray = fl.getDocumentDOM().getTimeline().layers[0].frames;
```

Para determinar si un fotograma es un fotograma clave, compruebe si la propiedad `frame.startFrame` coincide con el índice de la matriz, como se muestra en el ejemplo siguiente:

```
var frameArray = fl.getDocumentDOM().getTimeline().layers[0].frames;
var n = frameArray.length;
for (i=0; i<n; i++) {
  if (i==frameArray[i].startFrame) {
    alert("Keyframe at: " + i);
  }
}
```

layer.height

Disponibilidad

Flash MX 2004.

Uso

```
layer.height
```

Descripción

Propiedad; un entero que especifica la altura de la capa en porcentaje; equivale al valor de valor de altura de Capa del cuadro de diálogo Propiedades de Capa. Los valores válidos representan porcentajes de la altura predeterminada: 100, 200 o 300.

Ejemplo

El ejemplo siguiente almacena el valor de porcentaje de la altura de la primera capa:

```
var layerHeight = fl.getDocumentDOM().getTimeline().layers[0].height;
```

El ejemplo siguiente define la altura de la primera capa en 300%:

```
fl.getDocumentDOM().getTimeline().layers[0].height = 300;
```

layer.layerType

Disponibilidad

Flash MX 2004.

Uso

```
layer.layerType
```

Descripción

Propiedad; una cadena que especifica el uso actual de la capa; equivale a la opción Tipo del cuadro de diálogo Propiedades de Capa. Los valores aceptables son: `"normal"`, `"guide"`, `"guided"`, `"mask"`, `"masked"` y `"folder"`.

Ejemplo

El ejemplo siguiente define la primera capa de la línea de tiempo con el tipo `"folder"`:

```
fl.getDocumentDOM().getTimeline().layers[0].layerType = "folder";
```

layer.locked

Disponibilidad

Flash MX 2004.

Uso

```
layer.locked
```

Descripción

Propiedad; un valor booleano que especifica el estado bloqueado de la capa. Si se define como `true`, la capa se bloqueará. El valor predeterminado es `false`.

Ejemplo

El ejemplo siguiente almacena el valor booleano del estado de la primera capa en la variable `lockStatus`:

```
var lockStatus = fl.getDocumentDOM().getTimeline().layers[0].locked;
```

El ejemplo siguiente define el estado de la primera capa como desbloqueado:

```
fl.getDocumentDOM().getTimeline().layers[0].locked = false;
```

layer.name

Disponibilidad

Flash MX 2004.

Uso

```
layer.name
```

Descripción

Propiedad; una cadena que especifica el nombre de la capa.

Ejemplo

El ejemplo siguiente define el nombre de la primera capa del documento actual como "primer plano":

```
fl.getDocumentDOM().getTimeline().layers[0].name = "foreground";
```

layer.outline

Disponibilidad

Flash MX 2004.

Uso

```
layer.outline
```

Descripción

Propiedad; un valor booleano que especifica el estado de los contornos para todos los objetos de la capa. Si se define como `true`, todos los objetos de la capa aparecerán sólo con contornos. Si es `false`, los objetos aparecerán tal como se crearon.

Ejemplo

El ejemplo siguiente hace que todos los objetos de la primera capa aparezcan sólo con contornos:

```
fl.getDocumentDOM().getTimeline().layers[0].outline = true;
```

layer.parentLayer

Disponibilidad

Flash MX 2004.

Uso

```
layer.parentLayer
```

Descripción

Propiedad; un objeto Layer que representa la carpeta o capa de guía o enmascaramiento que contiene la capa. Los valores aceptables para la capa superior son una carpeta, guía o capa de máscara que precede a la capa, o la `parentLayer` de la capa anterior o siguiente. Al definir la `parentLayer` de la capa no se moverá la posición de la capa en la lista; el intento de definir una `parentLayer` de una capa que requiera movimiento no surtirá ningún efecto. Utiliza `null` para una capa de nivel superior.

Ejemplo

El ejemplo siguiente utiliza dos capas al mismo nivel en la misma línea de tiempo. La primera capa (layers[0]) se convierte en una carpeta y, a continuación, se define como carpeta principal de la segunda carpeta (layers[1]). Esta acción desplaza la segunda capa dentro de la primera.

```
var parLayer = fl.getDocumentDOM().getTimeline().layers[0];
parLayer.layerType = "folder";
fl.getDocumentDOM().getTimeline().layers[1].parentLayer = parLayer;
```

layer.visible

Disponibilidad

Flash MX 2004.

Uso

```
layer.visible
```

Descripción

Propiedad; un valor booleano que especifica si se muestran u ocultan los objetos de la capa en el escenario. Si se define como `true`, todos los objetos de la capa estarán visibles; si es `false`, estarán ocultos. El valor predeterminado es `true`.

Ejemplo

El ejemplo siguiente oculta todos los objetos de la primera capa:

```
fl.getDocumentDOM().getTimeline().layers[0].visible = false;
```

Objeto library

Disponibilidad

Flash MX 2004.

Descripción

El objeto library representa el panel Biblioteca. Es una propiedad del objeto Document (consulte `document.library`) y `fl.getDocumentDOM().library` puede acceder a él.

El objeto library contiene una matriz de elementos de distintos tipos, como símbolos, mapas de bits, sonidos y vídeo.

Resumen de métodos del objeto library

Los métodos siguientes están disponibles para el objeto library:

Método	Descripción
`library.addItemToDocument()`	Añade el elemento actual o especificado al escenario en la posición especificada.
`library.addNewItem()`	Crea un elemento nuevo del tipo especificado en el panel Biblioteca y define el nuevo elemento como elemento seleccionado actualmente.
`library.deleteItem()`	Elimina los elementos actuales o un elemento especificado del panel Biblioteca.
`library.duplicateItem()`	Realiza una copia del elemento especificado o seleccionado actualmente.
`library.editItem()`	Abre el elemento especificado o seleccionado actualmente en modo de edición.
`library.expandFolder()`	Expande o contrae la carpeta especificada o seleccionada actualmente en la biblioteca.
`library.findItemIndex()`	Devuelve el valor de índice del elemento de biblioteca (basado en cero).
`library.getItemProperty()`	Obtiene la propiedad del elemento seleccionado.
`library.getItemType()`	Obtiene el tipo de objeto seleccionado actualmente o especificado por una ruta de biblioteca.
`library.getSelectedItems()`	Obtiene la matriz de todos los elementos seleccionados actualmente en la biblioteca.

Método	Descripción
`library.importEmbeddedSWF()`	Importa un archivo Shockwave (SWF) a la biblioteca como clip compilado.
`library.itemExists()`	Comprueba si el elemento especificado existe en la biblioteca.
`library.moveToFolder()`	Desplaza el elemento de biblioteca seleccionado actualmente o especificado a una carpeta especificada.
`library.newFolder()`	Crea una carpeta con el nombre especificado o un nombre predeterminado en la carpeta seleccionada (`"untitled folder #"`) si no se suministra ningún parámetro `folderName`.
`library.renameItem()`	Cambia el nombre del elemento de biblioteca seleccionado actualmente en el panel Biblioteca.
`library.selectAll()`	Selecciona o anula la selección de todos los elementos de la biblioteca.
`library.selectItem()`	Selecciona un elemento de biblioteca especificado.
`library.selectNone()`	Selecciona todos los elementos de la biblioteca.
`library.setItemProperty()`	Define la propiedad de todos los elementos de biblioteca seleccionados (ignorando las carpetas).
`library.updateItem()`	Actualiza el elemento especificado.

Resumen de propiedades del objeto library

La propiedad siguiente está disponible para el objeto library.

Propiedad	Descripción
`library.items`	Una matriz de objetos de elemento de la biblioteca

library.addItemToDocument()

Disponibilidad

Flash MX 2004.

Uso

```
library.addItemToDocument( position [, namePath] )
```

Parámetros

position Un punto que especifica la posición *x,y* del centro del elemento en el escenario.

namePath Una cadena que especifica el nombre del elemento. Si el elemento se encuentra en una carpeta, puede especificar su nombre y su ruta empleando notación con barras. Si no se especifica *namePath*, se utilizará la selección actual de la biblioteca. Este parámetro es opcional.

Valor devuelto

Un valor booleano: `true` si el elemento se añade correctamente al documento; `false` en caso contrario.

Descripción

Método; añade el elemento actual o especificado al escenario en la posición especificada.

Ejemplo

El ejemplo siguiente añade el elemento seleccionado actualmente al escenario en la posición (3, 60):

```
fl.getDocumentDOM().library.addItemToDocument({x:3, y:60});
```

El ejemplo siguiente añade el elemento Symbol1 situado en la carpeta 1 de la biblioteca al escenario en la posición (550, 485):

```
fl.getDocumentDOM().library.addItemToDocument({x:550.0, y:485.0}, "folder1/
  Symbol1");
```

library.addNewItem()

Disponibilidad

Flash MX 2004.

Uso

```
library.addNewItem( type [, namePath] )
```

Parámetros

type Una cadena que especifica el tipo de elemento que se va a crear. Los únicos valores aceptables de *type* son `"video"`, `"movie clip"`, `"button"`, `"graphic"`, `"bitmap"`, `"screen"` y `"folder"` (por ejemplo, no se puede añadir un sonido a la biblioteca con este método). Especificar una ruta de carpeta es lo mismo que utilizar `library.newFolder()` antes de llamar a este método.

namePath Una cadena que especifica el nombre del elemento que se va a añadir. Si el elemento se encuentra en una carpeta, especifique su nombre y su ruta empleando notación con barras. Este parámetro es opcional.

Valor devuelto

Un valor booleano: `true` si el elemento se crea correctamente, y `false` en caso contrario.

Descripción

Método; crea un elemento nuevo del tipo especificado en el panel Biblioteca y define el nuevo elemento como elemento seleccionado actualmente. Para más información sobre la importación de elementos a la biblioteca, incluidos elementos como sonidos, consulte `document.importFile()`.

Ejemplo

El ejemplo siguiente crea un elemento de botón nuevo llamado start en una biblioteca nueva llamada folderTwo:

```
fl.getDocumentDOM().library.addNewItem("button", "folderTwo/start");
```

library.deleteItem()

Disponibilidad

Flash MX 2004.

Uso

```
library.deleteItem( [ namePath ] )
```

Parámetros

namePath Una cadena que especifica el nombre del elemento que se va a eliminar. Si el elemento se encuentra en una carpeta, puede especificar su nombre y su ruta empleando notación con barras. Si transfiere un nombre de carpeta, se eliminarán la carpeta y todos sus elementos. Si no especifica ningún nombre, Flash eliminará el elemento o elementos seleccionados actualmente. Para eliminar todos los elementos del panel Biblioteca, seleccione todos los elementos antes de utilizar este método. Este parámetro es opcional.

Valor devuelto

Un valor booleano: `true` si los elementos se eliminan correctamente, y `false` en caso contrario.

Descripción

Método; elimina los elementos actuales o un elemento especificado del panel Biblioteca. Este método puede afectar a múltiples elementos si hay varios seleccionados.

Ejemplo

El ejemplo siguiente elimina el elemento seleccionado actualmente:

```
fl.getDocumentDOM().library.deleteItem();
```

El ejemplo siguiente elimina el elemento Symbol_1 de la carpeta de biblioteca Folder_1:

```
fl.getDocumentDOM().library.deleteItem("Folder_1/Symbol_1");
```

library.duplicateItem()

Disponibilidad

Flash MX 2004.

Uso

```
library.duplicateItem( [ namePath ] )
```

Parámetros

namePath Una cadena que especifica el nombre del elemento que se va a duplicar. Si el elemento se encuentra en una carpeta, puede especificar su nombre y su ruta empleando notación con barras. Este parámetro es opcional.

Valor devuelto

Un valor booleano: `true` si el elemento se duplica correctamente, y `false` en caso contrario. Si hay varios elementos seleccionados, Flash devolverá `false`.

Descripción

Método; realiza una copia del elemento especificado o seleccionado actualmente. El nuevo elemento tiene un nombre predeterminado (como `item copy`) y se define como el elemento seleccionado actualmente. Si hay varios elementos seleccionados, el comando produce un error.

Ejemplo

El ejemplo siguiente crea una copia del elemento cuadrado en la carpeta de prueba de la biblioteca:

```
fl.getDocumentDOM().library.duplicateItem("test/square");
```

library.editItem()

Disponibilidad

Flash MX 2004.

Uso

```
library.editItem( [ namePath ] )
```

Parámetros

namePath Una cadena que especifica el nombre del elemento. Si el elemento se encuentra en una carpeta, puede especificar su nombre y su ruta empleando notación con barras. Si no se especifica *namePath*, se abrirá el elemento de biblioteca seleccionado en modo de edición. Si no hay ningún elemento seleccionado en la biblioteca o hay varios, aparecerá la primera escena de la línea de tiempo principal para su edición. Este parámetro es opcional.

Valor devuelto

Un valor booleano: `true` si existe el elemento especificado y se puede editar, y `false` en caso contrario.

Descripción

Método; abre el elemento especificado o seleccionado actualmente en modo de edición.

Ejemplo

El ejemplo siguiente abre el elemento círculo en la carpeta de prueba de la biblioteca para su edición:

```
fl.getDocumentDOM().library.editItem("test/circle");
```

library.expandFolder()

Disponibilidad

Flash MX 2004.

Uso

```
library.expandFolder( bExpand [, bRecurseNestedParents [, namePath ] ] )
```

Parámetros

bExpand Un valor booleano: si es `true`, la carpeta se expande; si es `false` (opción predeterminada), la carpeta se contrae.

bRecurseNestedParents Un valor booleano: si es `true`, todas las carpetas de la carpeta especificada se expanden o contraen, según el valor de *bExpand*. El valor predeterminado es `false`. Este parámetro es opcional.

namePath Una cadena que especifica el nombre y, opcionalmente, la ruta de la carpeta que se va a expandir o contraer. Si no se especifica este parámetro, el método se aplica a la carpeta seleccionada actualmente. Este parámetro es opcional.

Valor devuelto

Un valor booleano: `true` si el elemento se expande o contrae correctamente; `false` si se produce un error o el elemento especificado no es una carpeta.

Descripción

Método; expande o contrae la carpeta especificada o seleccionada actualmente en la biblioteca.

Ejemplo

El ejemplo siguiente contrae la carpeta de prueba de la biblioteca, así como todas las carpetas que se encuentran en la carpeta de prueba (en su caso):

```
fl.getDocumentDOM().library.expandFolder(false, true, "test");
```

library.findItemIndex()

Disponibilidad

Flash MX 2004.

Uso

```
library.findItemIndex( namePath )
```

Parámetros

namePath Una cadena que especifica el nombre del elemento. Si el elemento se encuentra en una carpeta, puede especificar su nombre y su ruta empleando notación con barras.

Valor devuelto

Un valor entero que representa el valor de índice basado en cero del elemento.

Descripción

Método; devuelve el valor de índice del elemento de biblioteca (basado en cero). El índice de biblioteca es plano, por lo que las carpetas se consideran parte del índice principal. Puede utilizar rutas de carpeta para especificar un elemento anidado.

Ejemplo

El ejemplo siguiente almacena el valor de índice basado en cero del elemento de biblioteca square, que se encuentra en una carpeta de prueba, en la variable `sqIndex` y, a continuación, muestra el valor de índice en un cuadro de diálogo:

```
var sqIndex = fl.getDocumentDOM().library.findItemIndex("test/square");
alert(sqIndex);
```

library.getItemProperty()

Disponibilidad

Flash MX 2004.

Uso

```
library.getItemProperty( property )
```

Parámetros

property Una cadena. Para obtener una lista de valores que puede utilizar como parámetro *property*, consulte Resumen de propiedades del objeto Item, junto con el resumen de propiedades para sus subclases.

Valor devuelto

Un valor de cadena para la propiedad.

Descripción

Método; obtiene la propiedad del elemento seleccionado.

Ejemplo

El ejemplo siguiente muestra un cuadro de diálogo que contiene el valor del identificador de vinculación para el símbolo cuando se hace referencia a él empleando ActionScript o para compartir tiempo de ejecución:

```
alert(fl.getDocumentDOM().library.getItemProperty("linkageIdentifier"));
```

library.getItemType()

Disponibilidad

Flash MX 2004.

Uso

```
library.getItemType( [ namePath ] )
```

Parámetros

namePath Una cadena que especifica el nombre del elemento. Si el elemento se encuentra en una carpeta, especifique su nombre y su ruta empleando notación con barras. Si no se especifica *namePath*, Flash proporcionará el tipo de la selección actual. Si hay varios elementos seleccionados actualmente y no se suministra *namePath*, Flash ignorará el comando. Este parámetro es opcional.

Valor devuelto

Un valor de cadena que especifica el tipo de objeto. Para posibles valores devueltos, consulte `item.itemType`.

Descripción

Método; obtiene el tipo de objeto seleccionado actualmente o especificado por una ruta de biblioteca.

Ejemplo

El ejemplo siguiente muestra un cuadro de diálogo que contiene el tipo de elemento de Symbol_1 situado en la carpeta Folder_1/Folder_2:

```
alert(fl.getDocumentDOM().library.getItemType("Folder_1/Folder_2/
  Symbol_1"));
```

library.getSelectedItems()

Disponibilidad

Flash MX 2004.

Parámetros

Ninguno.

Valor devuelto

Una matriz de valores para todos los elementos seleccionados actualmente en la biblioteca.

Descripción

Método; obtiene la matriz de todos los elementos seleccionados actualmente en la biblioteca.

Ejemplo

El ejemplo siguiente almacena la matriz de elementos de biblioteca seleccionados actualmente (en este caso, varios archivos de audio) en la variable `selItems` y, a continuación, cambia la propiedad `sampleRate` del primer archivo de audio de la matriz a `"11 kHz"`:

```
var selItems = fl.getDocumentDOM().library.getSelectedItems();
selItems[0].sampleRate = "11 kHz";
```

library.importEmbeddedSWF()

Disponibilidad

Flash MX 2004.

Uso

```
library.importEmbeddedSWF( linkageName, swfData [, libName] )
```

Parámetros

linkageName Una cadena que proporciona el nombre de la vinculación SWF del clip de película raíz.

swfData Una matriz de datos SWF binarios que procede de una biblioteca externa o DLL.

libName Una cadena que especifica el nombre de biblioteca del elemento creado. Si ya se utiliza el nombre, el método creará un nombre alternativo. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; importa un archivo Shockwave (SWF) en la biblioteca como clip compilado. A diferencia de Archivo > Importar > SWF, este método permite incorporar un archivo SWF compilado dentro de la biblioteca. No hay funcionalidad equivalente en la interfaz de usuario, y este método debe emplearse con una biblioteca externa o DLL (consulte el Capítulo 3, “Extensibilidad de nivel C”, en la página 543).

El archivo SWF que está importando debe tener un clip de película de nivel superior que incluya todo el contenido. Dicho clip deberá disponer de su propio identificador de vinculación definido con el mismo valor que el parámetro *linkageName* transferido a este método.

Ejemplo

El ejemplo siguiente añade el archivo SWF con el valor *linkageName* de MyMovie a la biblioteca como clip compilado con el nombre Intro:

```
fl.getDocumentDOM().library.importEmbeddedSWF("MyMovie", swfData, "Intro");
```

library.itemExists()

Disponibilidad

Flash MX 2004.

Uso

```
library.itemExists( namePath )
```

Parámetros

namePath Una cadena que especifica el nombre del elemento. Si el elemento se encuentra en una carpeta, especifique su nombre y su ruta empleando notación con barras.

Valor devuelto

Un valor booleano: `true` si existe el elemento especificado en la biblioteca, y `false` en caso contrario.

Descripción

Método; comprueba si el elemento especificado existe en la biblioteca.

Ejemplo

El ejemplo siguiente muestra `true` o `false` en un cuadro de diálogo, dependiendo de si el elemento Symbol_1 se encuentra en la carpeta de biblioteca Folder_1:

```
alert(fl.getDocumentDOM().library.itemExists('Folder_1/Symbol_1'));
```

library.items

Disponibilidad

Flash MX 2004.

Uso

```
library.items
```

Descripción

Propiedad; una matriz de objetos de elemento de la biblioteca.

Ejemplo

El ejemplo siguiente almacena la matriz de todos los elementos de biblioteca en la variable `itemArray`:

```
var itemArray = fl.getDocumentDOM().library.items;
```

library.moveToFolder()

Disponibilidad

Flash MX 2004.

Uso

```
library.moveToFolder( folderPath [, itemToMove [, bReplace ] ] )
```

Parámetros

folderPath Una cadena que especifica la ruta de la carpeta con la forma "FolderName" o "FolderName/FolderName". Para desplazar un elemento al nivel superior, especifique una cadena vacía ("") para *folderPath*.

itemToMove Una cadena que especifica el nombre del elemento que se va a desplazar. Si no especifica *itemToMove*, se desplazarán los elementos seleccionados actualmente. Este parámetro es opcional.

bReplace Un valor booleano. Si ya existe un elemento con el mismo nombre y especifica `true` para el parámetro *bReplace*, se sustituirá el elemento existente por el elemento que se está desplazando. Si es `false`, el nombre del elemento colocado cambiará a un nombre exclusivo. El valor predeterminado es `false`. Este parámetro es opcional.

Valor devuelto

Un valor booleano: `true` si el elemento se desplaza correctamente, y `false` en caso contrario.

Descripción

Método; desplaza el elemento de biblioteca seleccionado actualmente o especificado a una carpeta especificada. Si el parámetro *folderPath* está vacío, los elementos se desplazan al nivel superior.

Ejemplo

El ejemplo siguiente desplaza el elemento Symbol_1 a la carpeta de biblioteca new y reemplaza el elemento en esa carpeta con el mismo nombre:

```
fl.getDocumentDOM().library.moveToFolder("new", "Symbol_1", true);
```

library.newFolder()

Disponibilidad

Flash MX 2004.

Uso

```
library.newFolder( [folderPath] )
```

Parámetros

folderPath Una cadena que especifica el nombre de la carpeta que se va a crear. Si se especifica como una ruta y la ruta no existe, se creará la ruta. Este parámetro es opcional.

Valor devuelto

Un valor booleano: `true` si la carpeta se crea correctamente, y `false` en caso contrario.

Descripción

Método; crea una carpeta con el nombre especificado o un nombre predeterminado en la carpeta seleccionada (`"untitled folder #"`) si no se suministra ningún parámetro *folderName*.

Ejemplo

El ejemplo siguiente crea dos nuevas carpetas de biblioteca; la segunda carpeta es una subcarpeta de la primera carpeta:

```
fl.getDocumentDOM().library.newFolder("first/second");
```

library.renameItem()

Disponibilidad

Flash MX 2004.

Uso

```
library.renameItem(name)
```

Parámetros

name Una cadena que especifica un nombre nueva para el elemento de biblioteca.

Valor devuelto

Un valor booleano de `true` si el nombre del elemento se cambia correctamente, `false` en caso contrario. Si hay varios elementos seleccionados, los nombres no cambiarán y el valor devuelto será `false` (de acuerdo con el comportamiento de la interfaz de usuario).

Descripción

Método; cambia el nombre del elemento de biblioteca seleccionado actualmente en el panel Biblioteca.

Ejemplo

El ejemplo siguiente cambia el nombre del elemento de biblioteca seleccionado por `"new name"`:

```
fl.getDocumentDOM().library.renameItem("new name");
```

library.selectAll()

Disponibilidad

Flash MX 2004.

Uso

```
library.selectAll( [ bSelectAll ] )
```

Parámetros

`bSelectAll` Un valor booleano que especifica si se selecciona o anula la selección de todos los elementos de la biblioteca. Omita este parámetro o utilice el valor predeterminado de `true` para seleccionar todos los elementos de la biblioteca; `false` anula la selección de todos los elementos de biblioteca. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; selecciona o anula la selección de todos los elementos de la biblioteca.

Ejemplo

Los ejemplos siguientes seleccionan todos los elementos de la biblioteca:

```
fl.getDocumentDOM().library.selectAll();
fl.getDocumentDOM().library.selectAll(true);
```

Los ejemplos siguientes anulan la selección de todos los elementos de la biblioteca:

```
fl.getDocumentDOM().library.selectAll(false);
fl.getDocumentDOM().library.selectNone();
```

library.selectItem()

Disponibilidad

Flash MX 2004.

Uso

```
library.selectItem( namePath [, bReplaceCurrentSelection [, bSelect ] ] )
```

Parámetros

namePath Una cadena que especifica el nombre del elemento. Si el elemento se encuentra en una carpeta, puede especificar su nombre y su ruta empleando notación con barras.

bReplaceCurrentSelection Un valor booleano que especifica si se va a reemplazar la selección actual o si se va a añadir el elemento a la selección actual. El valor predeterminado es `true` (reemplaza la selección actual). Este parámetro es opcional.

bSelect Un valor booleano que especifica si se va a seleccionar o anular la selección de un elemento. El valor predeterminado es `true` (seleccionar). Este parámetro es opcional.

Valor devuelto

Un valor booleano: `true` si existe el elemento especificado; `false` en caso contrario.

Descripción

Método; selecciona un elemento de biblioteca especificado.

Ejemplo

El ejemplo siguiente cambia la selección actual de la biblioteca a símbolo 1 dentro de la carpeta sin título 1:

```
fl.getDocumentDOM().library.selectItem("untitled Folder_1/Symbol_1");
```

El ejemplo siguiente amplía lo que se encuentra seleccionado actualmente en la biblioteca para incluir el símbolo 1 dentro de la carpeta sin título 1:

```
fl.getDocumentDOM().library.selectItem("untitled Folder_1/Symbol_1",
  false);
```

El ejemplo siguiente anula la selección del símbolo 1 dentro de la carpeta sin título 1 y no cambia otros elementos seleccionados:

```
fl.getDocumentDOM().library.selectItem("untitled Folder_1/Symbol_1", true,
  false);
```

library.selectNone()

Disponibilidad

Flash MX 2004.

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; anula la selección de todos los elementos de biblioteca.

Ejemplo

Los ejemplos siguientes anulan la selección de todos los elementos de la biblioteca:

```
fl.getDocumentDOM().library.selectNone();
fl.getDocumentDOM().library.selectAll(false);
```

library.setItemProperty()

Disponibilidad

Flash MX 2004.

Uso

```
library.setItemProperty( property, value )
```

Parámetros

property Una cadena que es el nombre de la propiedad que se va a definir. Para obtener una lista de propiedades, consulte Resumen de propiedades del objeto Item y los resúmenes de propiedades de sus subclases. Para ver qué objetos son subclases del objeto Item, consulte Resumen de la estructura del DOM.

value El valor que se va a asignar a la propiedad especificada.

Valor devuelto

Ninguno.

Descripción

Método; define la propiedad de todos los elementos de biblioteca seleccionados (ignorando las carpetas).

Ejemplo

El ejemplo siguiente asigna el valor botón a la propiedad `symbolType` para el elemento o elementos de biblioteca seleccionados. En este caso, el elemento debe ser un Objeto SymbolItem; `symbolType` es una propiedad válida para objetos SymbolItem.

```
fl.getDocumentDOM().library.setItemProperty("symbolType", "button");
```

library.updateItem()

Disponibilidad

Flash MX 2004.

Uso

```
library.updateItem( [ namePath ] )
```

Parámetros

`namePath` Una cadena que especifica el nombre del elemento. Si el elemento se encuentra en una carpeta, especifique su nombre y su ruta empleando la notación con barras. En la interfaz de usuario equivale a hacer clic con el botón derecho del ratón en un elemento y seleccionar Actualizar en el menú. Si no se suministra ningún nombre, se actualizará la selección actual. Este parámetro es opcional.

Valor devuelto

Un valor booleano: `true` si Flash actualiza el elemento correctamente, y `false` en caso contrario.

Descripción

Método; actualiza el elemento especificado.

Ejemplo

El ejemplo siguiente muestra un cuadro de diálogo que indica si el elemento seleccionado se actualiza (`true`) o no (`false`):

```
alert(fl.getDocumentDOM().library.updateItem());
```

Objeto Math

Disponibilidad

Flash MX 2004.

Descripción

El objeto Math está disponible como propiedad de sólo lectura del objeto Flash; consulte `fl.Math`. Este objeto proporciona métodos que realizan operaciones matemáticas comunes.

Resumen de métodos del objeto Math

Los métodos siguientes están disponibles para el objeto Math:

Método	Descripción
`Math.concatMatrix()`	Realiza una concatenación de matrices y devuelve el resultado.
`Math.invertMatrix()`	Devuelve la inversa de la matriz especificada.
`Math.pointDistance()`	Calcula la distancia entre dos puntos.

Math.concatMatrix()

Disponibilidad

Flash MX 2004.

Uso

```
Math.concatMatrix(mat1, mat2)
```

Parámetros

`mat1` y *`mat2`* Especifica los objetos Matrix que se van a concatenar (véase Objeto Matrix). Cada parámetro debe ser un objeto con campos `a`, `b`, `c`, `d`, `tx` y `ty`.

Valor devuelto

Una matriz de objetos concatenados.

Descripción

Método; realiza una concatenación de matrices y devuelve el resultado.

Ejemplo

El ejemplo siguiente almacena el objeto seleccionado actualmente en la variable `elt`, multiplica la matriz de objetos por la matriz de vistas y almacena ese valora en la variable `mat`:

```
var elt = fl.getDocumentDOM().selection[0];
var mat = fl.Math.concatMatrix( elt.matrix , fl.getDocumentDOM().viewMatrix
  );
```

Math.invertMatrix()

Disponibilidad

Flash MX 2004.

Uso

```
Math.invertMatrix(mat)
```

Parámetros

mat Indica el objeto Matrix que se va a invertir (véase Objeto Matrix). Deberá tener los campos siguientes: `a`, `b`, `c`, `d`, `tx` y `ty`.

Valor devuelto

Un objeto Matrix que es la inversa de la matriz original.

Descripción

Método; devuelve la inversa de la matriz especificada.

Ejemplo

El ejemplo siguiente almacena el objeto seleccionado en la variable `elt`, asigna esa matriz a la variable `mat` y almacena la inversa de la matriz en la variable `inv`:

```
var elt = fl.getDocumentDOM().selection[0];
var mat = elt.matrix;
var inv = fl.Math.invertMatrix( mat );
```

Math.pointDistance()

Disponibilidad

Flash MX 2004.

Uso

```
Math.pointDistance(pt1, pt2)
```

Parámetros

pt1 y *pt2* Especifica los puntos entre los que se mide la distancia.

Valor devuelto

Un valor de coma flotante que representa la distancia entre los puntos.

Descripción

Método; calcula la distancia entre dos puntos.

Ejemplo

El ejemplo siguiente almacena el valor de la distancia entre *pt1* y *pt2* en la variable `dist`:

```
var pt1 = {x:10, y:20}
var pt2 = {x:100, y:200}
var dist = fl.Math.pointDistance(pt1, pt2);
```

Objeto Matrix

Disponibilidad

Flash MX 2004.

Descripción

El objeto Matrix representa una matriz de transformación.

Resumen de propiedades del objeto Matrix

Las propiedades siguientes están disponibles para el objeto Matrix:

Propiedad	Descripción
`matrix.a`	Un valor de coma flotante que especifica el elemento (0,0) de la matriz de transformación.
`matrix.b`	Un valor de coma flotante que especifica el elemento (0,1) de la matriz.
`matrix.c`	Un valor de coma flotante que especifica el elemento (1,0) de la matriz.
`matrix.d`	Un valor de coma flotante que especifica el elemento (1,1) de la matriz.
`matrix.tx`	Un valor de coma flotante que especifica la ubicación en el eje x del punto de registro de un símbolo o el centro de una forma.
`matrix.ty`	Un valor de coma flotante que especifica la ubicación en el eje y del punto de registro de un símbolo o el centro de una forma.

matrix.a

Disponibilidad

Flash MX 2004.

Uso

```
matrix.a
```

Descripción

Propiedad; un valor de coma flotante que especifica el elemento (0,0) de la matriz de transformación. Este valor representa el factor de escala del eje *x* del objeto.

Ejemplo

Las propiedades `a` y `d` de una matriz representan el escalado. En el ejemplo siguiente, los valore se definen como 2 y 3, respectivamente, para aumentar el objeto seleccionado dos veces su anchura y tres veces su altura:

```
var mat = fl.getDocumentDOM().selection[0].matrix;
mat.a = 2;
mat.d = 3;
fl.getDocumentDOM().selection[0].matrix = mat;
```

Puede girar un objeto definiendo las propiedades de matriz `a`, `b`, `c` y `d` en relación mutua, donde `a = d` y `b = -c`. Por ejemplo, los valores de 0,5, 0,8, -0,8 y 0,5 giran el objeto 60º:

```
var mat = fl.getDocumentDOM().selection[0].matrix;
mat.a = 0.5;
mat.b = 0.8;
mat.c = 0.8*(-1);
mat.d = 0.5;
fl.getDocumentDOM().selection[0].matrix = mat;
```

Puede definir `a = d = 1` y `c = b = 0` para devolver la forma original al objeto.

matrix.b

Disponibilidad

Flash MX 2004.

Uso

```
matrix.b
```

Descripción

Propiedad; un valor de coma flotante que especifica el elemento (0,1) de la matriz. Este valor representa el sesgo vertical de una forma; hace que Flash desplace el borde derecho de la forma a lo largo del eje vertical.

Las propiedades `matrix.b` y `matrix.c` de una matriz representan el sesgo (véase `matrix.c`).

Ejemplo

En el ejemplo siguiente, puede definir `b` y `c` como -1 y 0 respectivamente; estos valores sesgan el objeto en un ángulo vertical de 45º:

```
var mat = fl.getDocumentDOM().selection[0].matrix;
mat.b = -1;
mat.c = 0;
fl.getDocumentDOM().selection[0].matrix = mat;
```

Para sesgar el objeto en su forma original, puede definir `b` y `c` como 0.

Consulte el ejemplo de `matrix.a`.

matrix.c

Disponibilidad

Flash MX 2004.

Uso

```
matrix.c
```

Descripción

Propiedad; un valor de coma flotante que especifica el elemento (1,0) de la matriz. Este valor hace que Flash sesgue el objeto desplazando su borde inferior a lo largo del eje horizontal.

Las propiedades `matrix.b` y `matrix.c` de una matriz representan el sesgo.

Ejemplo

Consulte el ejemplo de `matrix.b`.

matrix.d

Disponibilidad

Flash MX 2004.

Uso

```
matrix.d
```

Descripción

Propiedad; un valor de coma flotante que especifica el elemento (1,1) de la matriz. Este valor representa el factor de escala del eje *y* del objeto.

Ejemplo

Consulte el ejemplo de `matrix.a`.

matrix.tx

Disponibilidad

Flash MX 2004.

Uso

```
matrix.tx
```

Descripción

Propiedad; un valor de coma flotante que especifica la ubicación en el eje *x* del punto de registro de un símbolo o el centro de una forma. Define la traducción de *x* de la transformación.

Puede desplazar un objeto definiendo las propiedades `matrix.tx` y `matrix.ty` (véase `matrix.ty`).

Ejemplo

En el ejemplo siguiente, si define `tx` y `ty` como 0 se desplazará el punto de registro del objeto al punto 0,0 del documento:

```
var mat = fl.getDocumentDOM().selection[0].matrix;
mat.tx = 0;
mat.ty = 0;
fl.getDocumentDOM().selection[0].matrix = mat;
```

matrix.ty

Disponibilidad

Flash MX 2004.

Uso

```
matrix.ty
```

Descripción

Propiedad; un valor de coma flotante que especifica la ubicación en el eje *y* del punto de registro de un símbolo o el centro de una forma. Define la traducción de *y* de la transformación.

Puede desplazar un objeto definiendo las propiedades `matrix.tx` y `matrix.ty`.

Ejemplo

Consulte el ejemplo de `matrix.tx`.

Objeto outputPanel

Disponibilidad

Flash MX 2004.

Descripción

Este objeto representa el panel Salida, que muestra información de resolución de problemas, como errores de sintaxis. Para acceder a este objeto, utilice `fl.outputPanel` (o `flash.outputPanel`). Véase `fl.outputPanel`.

Resumen de métodos del objeto outputPanel

El objeto outputPanel utiliza los métodos siguientes.

Método	Descripción
`outputPanel.clear()`	Borra el contenido del panel Salida.
`outputPanel.save()`	Guarda el contenido del panel Salida en un archivo de texto local.
`outputPanel.trace()`	Añade una línea al contenido del panel Salida, terminada con una línea nueva.

outputPanel.clear()

Disponibilidad

Flash MX 2004.

Uso

```
outputPanel.clear()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; borra el contenido del panel Salida. Puede utilizar este método en una aplicación de procesamiento por lotes para borrar una lista de errores o para guardarlos incrementalmente empleando este método con `outputPanel.save()`.

Ejemplo

El ejemplo siguiente borra el contenido actual del panel Salida:

```
fl.outputPanel.clear();
```

outputPanel.save()

Disponibilidad

Flash MX 2004; parámetro *bUseSystemEncoding* añadido en Flash 8.

Uso

```
outputPanel.save(fileURI [, bAppendToFile [ , bUseSystemEncoding ] ])
```

Parámetros

fileURI Una cadena, expresada como archivo:/// URI, que especifica el archivo local que albergará el contenido del panel Salida.

bAppendToFile Un valor booleano opcional. Si es `true`, añade el contenido del panel Salida al archivo de salida, y si es `false`, el método sobrescribe el archivo de salida si ya existe. El valor predeterminado es `false`.

bUseSystemEncoding Un valor booleano opcional. Si es `true`, guarda el texto del panel Salida con la codificación del sistema; si es `false`, guarda el texto del panel de salida con la codificación UTF-8, con los caracteres de marca de orden de byte al inicio del texto. El valor predeterminado es `false`.

Valor devuelto

Ninguno.

Descripción

Método; guarda el contenido del panel Salida en un archivo de texto local. También puede especificar que el contenido se añada al contenido de un archivo local en lugar de sobrescribirlo. Si *fileURI* no es válido o no se especifica, se produce un error.

Este método resulta útil para procesamiento por lotes. Por ejemplo, puede crear un archivo JSFL que compile varios componentes. Los errores de compilación aparecerán en el panel Salida y podrá utilizar este método para guardar los errores resultantes en un archivo de texto que el sistema de compilación utilizado puede analizar automáticamente.

Ejemplo

El ejemplo siguiente guarda el contenido del panel Salida en un archivo batch.log en la carpeta /tests, sobrescribiendo el archivo batch.log si ya existe:

```
fl.outputPanel.save("file:///c|/tests/batch.log");
```

outputPanel.trace()

Disponibilidad

Flash MX 2004.

Uso

```
outputPanel.trace(message)
```

Parámetros

El parámetro *message* es una cadena que contiene el texto que se va a añadir al panel Salida.

Valor devuelto

Ninguno.

Descripción

Método; envía una cadena de texto al panel Salida, terminada con una línea nueva y muestra el panel Salida si aún no está visible. Este método es idéntico a `fl.trace()` y funciona igual que la sentencia `trace()` en ActionScript.

Para enviar una línea en blanco, utilice `outputPanel.trace("")` o `outputPanel.trace("\n")`. Puede utilizar estos comandos en línea, convirtiendo `\n` en parte de la cadena *message*.

Ejemplo

El ejemplo siguiente muestra varias líneas de texto en el panel Salida:

```
fl.outputPanel.clear();
fl.outputPanel.trace("Hello World!!!");
var myPet = "cat";
fl.outputPanel.trace("\nI have a " + myPet);
fl.outputPanel.trace("");
fl.outputPanel.trace("I love my " + myPet);
fl.outputPanel.trace("Do you have a " + myPet +"?");
```

Objeto Parameter

Disponibilidad

Flash MX 2004.

Descripción

El acceso al tipo de objeto Parameter se realiza desde la matriz `screen.parameters` (que corresponde al inspector de propiedades de la pantalla en la herramienta de edición de Flash) o la matriz `componentInstance.parameters` (que corresponde al inspector de propiedades del componente en la herramienta de edición). Véase `screen.parameters` y `componentInstance.parameters`.

Resumen de métodos del objeto Parameter

Los métodos siguientes están disponibles para el objeto Parameter:

Método	Descripción
`parameter.insertItem()`	Inserta un elemento en la lista, objeto o matriz.
`parameter.removeItem()`	Suprime un elemento del tipo de lista, objeto o matriz de un parámetro de pantalla o componente.

Resumen de propiedades del objeto Parameter

Las propiedades siguientes están disponibles para el objeto Parameter:

Propiedad	Descripción
`parameter.category`	Propiedad; una cadena que especifica la propiedad `category` para el parámetro `screen` o el parámetro `componentInstance`.
`parameter.listIndex`	Un entero que especifica el valor del elemento de lista seleccionado.
`parameter.name`	De sólo lectura; una cadena que especifica el nombre del parámetro.
`parameter.value`	Propiedad; corresponde al campo Valor de la ficha Parámetros del panel Inspector de componentes, la ficha Parámetros del inspector de propiedades o el inspector de propiedades de la pantalla.
`parameter.valueType`	De sólo lectura; una cadena que indica el tipo de parámetro de pantalla o componente.
`parameter.verbose`	Especifica dónde se muestra el parámetro.

parameter.category

Disponibilidad

Flash MX 2004.

Uso

```
parameter.category
```

Descripción

Propiedad; una cadena que especifica la propiedad `category` para el parámetro `screen` o el parámetro `componentInstance`. Esta propiedad ofrece una forma alternativa de presentar una lista de parámetros. Esta funcionalidad no está disponible a través de la interfaz de usuario de Flash.

parameter.insertItem()

Disponibilidad

Flash MX 2004.

Uso

```
parameter.insertItem(index, name, value, type)
```

Parámetros

`index` Un índice entero basado en cero que indica dónde se insertará el elemento en la lista, objeto o matriz. Si el índice es 0, el elemento se insertará al principio de la lista. Si el índice es mayor que el tamaño de lista, el nuevo elemento se insertará al final de la matriz.

`name` Una cadena que especifica el nombre del elemento que se va a insertar. Es un parámetro necesario para los parámetros de objeto.

`value` Una cadena que especifica el valor del elemento que se va a insertar.

`type` Una cadena que especifica el tipo de elemento que se va a insertar.

Valor devuelto

Ninguno.

Descripción

Método; inserta un elemento en la lista, objeto o matriz. Si un parámetro es una lista, objeto o matriz, la propiedad *`value`* será una matriz.

Ejemplo

El ejemplo siguiente inserta el valor de "New Value" en el parámetro labelPlacement:

```
// Selecciona una instancia de componente de botón en el escenario.
var parms = fl.getDocumentDOM().selection[0].parameters;
parms[2].insertItem(0, "name", "New Value", "String");
var values = parms[2].value;
for(var prop in values){
     fl.trace("labelPlacement parameter value = " + values[prop].value);
}
```

parameter.listIndex

Disponibilidad

Flash MX 2004.

Uso

```
parameter.listIndex
```

Descripción

Propiedad; el valor del elemento de lista seleccionado. Esta propiedad sólo es válida si el parámetro valueType es "List".

Ejemplo

El ejemplo siguiente define el primer parámetro para una diapositiva, que es el parámetro autoKeyNav. Para definir el parámetro con uno de sus valores válidos (true, false o inherit) parameter.listIndex se define con el índice del elemento de la lista (0 para true, 1 para false, 2 para inherit).

```
var parms = fl.getDocumentDOM().screenOutline.screens[1].parameters;
parms[0].listIndex = 1;
```

parameter.name

Disponibilidad

Flash MX 2004.

Uso

```
parameter.name
```

Descripción

Propiedad de sólo lectura; una cadena que especifica el nombre del parámetro.

Ejemplo

El ejemplo siguiente muestra el nombre del quinto parámetro para el componente seleccionado:

```
var parms = fl.getDocumentDOM().selection[0].parameters;
fl.trace("name: " + parms[4].name);
```

El ejemplo siguiente muestra el nombre del quinto parámetro para la pantalla especificada:

```
var parms = fl.getDocumentDOM().screenOutline.screens[1].parameters;
  fl.trace("name: " + parms[4].name);
```

parameter.removeItem()

Disponibilidad

Flash MX 2004.

Uso

```
parameter.removeItem(index)
```

Parámetros

index El índice entero basado en cero del elemento que se va a suprimir eliminar de la propiedad de pantalla o componente.

Valor devuelto

Ninguno.

Descripción

Método; suprime un elemento del tipo de lista, objeto o matriz de un parámetro de pantalla o componente.

Ejemplo

El ejemplo siguiente suprime el elemento en el índice 1 del parámetro `labelPlacement` de un componente:

```
// Selecciona una instancia de componente de botón en el escenario.
var parms = fl.getDocumentDOM().selection[0].parameters;
var values = parms[2].value;
fl.trace("--Original--");
for(var prop in values){
  fl.trace("labelPlacement value = " + values[prop].value);
}
parms[2].removeItem(1);
```

```
var newValues = parms[2].value;
fl.trace("--After Removing Item--");
for(var prop in newValues){
  fl.trace("labelPlacement value = " + newValues[prop].value);
}
```

El ejemplo siguiente elimina el elemento en el índice 1 del parámetro `autoKeyNav` de una pantalla:

```
// Abre un documento de presentación.
var parms = fl.getDocumentDOM().screenOutline.screens[1].parameters;
var values = parms[0].value;
fl.trace("--Original--");
for(var prop in values){
  fl.trace("autoKeyNav value = " + values[prop].value);
}
parms[0].removeItem(1);

var newValues = parms[0].value;
fl.trace("--After Removing Item--");
for(var prop in newValues){
  fl.trace("autoKeyNav value = " + newValues[prop].value);
}
```

parameter.value

Disponibilidad

Flash MX 2004.

Uso

```
parameter.value
```

Descripción

Propiedad; corresponde al campo Valor de la ficha Parámetros del panel Inspector de componentes, la ficha Parámetros del inspector de propiedades o el inspector de propiedades de la pantalla. El tipo de la propiedad `value` está determinado por la propiedad `valueType` del parámetro (véase `parameter.valueType`).

parameter.valueType

Disponibilidad

Flash MX 2004.

Uso

```
parameter.valueType
```

Descripción

Propiedad de sólo lectura; una cadena que indica el tipo de parámetro de pantalla o componente. El tipo puede ser cualquiera de los valores siguientes: `"Default"`, `"Array"`, `"Object"`, `"List"`, `"String"`, `"Number"`, `"Boolean"`, `"Font Name"`, `"Color"`, `"Collection"`, `"Web Service URL"` o `"Web Service Operation"`.

Véase también

`parameter.value`

parameter.verbose

Disponibilidad

Flash MX 2004.

Uso

```
parameter.verbose
```

Descripción

Propiedad; especifica dónde se muestra el parámetro. Si el valor de esta propiedad es 0 (no detallado), el parámetro sólo se muestra en el inspector de componentes. Si es 1 (detallado), el parámetro se muestra en el inspector de componentes y en la ficha Parámetros del inspector de propiedades.

Objeto Path

Disponibilidad

Flash MX 2004.

Descripción

El objeto Path define una secuencia de segmentos de línea (recta, curva o ambas) que suele emplearse para crear herramientas ampliables. El ejemplo siguiente muestra una instancia de un objeto Path que devuelve el objeto flash:

```
path = fl.drawingLayer.newPath();
```

Véase también Objeto drawingLayer.

Resumen de métodos del objeto Path

Los métodos siguientes están disponibles para el objeto Path:

Método	Descripción
`path.addCubicCurve()`	Añade un segmento de curva cúbica de Bézier a la ruta.
`path.addCurve()`	Añade un segmento cuadrático de Bézier a la ruta.
`path.addPoint()`	Añade un punto a la ruta.
`path.clear()`	Elimina todos los puntos de la ruta.
`path.close()`	Añade un punto en la ubicación del primer punto de la ruta y extiende la ruta a ese punto, que cierra la ruta.
`path.makeShape()`	Crea una forma en el escenario utilizando la configuración actual de trazo y relleno.
`path.newContour()`	Inicia un nuevo contorno en la ruta.

Resumen de propiedades del objeto Path

Las propiedades siguientes están disponibles para el objeto Path:

Propiedad	Descripción
`path.nPts`	De sólo lectura; un entero que representa el número de puntos de la ruta.

path.addCubicCurve()

Disponibilidad

Flash MX 2004.

Uso

```
path.addCubicCurve(xAnchor, yAnchor, x2, y2, x3, y3, x4, y4)
```

Parámetros

xAnchor Un número de coma flotante que especifica la posición *x* del primer punto de control.

yAnchor Un número de coma flotante que especifica la posición *y* del primer punto de control.

x2 Un número de coma flotante que especifica la posición *x* del segundo punto de control.

y2 Un número de coma flotante que especifica la posición *y* del segundo punto de control.

x3 Un número de coma flotante que especifica la posición *x* del tercer punto de control.

y3 Un número de coma flotante que especifica la posición *y* del tercer punto de control.

x4 Un número de coma flotante que especifica la posición *x* del cuarto punto de control.

y4 Un número de coma flotante que especifica la posición *y* del cuarto punto de control.

Valor devuelto

Ninguno.

Descripción

Método; añade un segmento de curva cúbica de Bézier a la ruta.

Ejemplo

El ejemplo siguiente crea una ruta nueva, la almacena en la variable `myPath` y asigna la curva a la ruta:

```
var myPath = fl.drawingLayer.newPath();
myPath.addCubicCurve(0, 0, 10, 20, 20, 20, 30, 0);
```

path.addCurve()

Disponibilidad

Flash MX 2004.

Uso

```
path.addCurve(xAnchor, yAnchor, x2, y2, x3, y3)
```

Parámetros

xAnchor Un valor de coma flotante que especifica la posición *x* del primer punto de control.

yAnchor Un valor de coma flotante que especifica la posición *y* del primer punto de control.

x2 Un valor de coma flotante que especifica la posición *x* del segundo punto de control.

y2 Un valor de coma flotante que especifica la posición *y* del segundo punto de control.

x3 Un valor de coma flotante que especifica la posición *x* del tercer punto de control.

y3 Un valor de coma flotante que especifica la posición *y* del tercer punto de control.

Valor devuelto

Ninguno.

Descripción

Método; añade un segmento cuadrático de Bézier a la ruta.

Ejemplo

El ejemplo siguiente crea una ruta nueva, la almacena en la variable `myPath` y asigna la curva a la ruta:

```
var myPath = fl.drawingLayer.newPath();
myPath.addCurve(0, 0, 10, 20, 20, 0);
```

path.addPoint()

Disponibilidad

Flash MX 2004.

Uso

```
path.addPoint(x, y)
```

Parámetros

x Un valor de coma flotante que especifica la posición *x* del punto.

y Un valor de coma flotante que especifica la posición *y* del punto.

Valor devuelto

Ninguno.

Descripción

Método; añade un punto a la ruta.

Ejemplo

El ejemplo siguiente crea una ruta nueva, la almacena en la variable `myPath` y asigna el nuevo punto a la ruta:

```
var myPath = fl.drawingLayer.newPath();
myPath.addPoint(10, 100);
```

path.clear()

Disponibilidad

Flash MX 2004.

Uso

```
path.clear()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; elimina todos los puntos de la ruta.

Ejemplo

El ejemplo siguiente suprime todos los puntos de una ruta almacenada en la variable `myPath`:

```
var myPath = fl.drawingLayer.newPath();
myPath.clear();
```

path.close()

Disponibilidad

Flash MX 2004.

Uso

```
path.close()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; añade un punto en la ubicación del primer punto de la ruta y extiende la ruta a ese punto, que cierra la ruta. Si la ruta no tiene puntos, no se añadirán puntos.

Ejemplo

El ejemplo siguiente crea una ruta cerrada:

```
var myPath = fl.drawingLayer.newPath();
myPath.close();
```

path.makeShape()

Disponibilidad

Flash MX 2004.

Uso

```
path.makeShape([bSupressFill [, bSupressStroke]])
```

Parámetros

bSuppressFill Un valor booleano que, si define como `true`, suprime el relleno que se aplicaría a la forma. El valor predeterminado es `false`. Este parámetro es opcional.

bSupressStroke Un valor booleano que, si define como `true`, suprime el trazo que se aplicaría a la forma. El valor predeterminado es `false`. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; crea una forma en el escenario utilizando la configuración actual de trazo y relleno. La ruta se borra cuando se crea la forma. Este método tiene dos parámetros opcionales para suprimir el relleno y el trazo del objeto de forma resultante. Si omite estos parámetros o los define como `false` se emplearán los valores actuales de relleno y trazo.

Ejemplo

El ejemplo siguiente crea una forma con el relleno actual y sin trazo:

```
var myPath = fl.drawingLayer.newPath();
myPath.makeShape(false, true);
```

path.newContour()

Disponibilidad

Flash MX 2004.

Uso

```
path.newContour()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; inicia un nuevo contorno en la ruta.

Ejemplo

El ejemplo siguiente crea un cuadrado vacío:

```
var myPath = fl.drawingLayer.newPath();
myPath.addPoint( 0,  0);
myPath.addPoint( 0, 30);
myPath.addPoint(30, 30);
myPath.addPoint(30,  0);
myPath.addPoint( 0,  0);

myPath.newContour();
myPath.addPoint(10, 10);
myPath.addPoint(10, 20);
myPath.addPoint(20, 20);
myPath.addPoint(20, 10);
myPath.addPoint(10, 10);

myPath.makeShape();
```

path.nPts

Disponibilidad

Flash MX 2004.

Uso

```
path.nPts
```

Descripción

Propiedad de sólo lectura; un entero que representa el número de puntos de la ruta. Una nueva ruta tiene 0 puntos.

Ejemplo

El ejemplo siguiente utiliza el panel Salida para mostrar el número de puntos de la ruta a la que hace referencia la variable `myPath`:

```
var myPath = fl.drawingLayer.newPath();
var numOfPoints = myPath.nPts;
fl.trace("Number of points in the path: " + numOfPoints);
// Muestra: Número de puntos de la ruta: 0
```

Objeto Project

Disponibilidad

Flash 8.

Descripción

El objeto Project representa un archivo de proyecto de Flash (FLP). Puede utilizar los comandos siguientes para devolver un objeto Project:

- Para crear un nuevo archivo de proyecto, utilice `fl.createProject()`.
- Para abrir un archivo de proyecto existente, utilice `fl.openProject()`.
- Para devolver un objeto Project para el proyecto abierto actualmente, utilice `fl.getProject()`.

Resumen de métodos del objeto Project

Pueden emplearse los métodos siguientes con el objeto Project.

Método	Descripción
`project.addFile()`	Añade el archivo especificado al proyecto.
`project.canPublishProject()`	Determina si el proyecto se puede publicar.
`project.canTestProject()`	Determina si el proyecto se puede probar.
`project.findProjectItem()`	Busca un archivo especificado en el proyecto.
`project.publishProject()`	Publica los archivos FLA de un proyecto.
`project.testProject()`	Prueba el proyecto.

Resumen de propiedades del objeto Project

Pueden emplearse las propiedades siguientes con el objeto Project.

Propiedad	Descripción
`project.defaultItem`	Especifica el Objeto ProjectItem que representa el documento predeterminado del proyecto.
`project.items`	Una matriz de objetos ProjectItem (véase Objeto ProjectItem) contenida en el proyecto (propiedad de sólo lectura).

Propiedad	Descripción
`project.name`	El nombre del proyecto que aparece en el panel Proyecto.
`project.projectURI`	Una cadena que representa la ruta y el nombre del archivo de proyecto, expresada como archivo:/// URI (propiedad de sólo lectura).

project.addFile()

Disponibilidad

Flash 8.

Uso

```
project.addFile( fileURI [ , autoCreateFolder ] )
```

Parámetros

`fileURI` Una cadena que especifica el archivo que se va a añadir al proyecto, expresada como archivo:/// URI.

`autoCreateFolder` Un valor booleano opcional que especifica si se deben crear automáticamente carpetas en el panel Proyecto para reflejar la ruta de *`fileURI`*; el valor predeterminado es `false`.

Valor devuelto

Si se ejecuta correctamente, devuelve un objeto ProjectItem; en caso contrario, devuelve `undefined`. Véase Objeto ProjectItem.

Descripción

Método; añade el archivo especificado al proyecto. Puede utilizar *`autoCreateFolder`* para determinar dónde debe situarse el nuevo archivo en el panel Proyecto:

- Si omite *`autoCreateFolder`* o transfiere un valor de `false`, el archivo se añadirá en el nivel de la raíz del proyecto.
- Si transfiere un valor de `true` para *`autoCreateFolder`* y *`fileURI`* se encuentra debajo del archivo FLP en la estructura de carpetas del disco, la estructura de carpetas de los archivos se reflejará en el panel Proyecto. Es decir, se añadirán nuevas carpetas al panel Proyecto si es necesario para reflejar la ubicación del archivo en el disco.
- Si transfiere un valor de `true` para *`autoCreateFolder`* y *`fileURI`* se encuentra sobre el archivo FLP en la estructura de carpetas del disco, el archivo se añadirá a nivel de la raíz. Es decir, se ignorará *`autoCreateFolder`*.

Ejemplo

El ejemplo siguiente ilustra varios modos de utilizar este comando. En este caso, el archivo de proyecto abierto se encuentra en el directorio c:\Projects, y los únicos archivos que hay actualmente en el proyecto se han añadido en el nivel de la raíz.

```
// Obtiene el objeto del proyecto
var myProject = fl.getProject();

// El comando siguiente crea una carpeta llamada "files" debajo del nivel de
  la raíz del proyecto, y sitúa myFile.fla en esa carpeta.
var newFile = myProject.addFile("file:///C|Projects/files/myFile.fla",
  true)
fl.trace(newFile.isMissing); // false

// Los dos comandos siguientes tienen el mismo efecto: colocan myFile_02.fla
  en el nivel de la raíz del proyecto.
var newFile = myProject.addFile("file:///C|Projects/files/myFile_02.fla" ,
  false)
var newFile = myProject.addFile("file:///C|Projects/files/myFile_02.fla")
fl.trace(newFile.isMissing); // false

// El comando siguiente coloca myFile_03 en el nivel de la raíz del proyecto
  como archivo que falta.
var newFile = myProject.addFile("file:///C|myFile_03.fla")
fl.trace(newFile.isMissing); // true
```

El ejemplo siguiente intenta añadir un archivo nuevo al proyecto y muestra un mensaje en el panel Salida que indica si se ha añadido el archivo.

```
var myProject = fl.getProject();
var newItem = myProject.addFile("file:///C|Projects/files/Integra.fla",
  true);
fl.trace( "Item " + ( newItem ? "was" : "was not" ) + " added!" );
```

Véase también

`fl.getProject()`, `project.items`, Objeto ProjectItem

project.canPublishProject()

Disponibilidad

Flash 8.

Uso

```
project.canPublishProject()
```

Parámetros

Ninguno.

Valor devuelto

Un valor booleano que especifica si el proyecto se puede publicar.

Descripción

Método; determina si el proyecto se puede publicar. Un proyecto se puede publicar si contiene al menos un archivo FLA.

Ejemplo

El ejemplo siguiente muestra un mensaje en el panel Salida si no se puede publicar el proyecto:

```
if (!fl.getProject().canPublishProject()) {
  fl.trace("Project cannot be published!");
}
```

Véase también

`fl.getProject()`, `project.publishProject()`, `projectItem.canPublish()`

project.canTestProject()

Disponibilidad

Flash 8.

Uso

```
project.canTestProject()
```

Parámetros

Ninguno.

Valor devuelto

Un valor booleano que especifica si el proyecto se puede probar.

Descripción

Método; determina si el proyecto se puede probar. Un proyecto se puede probar si se ha especificado un documento predeterminado.

Ejemplo

El ejemplo siguiente muestra un mensaje en el panel Salida si no se puede probar el proyecto:

```
if (!fl.getProject().canTestProject()) {
  fl.trace("Project cannot be tested!");
}
```

Véase también

`fl.getProject()`, `project.testProject()`, `projectItem.canTest()`

project.defaultItem

Disponibilidad

Flash 8.

Uso

```
project.defaultItem
```

Descripción

Propiedad; especifica el objeto ProjectItem que representa el documento predeterminado del proyecto. Debe especificar un elemento predeterminado si desea probar el proyecto. Véase Objeto ProjectItem.

Ejemplo

El ejemplo siguiente define el documento predeterminado del proyecto como el archivo Flower.fla:

```
var myProject = fl.getProject();
var item = myProject.findProjectItem("file:///C|/Projects/files/
  Flower.fla");
fl.myProject.defaultItem = item;
```

El ejemplo siguiente muestra el nombre del documento predeterminado en el panel Salida:

```
fl.trace(fl.getProject().defaultItem.displayName);
```

Véase también

`fl.getProject()`, `project.findProjectItem()`, Objeto ProjectItem

project.findProjectItem()

Disponibilidad

Flash 8.

Uso

```
project.findProjectItem( fileURI )
```

Parámetros

`fileURI` Una cadena que especifica el archivo que se va a buscar en el proyecto, expresada como archivo:/// URI.

Valor devuelto

Un objeto ProjectItem para el elemento si se ejecuta correctamente. En caso contrario, devuelve `false`. Véase Objeto ProjectItem.

Descripción

Método; busca un archivo especificado en el proyecto.

Ejemplo

El ejemplo siguiente muestra un mensaje de error en el panel Salida si no se encuentra un archivo especificado en el proyecto:

```
var myProject = fl.getProject();
var item = myProject.findProjectItem("file:///C|Projects/files/
  Integra.fla");
if (item == undefined) {
  fl.trace("Integra.fla is missing!");
}
```

Véase también

`fl.getProject()`, Objeto ProjectItem, `projectItem.isMissing`

project.items

Disponibilidad

Flash 8.

Uso

```
project.items
```

Descripción

Propiedad de sólo lectura; una matriz de objetos ProjectItem (véase Objeto ProjectItem) contenida en el proyecto.

Ejemplo

El ejemplo siguiente muestra los nombres de todos los elementos del proyecto en el panel Salida:

```
for (i = 0; i < fl.getProject().items.length; i++) {
  fl.trace(fl.getProject().items[i].displayName);
}
```

Véase también

`fl.getProject()`, Objeto ProjectItem

project.name

Disponibilidad

Flash 8.

Uso

```
project.name
```

Descripción

Propiedad; el nombre del proyecto que aparece en el panel Proyecto.

Ejemplo

El ejemplo siguiente especifica un nombre nuevo que se mostrará en el panel Proyecto:

```
fl.getProject().name = "New project name";
```

Véase también

`fl.getProject()`, `project.projectURI`

project.projectURI

Disponibilidad

Flash 8.

Uso

```
project.projectURI
```

Descripción

Propiedad de sólo lectura; una cadena que representa la ruta y el nombre del archivo de proyecto, expresada como archivo:/// URI.

Ejemplo

El ejemplo siguiente muestra la ruta y el nombre del archivo de proyecto abierto actualmente en el panel Salida:

```
fl.trace("Project is located at: " + fl.getProject().projectURI);
```

Véase también

`fl.getProject()`, `project.name`

project.publishProject()

Disponibilidad

Flash 8.

Uso

```
project.publishProject()
```

Parámetros

Ninguno.

Valor devuelto

Un valor booleano que indica si el proyecto se ha publicado correctamente.

Descripción

Método; publica los archivos FLA de un proyecto.

Ejemplo

El ejemplo siguiente publica el proyecto después de confirmar que se puede publicar y, a continuación, indica si el proyecto se ha publicado en el panel Salida:

```
if (fl.getProject().canPublishProject()) {
  var bSucceeded = fl.getProject().publishProject();
}
fl.trace(bSucceeded);
```

Véase también

`fl.getProject()`, `project.canPublishProject()`, `projectItem.publish()`

project.testProject()

Disponibilidad

Flash 8.

Uso

```
project.testProject()
```

Parámetros

Ninguno.

Valor devuelto

Un valor booleano que indica si el proyecto se ha probado correctamente.

Descripción

Método; prueba el proyecto. Un proyecto debe tener un documento predeterminado para probarse.

Ejemplo

El ejemplo siguiente prueba el proyecto después de confirmar que se puede probar y, a continuación, indica si el proyecto se ha probado en el panel Salida:

```
if (fl.getProject().canTestProject()) {
  var bSucceeded = fl.getProject().testProject();
}
fl.trace(bSucceeded);
```

Véase también

`fl.getProject()`, `project.canTestProject()`, `project.defaultItem`, `projectItem.test()`

Objeto ProjectItem

Disponibilidad

Flash 8.

Descripción

El objeto ProjectItem representa un elemento (archivo en el disco) que se ha añadido a un proyecto. Este objeto es una propiedad del objeto Project (véase `project.items`). Puede utilizar los comandos siguientes para devolver un objeto ProjectItem.

- Para añadir un archivo nuevo a un proyecto, utilice `project.addFile()`.
- Para localizar un elemento que ya se ha añadido a un proyecto, utilice `project.findProjectItem()`.

Resumen de métodos del objeto ProjectItem

Pueden emplearse los métodos siguientes con el objeto ProjectItem.

Método	Descripción
`projectItem.canPublish()`	Determina si un elemento del proyecto se puede publicar.
`projectItem.canTest()`	Determina si un elemento del proyecto se puede probar.
`projectItem.publish()`	Publica un elemento del proyecto.
`projectItem.test()`	Prueba un elemento del proyecto.

Resumen de propiedades del objeto ProjectItem

Pueden emplearse las propiedades siguientes con el objeto ProjectItem.

Propiedad	Descripción
`projectItem.displayName`	De sólo lectura; una cadena que especifica el nombre de un elemento de parámetro.
`projectItem.isMissing`	Sólo lectura; valor booleano que indica si falta un archivo del disco.
`projectItem.itemURI`	De sólo lectura; una cadena que especifica la ruta y el nombre de un elemento de proyecto.
`projectItem.publishProfile`	Una cadena que especifica el perfil de publicación que se empleará al publicar un elemento de proyecto (archivo FLA).

projectItem.canPublish()

Disponibilidad

Flash 8.

Uso

```
projectItem.canPublish()
```

Parámetros

Ninguno.

Valor devuelto

Un valor booleano que especifica si el elemento de proyecto se puede publicar.

Descripción

Método; determina si un elemento se puede publicar. Un elemento sólo se puede publicar si es un archivo FLA.

Ejemplo

El ejemplo siguiente muestra un mensaje en el panel Salida si no se puede publicar el primer elemento del proyecto.

```
var item = fl.getProject().items[0];
if (!item.canPublish()) {
  fl.trace(item.displayName + " cannot be published!");
}
```

Véase también

`fl.getProject()`, `project.canPublishProject()`, `project.items`, `projectItem.publish()`

projectItem.canTest()

Disponibilidad

Flash 8.

Uso

```
projectItem.canTest()
```

Parámetros

Ninguno.

Valor devuelto

Un valor booleano que especifica si el elemento de proyecto se puede probar.

Descripción

Método; determina si un elemento se puede probar. Un elemento se puede probar si es un archivo FLA o HTML.

Ejemplo

El ejemplo siguiente muestra un mensaje en el panel Salida si no se puede probar el primer elemento del proyecto.

```
var item = fl.getProject().items[0];
if (!item.canTest()) {
  fl.trace(item.name + " cannot be tested!");
}
```

Véase también

`fl.getProject()`, `project.canTestProject()`, `project.items`, `projectItem.test()`

projectItem.displayName

Disponibilidad

Flash 8.

Uso

```
projectItem.displayName
```

Descripción

Propiedad de sólo lectura; una cadena que especifica el nombre de un elemento del proyecto, como “file.fla”.

Ejemplo

El ejemplo siguiente muestra los nombres de todos los archivos del proyecto en el panel Salida.

```
fl.trace( "These are all the files in the project: ");
var files = fl.getProject().items;
for (i = 0; i < files.length; i++) {
  fl.trace(files[i].displayName + " ");
}
```

Véase también

`fl.getProject()`, `project.items`, `projectItem.itemURI`

projectItem.isMissing

Disponibilidad

Flash 8.

Uso

```
projectItem.isMissing
```

Descripción

Propiedad de sólo lectura; un valor booleano que indica si falta en el disco un archivo (por ejemplo, si el elemento se ha movido, eliminado o cambiado de nombre).

Ejemplo

El ejemplo siguiente muestra un mensaje en el panel Salida que indica si un archivo específico se encuentra en el disco en la carpeta prevista.

```
var item = fl.getProject().findProjectItem("file:///C|/Projects/files/
  DynamicHighAscii.fla");
fl.trace("DynamicHighAscii.fla is missing: " + item.isMissing);
```

Véase también

`fl.getProject()`, `project.findProjectItem()`, `project.items`

projectItem.itemURI

Disponibilidad

Flash 8.

Uso

```
projectItem.itemURI
```

Descripción

Propiedad de sólo lectura; una cadena, especificada como archivo:/// URI, que especifica la ruta y el nombre del elemento del proyecto. Los elementos de la carpeta contienen una cadena vacía ("").

Ejemplo

El ejemplo siguiente muestra la ruta y el nombre de cada elemento del proyecto en el panel Salida.

```
files = fl.getProject().items;
for (i = 0; i < files.length; i++) {
  fl.trace(files[i].itemURI);
}
```

Véase también

`fl.getProject()`, `projectItem.displayName`, `project.items`

projectItem.publish()

Disponibilidad

Flash 8.

Uso

```
projectItem.publish()
```

Parámetros

Ninguno.

Valor devuelto

Un valor booleano de `true` si es correcto, y de `false` en caso contrario.

Descripción

Método; publica un elemento del proyecto. Sólo se pueden publicar archivos FLA.

Ejemplo

El ejemplo siguiente publica todos los elementos publicables del proyecto.

```
for (var i in fl.getProject().items) {
  var item = fl.getProject().items[i];
  if (item.canPublish()) {
    item.publish();
  }
}
```

Véase también

`fl.getProject()`, `project.canPublishProject()`, `project.items`, `projectItem.canPublish()`, `projectItem.publishProfile`

projectItem.publishProfile

Disponibilidad

Flash 8.

Uso

```
projectItem.publishProfile
```

Descripción

Propiedad; una cadena que especifica el perfil de publicación que se empleará al publicar un elemento de proyecto (archivo FLA). El perfil de publicación debe ser un perfil existente en el elemento. En caso contrario, las llamadas posteriores a `projectItem.publish()` no tendrán éxito. Véase `projectItem.publish()`.

Si el elemento no es un archivo FLA, esta propiedad es una cadena vacía ("") y fallarán los intentos de definir esta propiedad.

Ejemplo

El ejemplo siguiente define el perfil de publicación de todos los elementos del proyecto con un perfil especificado que ya existe en el elemento y, a continuación, publica cada elemento. Si no existe el perfil en un archivo, el archivo no se publica.

```
var items = fl.getProject().items;
for ( i = 0 ; i < items.length ; i++ ) {
  items[i].publishProfile = "mySpecialProfile";
  items[i].publish();
}
```

Véase también

`fl.getProject()`, `project.canPublishProject()`, `project.items`, `projectItem.canPublish()`, `projectItem.publish()`

projectItem.test()

Disponibilidad

Flash 8.

Uso

```
projectItem.test()
```

Parámetros

Ninguno.

Valor devuelto

Un valor booleano que indica si el elemento se ha probado correctamente o no.

Descripciónn

Método; prueba un elemento del proyecto. Si la operación de prueba falla porque el elemento no es un archivo FLA o HTML, este método devuelve `false`.

Ejemplo

El ejemplo siguiente prueba todos los archivos FLA y HTML del proyecto:

```
for (var i in fl.getProject().items) {
  var item = fl.getProject().items[i];
  if (item.canTest()) {
    item.test();
  }
}
```

Véase también

`fl.getProject()`, `project.canTestProject()`, `project.items`, `projectItem.canTest()`

Objeto Screen

Disponibilidad

Flash MX 2004.

Descripción

El objeto Screen representa una pantalla única en un documento de diapositivas o formularios. Este objeto contiene propiedades relacionadas con la diapositiva o el formulario. Para acceder a la matriz de todos los objetos Screen del documento, utilice el código siguiente:

```
fl.getDocumentDOM().screenOutline.screens
```

Resumen de propiedades del objeto Screen

El objeto Screen tiene las propiedades siguientes:

Propiedades	Descripción
`screen.accName`	Una cadena que equivale al campo Nombre del panel Accesibilidad.
`screen.childScreens`	De sólo lectura; la matriz de pantallas secundarias de esta pantalla. La matriz está vacía si no hay pantallas secundarias.
`screen.description`	Una cadena que equivale al campo Descripción del panel Accesibilidad.
`screen.forceSimple`	Un valor booleano que activa o desactiva la accesibilidad para los elementos secundarios del objeto.
`screen.hidden`	Un valor booleano que especifica si la pantalla es visible.
`screen.instanceName`	De sólo lectura; una cadena que representa el nombre de instancia empleado para acceder al objeto desde ActionScript.
`screen.name`	De sólo lectura; una cadena que representa el nombre de la pantalla.
`screen.nextScreen`	De sólo lectura; un objeto que representa la siguiente pantalla del mismo nivel de la matriz `childScreens` principal.
`screen.parameters`	De sólo lectura; una matriz de propiedades de ActionScript 2.0 accesibles desde el inspector de propiedades de la pantalla.
`screen.parentScreen`	De sólo lectura; un objeto que representa la pantalla principal.
`screen.prevScreen`	De sólo lectura; un objeto que representa la pantalla anterior del mismo nivel de la matriz `childScreens` principal.
`screen.silent`	Un valor booleano que especifica si el objeto es accesible.

Propiedades	Descripción
screen.tabIndex	Propiedad; equivale al campo Índice de fichas del panel Accesibilidad.
screen.timeline	De sólo lectura; el objeto Timeline para la pantalla. Véase Objeto Timeline.

screen.accName

Disponibilidad

Flash MX 2004.

Uso

```
screen.accName
```

Descripción

Propiedad; una cadena que equivale al campo Nombre del panel Accesibilidad. Los lectores de pantalla identifican los objetos mediante la lectura del nombre en voz alta.

Ejemplo

El ejemplo siguiente almacena el valor del nombre del objeto en la variable `theName`:

```
var theName = fl.getDocumentDOM().screenOutline.screens[1].accName;
```

El ejemplo siguiente define el nombre del objeto como `"Home Button"`:

```
fl.getDocumentDOM().screenOutline.screens[1].accName = 'Home Button';
```

screen.childScreens

Disponibilidad

Flash MX 2004.

Uso

```
screen.childScreens
```

Descripción

Propiedad de sólo lectura; la matriz de pantallas secundarias de esta pantalla. La matriz está vacía si no hay pantallas secundarias.

Ejemplo

El ejemplo siguiente comprueba si el documento actual es una diapositiva o un formulario. Si lo es, almacena la matriz de pantallas secundarias en la variable `myChildren` y muestra sus nombres en el panel Salida:

```
var myChildren = new Array();
if(fl.getDocumentDOM().allowScreens) {
  var myParent = fl.getDocumentDOM().screenOutline.rootScreen.name
  for (i in fl.getDocumentDOM().screenOutline.rootScreen.childScreens) {
    myChildren.push("
  "+fl.getDocumentDOM().screenOutline.rootScreen.childScreens[i].name);
  }
  fl.trace(" The child screens of "+myParent+" are "+myChildren+". ");
}
```

screen.description

Disponibilidad

Flash MX 2004.

Uso

```
screen.description
```

Descripción

Propiedad; una cadena que equivale al campo Descripción del panel Accesibilidad. El lector de pantalla lee esta descripción.

Ejemplo

El ejemplo siguiente obtiene la descripción del objeto y la almacena en la variable `theDescription`:

```
var theDescription =
  fl.getDocumentDOM().screenOutline.screens[1].description;
```

El ejemplo siguiente define la descripción del objeto como `"This is Screen 1"`:

```
fl.getDocumentDOM().screenOutline.screens[1].description = "This is Screen
  1"
```

screen.forceSimple

Disponibilidad

Flash MX 2004.

Uso

```
screen.forceSimple
```

Descripción

Propiedad; un valor booleano que activa o desactiva la accesibilidad para los elementos secundarios del objeto. Equivale a la lógica inversa de la opción Hacer que los objetos secundarios sean accesibles del panel Accesibilidad. Es decir, si `forceSimple` es `true`, equivale a la opción desactivada Hacer que los objetos secundarios sean accesibles. Si `forceSimple` es `false`, equivale a la opción activada Hacer que los objetos secundarios sean accesibles.

Ejemplo

El ejemplo siguiente almacena el valor de `forceSimple` en la variable `areChildrenAccessible` (un valor de `false` significa que los elementos secundarios del objeto son accesibles):

```
var areChildrenAccessible =
  fl.getDocumentDOM().screenOutline.screens[1].forceSimple
```

El ejemplo siguiente hace que los elementos secundarios del objeto sean accesibles:

```
fl.getDocumentDOM().screenOutline.screens[1].forceSimple = false;
```

screen.hidden

Disponibilidad

Flash MX 2004.

Uso

```
screen.hidden
```

Descripción

Propiedad; un valor booleano que especifica si la pantalla es visible. Una pantalla con la propiedad `hidden` definida como `true` no es visible en ninguna otra pantalla.

Ejemplo

El ejemplo siguiente comprueba si la primera pantalla del contorno está oculta y cambia la visibilidad de la pantalla en consonancia. A continuación, un mensaje muestra en el panel Salida cuál era la visibilidad de la pantalla antes del cambio:

```
if (fl.getDocumentDOM().screenOutline.screens[0].hidden) {
  fl.getDocumentDOM().screenOutline.setScreenProperty("hidden", false);
  fl.trace(fl.getDocumentDOM().screenOutline.screens[0].name+" had its
  'hidden' property set to 'false'");
}
else {
  fl.getDocumentDOM().screenOutline.setScreenProperty("hidden", true);
  fl.trace(fl.getDocumentDOM().screenOutline.screens[0].name+" had its
  'hidden' property set to 'true'");
}
```

screen.instanceName

Disponibilidad

Flash MX 2004.

Uso

```
screen.instanceName
```

Descripción

Propiedad de sólo lectura; una cadena que representa el nombre de instancia empleado para acceder al objeto desde ActionScript.

Ejemplo

El ejemplo siguiente comprueba si el documento actual admite pantallas (porque es una diapositiva o un formulario). A continuación, asigna el valor `instanceName` de la primera pantalla secundaria de la matriz a la variable `myInstanceName` y abre el panel Salida para mostrar el nombre de instancia de la pantalla:

```
var myChildren = new Array();
if(fl.getDocumentDOM().allowScreens) {
  var myInstanceName =
  fl.getDocumentDOM().screenOutline.rootScreen.childScreens[0].instanceNam
  e;
  fl.trace(" The instanceName is "+myInstanceName+". ");
}
```

screen.name

Disponibilidad

Flash MX 2004.

Uso

```
screen.name
```

Descripción

Propiedad de sólo lectura; una cadena que representa el nombre de la pantalla.

Ejemplo

El ejemplo siguiente comprueba si el documento actual admite pantallas (porque es una diapositiva o un documento de formulario). A continuación, asigna el valor `name` de la primera pantalla secundaria de la matriz a la variable `myName` y abre el panel Salida para mostrar el nombre de la pantalla:

```
var myChildren = new Array();
if(fl.getDocumentDOM().allowScreens) {
  var myName =
  fl.getDocumentDOM().screenOutline.rootScreen.childScreens[0].name;
  fl.trace("The name of the screen is "+myName+". ");
}
```

screen.nextScreen

Disponibilidad

Flash MX 2004.

Uso

```
screen.nextScreen
```

Descripción

Propiedad de sólo lectura; un objeto que representa la siguiente pantalla del mismo nivel de la matriz `childScreens` principal. Es decir, `screen.NextScreen` se encuentra bajando en una matriz de pantallas secundarias a la siguiente pantalla de la matriz. Véase `screen.prevScreen`.

Si no hay una pantalla del mismo nivel, el valor es `null`.

Ejemplo

El ejemplo siguiente comprueba primero si el documento actual es una diapositiva o un formulario y, si lo es, recupera y muestra la secuencia de pantallas en el panel Salida:

```
if(fl.getDocumentDOM().allowScreens) {
  var myCurrent =
  fl.getDocumentDOM().screenOutline.rootScreen.childScreens[0].name;
  var myNext =
  fl.getDocumentDOM().screenOutline.rootScreen.childScreens[0].nextScreen.
  name;
  fl.trace(" The next screen to "+myCurrent+" is "+myNext+". ");
}
```

screen.parameters

Disponibilidad

Flash MX 2004.

Uso

```
screen.parameters
```

Descripción

Propiedad de sólo lectura; una matriz de propiedades de ActionScript 2.0 accesibles desde el inspector de propiedades de la pantalla.

Ejemplo

El ejemplo siguiente almacena los parámetros de la segunda pantalla del contorno en la variable `parms` y, a continuación, asigna el valor `"some value"` a la primera propiedad:

```
var parms = fl.getDocumentDOM().screenOutline.screens[1].parameters;
parms[0].value = "some value";
```

Véase también

Objeto Parameter

screen.parentScreen

Disponibilidad

Flash MX 2004.

Uso

```
screen.parentScreen
```

Descripción

Propiedad de sólo lectura; un objeto que representa la pantalla principal. Si `parentScreen` es `null`, la pantalla es una pantalla de nivel superior.

Ejemplo

El ejemplo siguiente almacena los valores de las propiedades `childScreens` y `parentScreen` en variables y, a continuación, muestra esos valores y sus relaciones superior/secundario en el panel Salida:

```
if(fl.getDocumentDOM().allowScreens) {
  var myCurrent =
  fl.getDocumentDOM().screenOutline.rootScreen.childScreens[1].name;
  var myParent =
  fl.getDocumentDOM().screenOutline.rootScreen.childScreens[1].parentScree
  n.name;
  fl.trace(" The parent screen to "+myCurrent+" is "+myParent+". ");
}
```

screen.prevScreen

Disponibilidad

Flash MX 2004.

Uso

```
screen.prevScreen
```

Descripción

Propiedad de sólo lectura; un objeto que representa la pantalla anterior del mismo nivel de la matriz `childScreens` principal. Si no hay una pantalla del mismo nivel, el valor es `null`. Véase también `screen.nextScreen`.

Ejemplo

El ejemplo siguiente comprueba si el documento actual es una diapositiva o un formulario y, si lo es, recupera y muestra la secuencia de pantallas en el panel Salida:

```
if(fl.getDocumentDOM().allowScreens) {
  var myCurrent =
  fl.getDocumentDOM().screenOutline.rootScreen.childScreens[1].name;
  var myNext =
  fl.getDocumentDOM().screenOutline.rootScreen.childScreens[1].prevScreen.
  name;
  fl.trace(" The previous screen to "+myCurrent+" is "+myNext+". ");
}
```

screen.silent

Disponibilidad

Flash MX 2004.

Uso

```
screen.silent
```

Descripción

Propiedad; un valor booleano que especifica si el objeto es accesible. Equivale a la lógica inversa de la opción Hacer que el objeto sea accesible del panel Accesibilidad. Es decir, si `silent` es `true`, equivale a la opción desactivada Hacer que los objetos secundarios sean accesibles en el panel Accesibilidad. Si `silent` es `false`, equivale a la opción activada Hacer que los objetos secundarios sean accesibles en el panel Accesibilidad.

Ejemplo

El ejemplo siguiente recupera el valor `silent` del objeto (un valor de `false` significa que el objeto es accesible):

```
var isSilent = fl.getDocumentDOM().screenOutline.screens[1].silent;
```

El ejemplo siguiente define el objeto como accesible:

```
fl.getDocumentDOM().screenOutline.screens[1].silent = false;
```

screen.tabIndex

Disponibilidad

Flash MX 2004.

Uso

```
screen.tabIndex
```

Descripción

Propiedad; equivale al campo Índice de fichas del panel Accesibilidad. Este valor permite determinar el orden de acceso a los objetos cuando el usuario presiona la tecla Tabulador.

Ejemplo

El ejemplo siguiente obtiene el índice de tabulación del objeto:

```
var theTabIndex = fl.getDocumentDOM().screenOutline.screens[1].tabIndex;
```

El ejemplo siguiente define el índice de tabulación del objeto como 1:

```
fl.getDocumentDOM().screenOutline.screens[1].tabIndex = 1;
```

screen.timeline

Disponibilidad

Flash MX 2004.

Uso

```
screen.timeline
```

Descripción

Propiedad de sólo lectura; el Objeto Timeline para la pantalla.

Ejemplo

El ejemplo siguiente obtiene la propiedad `screenOutline` del documento de diapositiva actual, asigna la matriz de propiedades `timeline` para la primera pantalla a `myArray` y muestra esas propiedades en el panel Salida:

```
myArray = new Array();
if(fl.getDocumentDOM().screenOutline) {
  for(i in fl.getDocumentDOM().screenOutline.screens[0].timeline) {
  myArray.push(" "+i+" :
  "+fl.getDocumentDOM().screenOutline.screens[0].timeline[i]+" ") ;
  }
  fl.trace("Here are the properties of the screen named "+
  fl.getDocumentDOM().screenOutline.screens[0].name+": "+myArray);
}
```

Objeto ScreenOutline

Disponibilidad

Flash MX 2004.

Descripción

El objeto ScreenOutline representa el grupo de pantallas en un documento de diapositivas o formularios. El acceso al objeto se realiza utilizando `fl.getDocumentDOM().screenOutline`.

El objeto ScreenOutline sólo existe si el documento es una diapositiva o un documento de formulario. Por tanto, antes de acceder a la propiedad, utilice `document.allowScreens()` para verificar que existe un documento de pantallas, como se muestra en el ejemplo siguiente:

```
if(fl.getDocumentDOM().allowScreens) {
   var myName =
     fl.getDocumentDOM().screenOutline.rootScreen.childScreens[0].name;
   fl.trace("The name of the screen is " + myName + ". ");
}
```

Resumen de métodos para el objeto ScreenOutline

Puede emplear los métodos siguientes con el objeto ScreenOutline:

Método	Descripción
`screenOutline.copyScreenFromFile()`	Inserta todas las pantallas o una pantalla con nombre y sus elementos secundarios, desde un documento especificado en la pantalla seleccionada actualmente.
`screenOutline.deleteScreen()`	Elimina la pantalla o pantallas seleccionadas actualmente, o una pantalla especificada, y los elementos secundarios de dichas pantallas.
`screenOutline.duplicateScreen()`	Duplica la pantalla o pantallas seleccionadas actualmente o una pantalla especificada.
`screenOutline.getSelectedScreens()`	Devuelve una matriz de objetos Screen que se encuentran seleccionados actualmente en el contorno de pantalla.
`screenOutline.insertNestedScreen()`	Inserta una pantalla anidada de un tipo específico en una determinada ubicación del contorno de pantalla.

Método	Descripción
`screenOutline.insertScreen()`	Inserta una pantalla nueva vacía de un tipo especificado en el documento en una ubicación determinada.
`screenOutline.moveScreen()`	Mueve la pantalla especificada en relación con el valor del parámetro `referenceScreen`; antes, después, como primer elemento secundario o como último elemento secundario.
`screenOutline.renameScreen()`	Cambia el nombre especificado de la pantalla.
`screenOutline.setCurrentScreen()`	Define la selección actual en el contorno de pantalla en la pantalla especificada.
`screenOutline.setScreenProperty()`	Permite la propiedad especificada con el valor especificado para las pantallas seleccionadas.
`screenOutline.setSelectedScreens()`	Selecciona las pantallas especificadas en el panel Contorno de pantalla.

Resumen de propiedades del objeto ScreenOutline

Puede emplear las propiedades siguientes con el objeto ScreenOutline:

Propiedad	Descripción
`screenOutline.currentScreen`	Un Objeto Screen; la pantalla seleccionada actualmente.
`screenOutline.rootScreen`	De sólo lectura; la primera pantalla del contorno de pantalla.
`screenOutline.screens`	De sólo lectura; la matriz de los objetos Screen de nivel superior que contiene el documento (véase Objeto Screen).

screenOutline.copyScreenFromFile()

Disponibilidad

Flash MX 2004.

Uso

```
screenOutline.copyScreenFromFile( fileURI [, screenName] )
```

Parámetros

fileURI Una cadena, expresada como archivo:/// URI, que especifica el nombre del archivo de edición que contiene las pantallas que se van a copiar en el documento.

screenName El nombre de la pantalla que se va a copiar. Si está presente el parámetro *screenName*, Flash copiará esa pantalla y sus elementos secundarios. Si no se especifica *screenName*, Flash copiará todo el documento. Este parámetro es opcional.

Valor devuelto

Ninguno. Si no se encuentra el archivo, no es un archivo FLA válido o no se encuentra la pantalla especificada, se presenta un error y se cancela el script.

Descripción

Método; inserta todas las pantallas o una pantalla con nombre y sus elementos secundarios, desde un documento especificado en la pantalla seleccionada actualmente. Si hay varias pantallas seleccionadas, se insertarán bajo la última pantalla seleccionada, como elementos del mismo nivel.

Ejemplo

El ejemplo siguiente copia la pantalla "slide1" desde el archivo myTarget.fla el escritorio hasta el documento actual (utilice su nombre de usuario en *userName*):

```
fl.getDocumentDOM().screenOutline.copyScreenFromFile("file:///C|/Documents
  and Settings/userName/Desktop/myTarget.fla", "slide1");
```

screenOutline.currentScreen

Disponibilidad

Flash MX 2004.

Uso

```
screenOutline.currentScreen
```

Descripción

Propiedad; un objeto Screen, la pantalla seleccionada actualmente (véase Objeto Screen).

Ejemplo

El ejemplo siguiente almacena el objeto `currentScreen` en la variable `myScreen` y, a continuación, muestra el nombre de esa pantalla en el panel Salida:

```
var myScreen = fl.getDocumentDOM().screenOutline.currentScreen;
fl.trace(myScreen.name);
```

screenOutline.deleteScreen()

Disponibilidad

Flash MX 2004.

Uso

```
screenOutline.deleteScreen( [screenName] )
```

Parámetros

`screenName` Una cadena que especifica el nombre de la pantalla que se va a eliminar. Si no transfiere un valor para *`screenName`*, se eliminarán la pantalla o pantallas seleccionadas actualmente y sus elementos secundarios. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; elimina la pantalla o pantallas seleccionadas actualmente, o una pantalla especificada, y los elementos secundarios de dichas pantallas.

Ejemplo

El ejemplo siguiente elimina la pantalla llamada apple y todos sus elementos secundarios:

```
fl.getDocumentDOM().screenOutline.deleteScreen("apple");
```

screenOutline.duplicateScreen()

Disponibilidad

Flash MX 2004.

Uso

```
screenOutline.duplicateScreen( [screenName] )
```

Parámetros

screenName Un valor de cadena que especifica el nombre de la pantalla que se va a duplicar. Si no transfiere un valor para *screenName*, se duplicarán la pantalla o pantallas seleccionadas actualmente. Este parámetro es opcional.

Valor devuelto

Un valor booleano: `true` si la pantalla se duplica correctamente, y `false` en caso contrario.

Descripción

Método; duplica la pantalla o pantallas seleccionadas actualmente o una pantalla especificada. Las pantallas duplicadas reciben un nombre predeterminado añadiendo _copy al nombre original, como por ejemplo, Screen_copy, Screen_copy2, etc. Si duplica varias pantallas, los duplicados se colocarán directamente bajo la pantalla seleccionada que se encuentre en el nivel más bajo de la jerarquía de contornos de pantallas.

Ejemplo

El ejemplo siguiente duplica una pantalla llamada apple:

```
fl.getDocumentDOM().screenOutline.duplicateScreen("apple");
```

screenOutline.getSelectedScreens()

Disponibilidad

Flash MX 2004.

Uso

```
screenOutline.getSelectedScreens()
```

Parámetros

Ninguno.

Valor devuelto

Una matriz de objetos Screen seleccionados (consulte Objeto Screen).

Descripción

Método; devuelve una matriz de objetos Screen que se encuentran seleccionados actualmente en el contorno de pantalla.

Ejemplo

El ejemplo siguiente almacena los objetos Screen seleccionados en la variable `myArray` y muestra los nombres de pantalla en el panel Salida:

```
var myArray = fl.getDocumentDOM().screenOutline.getSelectedScreens();
for (var i in myArray) {
  fl.trace(myArray[i].name)
}
```

screenOutline.insertNestedScreen()

Disponibilidad

Flash MX 2004.

Uso

```
screenOutline.insertNestedScreen( [ name [, referenceScreen [,
  screenTypeName ] ] ])
```

Parámetros

`name` Una cadena que indica el nombre de la nueva pantalla que se va a insertar. Un nombre vacío insertará una pantalla con un nombre de pantalla predeterminado, como Diapositiva *`n`* o Formulario *`n`* (donde *`n`* es el primer número exclusivo disponible). Este parámetro es opcional.

`referenceScreen` Una cadena que indica el nombre de la pantalla en la que se insertará la nueva pantalla como elemento secundario. Si se omite este parámetro, se insertará la nueva pantalla como elemento secundario de la pantalla seleccionada actualmente. Este parámetro es opcional.

`screenTypeName` Una cadena que especifica el tipo de pantalla que se va a asociar a la nueva pantalla anidada. Se definen el tipo de pantalla y el nombre de clase para esta pantalla. Los valores válidos son: `"Form"` y `"Slide"`. Este parámetro es opcional. Si se omite este parámetro, el tipo se heredará de la pantalla principal.

Valor devuelto

Un Objeto Screen.

Descripción

Método; inserta una pantalla anidada de un tipo específico en una determinada ubicación del contorno de pantalla.

Ejemplo

El ejemplo siguiente inserta slide2 como elemento secundario de slide1:

```
fl.getDocumentDOM().screenOutline.insertNestedScreen("slide2", "slide1",
  "Slide");
```

screenOutline.insertScreen()

Disponibilidad

Flash MX 2004.

Uso

```
screenOutline.insertScreen( [name [, referenceScreen [, screenTypeName ] ]
  ])
```

Parámetros

name Una cadena que indica el nombre de la nueva pantalla que se va a insertar. Si se omite este parámetro, el método inserta una pantalla con un nombre de pantalla predeterminado, como Diapositiva *n* o Formulario *n* (donde *n* es el primer número exclusivo disponible). Este parámetro es opcional.

referenceScreen Una cadena que indica el nombre de la pantalla anterior a la nueva pantalla. Si se omite este parámetro, la nueva pantalla se insertará después de la pantalla seleccionada actualmente. Si el parámetro *referenceScreen* identifica una pantalla secundaria, la nueva pantalla será un elemento del mismo nivel de la pantalla secundaria y una pantalla secundaria de la misma pantalla principal. Este parámetro es opcional.

screenTypeName Una cadena que especifica el tipo de pantalla que se va a asociar a la nueva pantalla. Se definen el tipo de pantalla y el nombre de clase para esta pantalla. Los valores válidos son: `"Form"` y `"Slide"`. Este parámetro es opcional.

Valor devuelto

Un Objeto Screen.

Descripción

Método; inserta una pantalla nueva vacía de un tipo especificado en el documento en una ubicación determinada.

Ejemplo

El ejemplo siguiente inserta un formulario llamado slide2 después de la pantalla llamada slide1:

```
fl.getDocumentDOM().screenOutline.insertScreen("slide2","slide1","Form");
```

El ejemplo siguiente inserta una diapositiva llamada slide4 después de la pantalla slide3:

```
fl.getDocumentDOM().screenOutline.insertScreen("slide4","slide3","Slide");
```

screenOutline.moveScreen()

Disponibilidad

Flash MX 2004.

Uso

```
screenOutline.moveScreen( screenToMove, referenceScreen, position )
```

Parámetros

screenToMove Una cadena que es el nombre de pantalla que se va a desplazar.

referenceScreen Una cadena que especifica la pantalla cerca de la cual se colocará *screenToMove*.

position Una cadena que especifica dónde se desplazará la pantalla en relación con *referenceScreen*. Los valores válidos son: `"before"`, `"after"`, `"firstChild"` y `"lastChild"`.

Valor devuelto

Un valor booleano: `true` si el desplazamiento es correcto; `false` en caso contrario.

Descripción

Método; mueve la pantalla especificada en relación con el valor del parámetro *referenceScreen*; antes, después, como primer elemento secundario o como último elemento secundario.

Ejemplo

El ejemplo siguiente desplaza la pantalla slide1 para que sea el primer elemento secundario de slide2:

```
fl.getDocumentDOM().screenOutline.moveScreen("slide1", "slide2",
  "firstChild");
```

screenOutline.renameScreen()

Disponibilidad

Flash MX 2004.

Uso

```
screenOutline.renameScreen( newScreenName [, oldScreenName [,
  bDisplayError] ] )
```

Parámetros

newScreenName Una cadena que especifica el nuevo nombre de la pantalla.

oldScreenName Una cadena que especifica el nombre de la pantalla existente que se va a cambiar. Si no se especifica, cambiará el nombre de la pantalla seleccionada actualmente. Este parámetro es opcional.

bDisplayError Un valor booleano que, si se define como `true`, muestra un mensaje de error si se produce un error, por ejemplo, si ya existe una pantalla con el mismo nombre que el valor transferido a *newScreenName*. El valor predeterminado es `false`.

Valor devuelto

Un valor booleano: `true` si el cambio de nombre es correcto; `false` en caso contrario.

Descripción

Método; cambia el nombre especificado de la pantalla.

Ejemplo

El ejemplo siguiente cambia el nombre de slide1 a Intro:

```
fl.getDocumentDOM().screenOutline.renameScreen("Intro", "slide1");
```

screenOutline.rootScreen

Disponibilidad

Flash MX 2004.

Uso

```
screenOutline.rootScreen
```

Descripción

Propiedad de sólo lectura; la primera pantalla del contorno de pantalla. Puede utilizar `screenOutline.rootScreen` como método abreviado de `screenOutline.screens[0]`.

Ejemplo

El ejemplo siguiente muestra el nombre del primer elemento secundario de la primera pantalla en el contorno de pantalla:

```
fl.trace(fl.getDocumentDOM().screenOutline.rootScreen.childScreens[0].name)
  ;
```

screenOutline.screens

Disponibilidad

Flash MX 2004.

Uso

```
screenOutline.screens
```

Descripción

Propiedad de sólo lectura; la matriz de los objetos Screen de nivel superior que contiene el documento (consulte Objeto Screen).

Ejemplo

El ejemplo siguiente almacena la matriz de objetos Screen en la variable `myArray` y, a continuación, muestra sus nombres en el panel Salida:

```
var myArray = new Array();
if(fl.getDocumentDOM().allowScreens) {
  for(var i in fl.getDocumentDOM().screenOutline.screens) {
     myArray.push(" "+fl.getDocumentDOM().screenOutline.screens[i].name);
  }
  fl.trace("The screens array contains objects whose names are:
  "+myArray+". ");
}
```

screenOutline.setCurrentScreen()

Disponibilidad

Flash MX 2004.

Uso

```
screenOutline.setCurrentScreen( name )
```

Parámetros

name Una cadena que especifica el nombre de la pantalla que debe convertirse en la pantalla seleccionada actualmente. Si la pantalla es un elemento secundario de otra pantalla, no será necesario que indique una ruta o una jerarquía.

Valor devuelto

Ninguno.

Descripción

Método; define la selección actual en el contorno de pantalla en la pantalla especificada.

Ejemplo

El ejemplo siguiente define la pantalla actual en la pantalla llamada ChildOfSlide_1:

```
fl.getDocumentDOM().screenOutline.setCurrentScreen("ChildOfSlide_1");
```

screenOutline.setScreenProperty()

Disponibilidad

Flash MX 2004.

Uso

```
screenOutline.setScreenProperty( property, value )
```

Parámetros

property Una cadena que especifica la propiedad que se va a definir.

value El nuevo valor de la propiedad. El tipo de valor depende de la propiedad que se está definiendo.

Para ver una lista de propiedades y valores, consulte Resumen de propiedades del objeto Screen.

Valor devuelto

Ninguno.

Descripción

Método; define la propiedad especificada con el valor especificado para las pantallas seleccionadas.

Ejemplo

El ejemplo siguiente cambia la visibilidad de las pantallas seleccionadas actualmente de oculta a visible:

```
fl.getDocumentDOM().screenOutline.setScreenProperty("hidden", false);
```

screenOutline.setSelectedScreens()

Disponibilidad

Flash MX 2004.

Uso

```
screenOutline.setSelectedScreens ( selection [, bReplaceCurrentSelection ]
  )
```

Parámetros

selection Una matriz de nombres de pantalla que se seleccionarán en el contorno de pantalla.

bReplaceCurrentSelection Un valor booleano que, si es `true`, permite anular la selección actual. El valor predeterminado es `true`. Si es `false`, Flash extiende la selección actual para incluir las pantallas especificadas. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; selecciona las pantallas especificadas en el contorno de pantalla. Si se especifican múltiples pantallas, se seleccionará en el escenario la pantalla con el último valor de índice de la matriz de selección.

Ejemplo

El ejemplo siguiente anula la selección de las pantallas seleccionadas actualmente y, a continuación, selecciona las pantallas `slide1`, `slide2`, `slide3` y slide4 en el contorno de pantalla:

```
myArray = new Array("slide1", "slide2", "slide3", "slide4");
fl.getDocumentDOM().screenOutline.setSelectedScreens(myArray, true);
```

Objeto Shape

Herencia Objeto Element > Objeto Shape

Disponibilidad

Flash MX 2004.

Descripción

El objeto Shape es una subclase del objeto Element. El objeto Shape proporciona un control más preciso que las API de dibujo a la hora de manipular o crear geometría en el escenario. Este control es necesario para que los scripts pueden crear efectos útiles y otros comandos de dibujo. (Véase Objeto Element.)

Todos los métodos y las propiedades de Shape que pueden modificar una forma o cualquiera de sus partes subordinadas deben colocarse entre llamadas `shape.beginEdit()` y `shape.endEdit()` para que funcionen correctamente.

Resumen de métodos del objeto Shape

Además de los métodos del Objeto Element, puede emplear los métodos siguientes con el objeto Shape:

Método	Descripción
`shape.beginEdit()`	Define el comienzo de una sesión de edición.
`shape.deleteEdge()`	Elimina el borde especificado.
`shape.endEdit()`	Define el final de una sesión de edición para la forma.

Resumen de propiedades del objeto Shape

Además de las propiedades del Objeto Element, el objeto Shape dispone de las siguientes:

Propiedad	Descripción
`shape.contours`	De sólo lectura; una matriz de objetos Contour para la forma (consulte Objeto Contour).
`shape.edges`	De sólo lectura; una matriz de objetos Edge (consulte Objeto Edge).
`shape.isDrawingObject`	De sólo lectura; si es `true`, la forma será un objeto de dibujo.
`shape.isGroup`	De sólo lectura; si es `true`, la forma será un grupo.
`shape.vertices`	De sólo lectura; una matriz de objetos Vertex (consulte Objeto Vertex).

shape.beginEdit()

Disponibilidad

Flash MX 2004.

Uso

```
shape.beginEdit()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; define el comienzo de una sesión de edición. Deberá utilizar este método antes de ejecutar comandos que cambien el objeto Shape o cualquiera de sus partes subordinadas.

Ejemplo

El ejemplo siguiente toma la forma seleccionada actualmente y le quita el primer borde de la matriz de bordes:

```
var shape = fl.getDocumentDOM().selection[0];
shape.beginEdit();
shape.deleteEdge(0);
shape.endEdit();
```

shape.contours

Disponibilidad

Flash MX 2004.

Uso

```
shape.contours
```

Descripción

Propiedad de sólo lectura; una matriz de objetos Contour para la forma (consulte Objeto Contour).

Ejemplo

El ejemplo siguiente almacena el primer contorno de la matriz de contornos en la variable *c* y, a continuación, almacena el Objeto HalfEdge de ese contorno en la variable `he`:

```
var c = fl.getDocumentDOM().selection[0].contours[0];
var he = c.getHalfEdge();
```

shape.deleteEdge()

Disponibilidad

Flash MX 2004.

Uso

```
shape.deleteEdge( index )
```

Parámetros

index Un índice basado en cero que especifica el borde que se va a eliminar de la matriz `shape.edges`. Este método cambia la longitud de la matriz `shape.edges`.

Valor devuelto

Ninguno.

Descripción

Método; elimina el borde especificado. Deberá llamar a `shape.beginEdit()` antes de utilizar este método.

Ejemplo

El ejemplo siguiente toma la forma seleccionada actualmente y quita el primer borde de la matriz de bordes:

```
var shape = fl.getDocumentDOM().selection[0];
shape.beginEdit();
shape.deleteEdge(0);
shape.endEdit();
```

shape.edges

Disponibilidad

Flash MX 2004.

Uso

```
shape.edges
```

Descripción

Propiedad de sólo lectura; una matriz de objetos Edge (consulte Objeto Edge).

shape.endEdit()

Disponibilidad

Flash MX 2004.

Uso

```
shape.endEdit()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; define el final de una sesión de edición para la forma. Todos los cambios realizados en el objeto Shape o en cualquiera de sus partes subordinadas se aplicarán a la forma. Deberá utilizar este método después de ejecutar comandos que cambien el objeto Shape o cualquiera de sus partes subordinadas.

Ejemplo

El ejemplo siguiente toma la forma seleccionada actualmente y le quita el primer borde de la matriz de bordes:

```
var shape = fl.getDocumentDOM().selection[0];
shape.beginEdit();
shape.deleteEdge(0);
shape.endEdit();
```

shape.isDrawingObject

Disponibilidad

Flash 8.

Uso

```
shape.isDrawingObject
```

Descripción

Propiedad de sólo lectura; si es `true`, la forma será un objeto de dibujo.

Ejemplo

El ejemplo siguiente almacena el primer objeto seleccionado `item` en la variable `sel` y, a continuación, utiliza las propiedades `element.elementType` y `shape.isDrawingObject` para determinar si el elemento seleccionado es un objeto de dibujo.

```
var sel = fl.getDocumentDOM().selection[0];
var shapeDrawingObject = (sel.elementType == "shape") &&
  sel.isDrawingObject;
fl.trace(shapeDrawingObject);
```

Véase también

`document.crop()`, `document.deleteEnvelope()`, `document.intersect()`, `document.punch()`, `document.union()`, `shape.isGroup`

shape.isGroup

Disponibilidad

Flash MX 2004.

Uso

```
shape.isGroup
```

Descripción

Propiedad de sólo lectura; si es `true`, la forma será un grupo.

Ejemplo

El ejemplo siguiente almacena el primer objeto seleccionado `item` en la variable `sel` y, a continuación, utiliza las propiedades `element.elementType` y `shape.isGroup` para determinar si el elemento seleccionado es un grupo:

```
var sel = fl.getDocumentDOM().selection[0];
var shapeGroup = (sel.elementType == "shape") && sel.isGroup;
fl.trace(shapeGroup);
```

Véase también

`shape.isDrawingObject`

shape.vertices

Disponibilidad

Flash MX 2004.

Uso

```
shape.vertices
```

Descripción

Propiedad de sólo lectura; una matriz de objetos Vertex (consulte Objeto Vertex).

Ejemplo

El ejemplo siguiente almacena el primer objeto seleccionado `item` en la variable `someShape` y, a continuación, muestra el número de vértices de ese objeto en el panel Salida:

```
var someShape = fl.getDocumentDOM().selection[0];
fl.trace("The shape has " + someShape.vertices.length + " vertices.");
```

Objeto SoundItem

Herencia Objeto Item > Objeto SoundItem

Disponibilidad

Flash MX 2004.

Descripción

El objeto SoundItem es una subclase del objeto Item. Representa un elemento de biblioteca empleado para crear un sonido. Véase también `frame.soundLibraryItem` y Objeto Item.

Resumen de propiedades del objeto SoundItem

Además de las propiedades del Objeto Item, el objeto SoundItem dispone de las siguientes:

Propiedad	Descripción
`soundItem.bitRate`	Una cadena que especifica la velocidad de transmisión de un sonido de la biblioteca. Sólo está disponible para el tipo de compresión MP3.
`soundItem.bits`	Una cadena que especifica el valor en bits de un sonido de la biblioteca con compresión ADPCM.
`soundItem.compressionType`	Una cadena que especifica el tipo de compresión de un sonido de la biblioteca.
`soundItem.convertStereoToMono`	Un valor booleano disponible sólo para tipos de compresión MP3 y Raw.
`soundItem.quality`	Una cadena que especifica la calidad de reproducción de un sonido de la biblioteca. Sólo está disponible para el tipo de compresión MP3.
`soundItem.sampleRate`	Una cadena que especifica la velocidad de muestreo del clip de audio.
`soundItem.useImportedMP3Quality`	Un valor booleano; si es `true`, se ignorarán todas las demás propiedades y se utilizará la calidad del MP3 importado.

soundItem.bitRate

Disponibilidad

Flash MX 2004.

Uso

```
soundItem.bitRate
```

Descripción

Propiedad; una cadena que especifica la velocidad de transmisión de un sonido de la biblioteca. Esta propiedad sólo está disponible para el tipo de compresión MP3. Los valores aceptables son: `"8 kbps"`, `"16 kbps"`, `"20 kbps"`, `"24 kbps"`, `"32 kbps"`, `"48 kbps"`, `"56 kbps"`, `"64 kbps"`, `"80 kbps"`, `"112 kbps"`, `"128 kbps"` y `"160 kbps"`. Los sonidos estéreo exportados a 8 ó 16 kbps se convierten en mono. La propiedad es `undefined` para otros tipos de compresión.

NOTA	Si desea especificar un valor para esta propiedad, defina `soundItem.useImportedMP3Quality` como `false`.

Ejemplo

El ejemplo siguiente muestra el valor `bitRate` en el panel Salida si el elemento especificado en la biblioteca tiene compresión MP3:

```
alert(fl.getDocumentDOM().library.items[0].bitRate);
```

Véase también

`soundItem.compressionType`, `soundItem.convertStereoToMono`

soundItem.bits

Disponibilidad

Flash MX 2004.

Uso

```
soundItem.bits
```

Descripción

Propiedad; una cadena que especifica el valor en bits de un sonido de la biblioteca con compresión ADPCM. Los valores válidos son: `"2 bit"`, `"3 bit"`, `"4 bit"` y `"5 bit"`.

NOTA	Si desea especificar un valor para esta propiedad, defina `soundItem.useImportedMP3Quality` como `false`.

Ejemplo

El ejemplo siguiente muestra el valor en bits en el panel Salida si el elemento seleccionado actualmente en la biblioteca tiene compresión ADPCM:

```
alert(fl.getDocumentDOM().library.items[0].bits);
```

Véase también

`soundItem.compressionType`

soundItem.compressionType

Disponibilidad

Flash MX 2004.

Uso

```
soundItem.compressionType
```

Descripción

Propiedad; una cadena que especifica el tipo de compresión de un sonido de la biblioteca. Los valores válidos son: `"Default"`, `"ADPCM"`, `"MP3"`, `"Raw"` y `"Speech"`.

NOTA	Si desea especificar un valor para esta propiedad, defina `soundItem.useImportedMP3Quality` como `false`.

Ejemplo

El ejemplo siguiente cambia un elemento de la biblioteca al tipo de compresión Raw.

```
fl.getDocumentDOM().library.items[0].compressionType = "Raw";
```

El ejemplo siguiente cambia el tipo de compresión de un elemento seleccionado a Speech:

```
fl.getDocumentDOM().library.getSelectedItems()[0].compressionType =
  "Speech";
```

soundItem.convertStereoToMono

Disponibilidad

Flash MX 2004.

Uso

```
soundItem.convertStereoToMono
```

Descripción

Propiedad; un valor booleano disponible sólo para tipos de compresión MP3 y Raw. Si se define como `true`, se convertirá un sonido estéreo en mono; `false` lo dejará como estéreo. Para el tipo de compresión MP3, si `soundItem.bitRate` es menor que 20 Kbps, se ignorará esta propiedad y se impondrá como `true` (consulte `soundItem.bitRate`).

NOTA	Si desea especificar un valor para esta propiedad, defina `soundItem.useImportedMP3Quality` como `false`.

Ejemplo

El ejemplo siguiente convierte un elemento de la biblioteca en mono sólo si el elemento tiene el tipo de compresión MP3 o Raw:

```
fl.getDocumentDOM().library.items[0].convertStereoToMono = true;
```

Véase también

`soundItem.compressionType`

soundItem.quality

Disponibilidad

Flash MX 2004.

Uso

```
soundItem.quality
```

Descripción

Propiedad; una cadena que especifica la calidad de reproducción de un sonido de la biblioteca. Esta propiedad sólo está disponible para el tipo de compresión MP3. Los valores válidos son: `"Fast"`, `"Medium"` y `"Best"`.

NOTA	Si desea especificar un valor para esta propiedad, defina `soundItem.useImportedMP3Quality` como `false`.

Ejemplo

El ejemplo siguiente define la calidad de reproducción de un elemento de la biblioteca como Best si dicha elemento tiene el tipo de compresión MP3:

```
fl.getDocumentDOM().library.items[0].quality = "Best";
```

Véase también

`soundItem.compressionType`

soundItem.sampleRate

Disponibilidad

Flash MX 2004.

Uso

```
soundItem.sampleRate
```

Descripción

Propiedad; una cadena que especifica la velocidad de muestreo del clip de audio. Esta propiedad sólo está disponible para los tipos de compresión ADPCM, Raw y Speech. Los valores válidos son: `"5 kHz"`, `"11 kHz"`, `"22 kHz"` y `"44 kHz"`.

NOTA	Si desea especificar un valor para esta propiedad, defina `soundItem.useImportedMP3Quality` como `false`.

Ejemplo

El ejemplo siguiente define la velocidad de muestreo de un elemento de la biblioteca como 5 kHz, si el elemento tiene compresión ADPCM, Raw o Speech.

```
fl.getDocumentDOM().library.items[0].sampleRate = "5 kHz";
```

Véase también

`soundItem.compressionType`

soundItem.useImportedMP3Quality

Disponibilidad

Flash MX 2004.

Uso

```
soundItem.useImportedMP3Quality
```

Descripción

Propiedad; un valor booleano. Si es `true`, se ignorarán todas las demás propiedades y se utilizará la calidad del MP3 importado.

Ejemplo

El ejemplo siguiente define un elemento de la biblioteca para utilizar la calidad del MP3 importado:

```
fl.getDocumentDOM().library.items[0].useImportedMP3Quality = true;
```

Véase también

`soundItem.compressionType`

Objeto Stroke

Disponibilidad

Flash MX 2004.

Descripción

El objeto Stroke contiene toda la configuración de un trazo, incluida la configuración personalizada. Este objeto representa la información que contiene el inspector de propiedades. Utilizando el objeto Stroke con el método `document.setCustomStroke()` puede cambiar la configuración de trazo para el panel Herramientas, el inspector de propiedades y la selección actual. También puede obtener la configuración de trazo del panel Herramientas y del inspector de propiedades, o de la selección actual, empleando el método `document.getCustomStroke()`.

Este objeto siempre tiene las cuatro propiedades siguientes: `style`, `thickness`, `color` y `breakAtCorners`. Se pueden definir otras propiedades, según el valor de la propiedad `stroke.style`.

Resumen de propiedades del objeto Stroke

Las propiedades siguientes están disponibles para el objeto Stroke:

Propiedad	Descripción
`stroke.breakAtCorners`	Equivale a la opción Esquinas Marcadas del cuadro de diálogo Estilo del Trazo personalizado.
`stroke.capType`	Una cadena que especifica el tipo de extremo del trazo.
`stroke.color`	Una cadena, valor hexadecimal o entero que representa el color de trazo.
`stroke.curve`	Una cadena que especifica el tipo de trama del trazo.
`stroke.dash1`	Un entero que especifica las longitudes de la parte sólida de una línea discontinua.
`stroke.dash2`	Un entero que especifica las longitudes de la parte en blanco de una línea discontinua.
`stroke.density`	Una cadena que especifica la densidad de una línea punteada.
`stroke.dotSize`	Una cadena que especifica el tamaño de punto de una línea punteada.
`stroke.dotSpace`	Un entero que especifica el espaciado entre puntos en una línea de puntos.

Propiedad	Descripción
stroke.hatchThickness	Una cadena que especifica el grosor de una línea de sombreado.
stroke.jiggle	Una cadena que especifica la propiedad de vaivén de una línea de sombreado.
stroke.joinType	Una cadena que especifica el tipo de unión del trazo.
stroke.length	Una cadena que especifica la longitud de una línea de sombreado.
stroke.miterLimit	Un valor flotante que especifica el ángulo sobre el cual se truncará la punta del angular en un segmento.
stroke.pattern	Una cadena que especifica el patrón de una línea no justificada.
stroke.rotate	Una cadena que especifica la rotación de una línea de sombreado.
stroke.scaleType	Una cadena que especifica el tipo de escala que se aplicará al trazo.
stroke.shapeFill	Un Objeto Fill que representa la configuración de relleno del trazo.
stroke.space	Una cadena que especifica el espaciado de una línea de sombreado.
stroke.strokeHinting	Un valor booleano que especifica si se definen sugerencias en el trazo.
stroke.style	Una cadena que describe el estilo de trazo.
stroke.thickness	Un entero que especifica el tamaño del trazo.
stroke.variation	Una cadena que especifica la variación de una línea punteada.
stroke.waveHeight	Una cadena que especifica la altura de onda de una línea no justificada.
stroke.waveLength	Una cadena que especifica la longitud de onda de una línea no justificada.

stroke.breakAtCorners

Disponibilidad

Flash MX 2004.

Uso

```
stroke.breakAtCorners
```

Descripción

Propiedad; un valor booleano. Esta propiedad equivale a la opción Esquinas Marcadas del cuadro de diálogo Estilo del Trazo personalizado.

Ejemplo

El ejemplo siguiente define la propiedad `breakAtCorners` como `true`:

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.breakAtCorners = true;
fl.getDocumentDOM().setCustomStroke( myStroke );
```

stroke.capType

Disponibilidad

Flash 8.

Uso

```
stroke.capType
```

Descripción

Propiedad; una cadena que especifica el tipo de extremo del trazo. Los valores válidos son: `"none"`, `"round"` y `"square"`.

Ejemplo

El ejemplo siguiente define el tipo de extremo del trazo como `"round"`:

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.capType = "round";
fl.getDocumentDOM().setCustomStroke(myStroke);
```

stroke.color

Disponibilidad

Flash MX 2004.

Uso

```
stroke.color
```

Descripción

Propiedad; el color del trazo, en uno de los formatos siguientes:

- Una cadena con el formato `"#RRGGBB"` o `"#RRGGBBAA"`
- Un número hexadecimal con el formato `0xRRGGBB`
- Un entero que representa el equivalente decimal del número hexadecimal

Ejemplo

El ejemplo siguiente define el color del trazo:

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.color = "#000000";
fl.getDocumentDOM().setCustomStroke( myStroke );
```

stroke.curve

Disponibilidad

Flash MX 2004.

Uso

```
stroke.curve
```

Descripción

Propiedad; una cadena que especifica el tipo de trama del trazo. Esta propiedad sólo se puede definir si la propiedad `stroke.style` es `"hatched"` (véase `stroke.style`). Los valores válidos son: `"straight"`, `"slight curve"`, `"medium curve"` y `"very curved"`.

Ejemplo

El ejemplo siguiente define la propiedad de la curva, entre otras, para un trazo que tiene el estilo `"hatched"`:

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "hatched";
myStroke.curve = "straight";
myStroke.space = "close";
myStroke.jiggle = "wild";
myStroke.rotate = "free";
myStroke.length = "slight";
myStroke.hatchThickness = "thin";
fl.getDocumentDOM().setCustomStroke( myStroke );
```

stroke.dash1

Disponibilidad

Flash MX 2004.

Uso

```
stroke.dash1
```

Descripción

Propiedad; un entero que especifica las longitudes de las partes sólidas de una línea discontinua. Esta propiedad sólo está disponible si la propiedad `stroke.style` se define como `"dashed"` (véase `stroke.style`).

Ejemplo

El ejemplo siguiente define las propiedades `dash1` y `dash2` para un estilo de trazo de `dashed`:

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "dashed";
myStroke.dash1 = 1;
myStroke.dash2 = 2;
fl.getDocumentDOM().setCustomStroke( myStroke );
```

stroke.dash2

Disponibilidad

Flash MX 2004.

Uso

```
stroke.dash2
```

Descripción

Propiedad; un entero que especifica las longitudes de las partes en blanco de una línea discontinua. Esta propiedad sólo está disponible si la propiedad `stroke.style` se define como `"dashed"` (véase `stroke.style`).

Ejemplo

Véase `stroke.dash1`.

stroke.density

Disponibilidad

Flash MX 2004.

Uso

`stroke.density`

Descripción

Propiedad; una cadena que especifica la densidad de una línea punteada. Esta propiedad sólo está disponible si la propiedad `stroke.style` se define como `"stipple"` (véase `stroke.style`). Los valores válidos son: `"very dense"`, `"dense"`, `"sparse"` y `"very sparse"`.

Ejemplo

El ejemplo siguiente define la propiedad de densidad como `"sparse"` para el estilo de trazo de `stipple`:

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "stipple";
myStroke.dotSpace= 3;
myStroke.variation = "random sizes";
myStroke.density = "sparse";
fl.getDocumentDOM().setCustomStroke( myStroke );
```

stroke.dotSize

Disponibilidad

Flash MX 2004.

Uso

`stroke.dotSize`

Descripción

Propiedad; una cadena que especifica el tamaño de punto de una línea punteada. Esta propiedad sólo está disponible si la propiedad `stroke.style` se define como `"stipple"` (véase `stroke.style`). Los valores válidos son: `"tiny"`, `"small"`, `"medium"` y `"large"`.

El ejemplo siguiente define la propiedad `dotsize` como `"tiny"` para el estilo de trazo de `stipple`:

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "stipple";
myStroke.dotSpace= 3;
myStroke.dotsize = "tiny";
myStroke.variation = "random sizes";
myStroke.density = "sparse";
fl.getDocumentDOM().setCustomStroke( myStroke );
```

stroke.dotSpace

Disponibilidad

Flash MX 2004.

Uso

```
stroke.dotSpace
```

Descripción

Propiedad; un entero que especifica el espaciado entre puntos en una línea de puntos. Esta propiedad sólo está disponible si la propiedad `stroke.style` se define como `"dotted"`. Véase `stroke.style`.

Ejemplo

El ejemplo siguiente define la propiedad `dotSpace` como `3` para un estilo de trazo de `dotted`:

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "dotted";
myStroke.dotSpace= 3;
fl.getDocumentDOM().setCustomStroke( myStroke );
```

stroke.hatchThickness

Disponibilidad

Flash MX 2004.

Uso

```
stroke.hatchThickness
```

Descripción

Propiedad; una cadena que especifica el grosor de una línea de sombreado. Esta propiedad sólo está disponible si la propiedad `stroke.style` se define como `"hatched"` (véase `stroke.style`). Los valores válidos son: `"hairline"`, `"thin"`, `"medium"` y `"thick"`.

Ejemplo

El ejemplo siguiente define la propiedad `hatchThickness` como `"thin"` para un estilo de trazo de `hatched`:

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "hatched";
myStroke.curve = "straight";
myStroke.space = "close";
myStroke.jiggle = "wild";
myStroke.rotate = "free";
myStroke.length = "slight";
myStroke.hatchThickness = "thin";
fl.getDocumentDOM().setCustomStroke( myStroke );
```

stroke.jiggle

Disponibilidad

Flash MX 2004.

Uso

```
stroke.jiggle
```

Descripción

Propiedad; una cadena que especifica la propiedad de vaivén de una línea de sombreado. Esta propiedad sólo está disponible si la propiedad `stroke.style` se define como `"hatched"` (véase `stroke.style`). Los valores válidos son: `"none"`, `"bounce"`, `"loose"` y `"wild"`.

Ejemplo

El ejemplo siguiente define la propiedad `jiggle` como `"wild"` para un estilo de trazo de `hatched`:

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "hatched";
myStroke.curve = "straight";
myStroke.space = "close";
myStroke.jiggle = "wild";
myStroke.rotate = "free";
myStroke.length = "slight";
myStroke.hatchThickness = "thin";
fl.getDocumentDOM().setCustomStroke( myStroke );
```

stroke.joinType

Disponibilidad

Flash 8.

Uso

```
stroke.joinType
```

Descripción

Propiedad; una cadena que especifica el tipo de unión del trazo. Los valores aceptables son: `"miter"`, `"round"` y `"bevel"`.

Véase también

`stroke.capType`

stroke.length

Disponibilidad

Flash MX 2004.

Uso

```
stroke.length
```

Descripción

Propiedad; una cadena que especifica la longitud de una línea de sombreado. Esta propiedad sólo está disponible si la propiedad `stroke.style` se define como `"hatched"` (véase `stroke.style`). Los valores válidos son: `"equal"`, `"slight"`, `"variation"`, `"medium variation"` y `"random"`.

Ejemplo

El ejemplo siguiente define la propiedad `length` como `"slight"` para un estilo de trazo de `hatched`:

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "hatched";
myStroke.curve = "straight";
myStroke.space = "close";
myStroke.jiggle = "wild";
myStroke.rotate = "free";
myStroke.length = "slight";
myStroke.hatchThickness = "thin";
fl.getDocumentDOM().setCustomStroke( myStroke );
```

stroke.miterLimit

Disponibilidad

Flash 8.

Uso

```
stroke.miterLimit
```

Descripción

Propiedad; un valor flotante que especifica el ángulo sobre el cual se truncará la punta del angular en un segmento. Esto significa que el angular sólo se trunca si su ángulo es mayor que el valor de `miterLimit`.

Ejemplo

El ejemplo siguiente cambia el límite del angular del trazo a 3. Si su ángulo es mayor que 3, se truncará el angular.

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.miterLimit = 3;
var myStroke = fl.getDocumentDOM().setCustomStroke();
```

stroke.pattern

Disponibilidad

Flash MX 2004.

Uso

```
stroke.pattern
```

Descripción

Propiedad; una cadena que especifica el patrón de una línea no justificada. Esta propiedad sólo está disponible si la propiedad `stroke.style` se define como `"ragged"` (véase `stroke.style`). Los valores válidos son: `"solid"`, `"simple"`, `"random"`, `"dotted"`, `"random dotted"`, `"triple dotted"` y `"random triple dotted"`.

Ejemplo

El ejemplo siguiente define la propiedad `pattern` como `"random"` para un estilo de trazo de `ragged`:

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "ragged";
myStroke.pattern = "random";
fl.getDocumentDOM().setCustomStroke( myStroke );
```

stroke.rotate

Disponibilidad

Flash MX 2004.

Uso

```
stroke.rotate
```

Descripción

Propiedad; una cadena que especifica la rotación de una línea de sombreado. Esta propiedad sólo está disponible si la propiedad `stroke.style` se define como `"hatched"` (véase `stroke.style`). Los valores válidos son: `"none"`, `"slight"`, `"medium"` y `"free"`.

Ejemplo

El ejemplo siguiente define la propiedad `rotate` como `"free" para un estilo de trazo de hatched`:

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "hatched";
myStroke.curve = "straight";
myStroke.space = "close";
myStroke.jiggle = "wild";
myStroke.rotate = "free";
myStroke.length = "slight";
myStroke.hatchThickness = "thin";
```

stroke.scaleType

Disponibilidad

Flash 8.

Uso

```
stroke.scaleType
```

Descripción

Propiedad; una cadena que especifica el tipo de escala que se aplicará al trazo. Los valores aceptables son: `"normal"`, `"horizontal"`, `"vertical"` y `"none"`.

Ejemplo

El ejemplo siguiente define el tipo de escala del trazo como `""horizontal"`:

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.scaleType = "horizontal";
fl.getDocumentDOM().setCustomStroke(myStroke);
```

stroke.shapeFill

Disponibilidad

Flash 8.

Uso

```
stroke.shapeFill
```

Descripción

Propiedad; un Objeto Fill que representa el valor de relleno del trazo.

Ejemplo

El ejemplo siguiente especifica la configuración de relleno y, a continuación, la aplica al trazo:

```
var fill = fl.getDocumentDOM().getCustomFill();
fill.linearGradient = true;
fill.colorArray = [ 00ff00, ff0000, fffff ];
var stroke = fl.getDocumentDOM().getCustomStroke();
stroke.shapeFill = fill;
fl.getDocumentDOM().setCustomStroke(stroke);
```

stroke.space

Disponibilidad

Flash MX 2004.

Uso

```
stroke.space
```

Descripción

Propiedad; una cadena que especifica el espaciado de una línea de sombreado. Esta propiedad sólo está disponible si la propiedad `stroke.style` se define como `"hatched"` (véase `stroke.style`). Los valores válidos son: `"very close"`, `"close"`, `"distant"` y `"very distant"`.

Ejemplo

El ejemplo siguiente define la propiedad `space` como `"close"` para un estilo de trazo de `hatched`:

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "hatched";
myStroke.curve = "straight";
myStroke.space = "close";
myStroke.jiggle = "wild";
myStroke.rotate = "free";
myStroke.length = "slight";
myStroke.hatchThickness = "thin";
fl.getDocumentDOM().setCustomStroke( myStroke );
```

stroke.strokeHinting

Disponibilidad

Flash 8.

Uso

```
stroke.strokeHinting
```

Descripción

Propiedad; un valor booleano que especifica si se definen sugerencias en el trazo.

Ejemplo

El ejemplo siguiente activa las sugerencias para el trazo:

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.strokeHinting = true;
fl.getDocumentDOM().setCustomStroke(myStroke);
```

stroke.style

Disponibilidad

Flash MX 2004.

Uso

```
stroke.style
```

Descripción

Propiedad; una cadena que describe el estilo de trazo. Los valores válidos son: `"noStroke"`, `"solid"`, `"dashed"`, `"dotted"`, `"ragged"`, `"stipple"` y `"hatched"`. Algunos de estos valores requieren la definición de propiedades adicionales del objeto stroke, como se describe en la lista siguiente:

- Si el valor es `"solid"` o `"noStroke"`, no hay otras propiedades.
- Si el valor es `"dashed"`, hay dos propiedades adicionales: `"dash1"` y `"dash2"`.
- Si el valor es `"dotted"`, hay una propiedad adicional: `"dotSpace"`.
- Si el valor es `"ragged"`, hay tres propiedades adicionales: `"pattern"`, `"waveHeight"` y `"waveLength"`.
- Si el valor es `"stipple"`, hay tres propiedades adicionales: `"dotSize"`, `"variation"` y `"density"`.
- Si el valor es `"hatched"`, hay seis propiedades adicionales: `"hatchThickness"`, `"space"`, `"jiggle"`, `"rotate"`, `"curve"` y `"length"`.

Ejemplo

El ejemplo siguiente define el estilo de trazo como `"ragged"`:

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "ragged";
fl.getDocumentDOM().setCustomStroke( myStroke );
```

stroke.thickness

Disponibilidad

Flash MX 2004.

Uso

```
stroke.thickness
```

Descripción

Propiedad; un entero que especifica el tamaño del trazo.

Ejemplo

El ejemplo siguiente define la propiedad `thickness` del trazo con un valor de `2`:

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.thickness = 2;
fl.getDocumentDOM().setCustomStroke( myStroke );
```

stroke.variation

Disponibilidad

Flash MX 2004.

Uso

```
stroke.variation
```

Descripción

Propiedad; una cadena que especifica la variación de una línea punteada. Esta propiedad sólo está disponible si la propiedad `stroke.style` se define como `"stipple"` (véase `stroke.style`). Los valores válidos son: `"one size"`, `"small variation"`, `"varied sizes"` y `"random sizes"`.

Ejemplo

El ejemplo siguiente define la propiedad de variación como `"random sizes"` para el estilo de trazo de `stipple`:

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "stipple";
myStroke.dotSpace= 3;
myStroke.variation = "random sizes";
myStroke.density = "sparse";
fl.getDocumentDOM().setCustomStroke( myStroke );
```

stroke.waveHeight

Disponibilidad

Flash MX 2004.

Uso

```
stroke.waveHeight
```

Descripción

Propiedad; una cadena que especifica la altura de onda de una línea no justificada. Esta propiedad sólo está disponible si la propiedad `stroke.style` se define como `"ragged"` (véase `stroke.style`). Los valores válidos son: `"flat"`, `"wavy"`, `"very wavy"` y `"wild"`.

Ejemplo

El ejemplo siguiente define la propiedad `waveHeight` como `"flat"` para un estilo de trazo de `ragged`:

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "ragged";
myStroke.pattern = "random";
myStroke.waveHeight = "flat";
myStroke.waveLength = "short";
fl.getDocumentDOM().setCustomStroke( myStroke );
```

stroke.waveLength

Disponibilidad

Flash MX 2004.

Uso

```
stroke.waveLength
```

Descripción

Propiedad; una cadena que especifica la longitud de onda de una línea no justificada. Esta propiedad sólo está disponible si la propiedad `stroke.style` se define como `"ragged"` (véase `stroke.style`). Los valores válidos son: `"very short"`, `"short"`, `"medium"` y`"long"`.

Ejemplo

El ejemplo siguiente define la propiedad `waveLength` como `"short"` para un estilo de trazo de `ragged`:

```
var myStroke = fl.getDocumentDOM().getCustomStroke();
myStroke.style = "ragged";
myStroke.pattern = "random";
myStroke.waveHeight = 'flat";
myStroke.waveLength = "short";
fl.getDocumentDOM().setCustomStroke( myStroke );
```

Objeto SymbolInstance

Herencia Objeto Element > Objeto Instance > Objeto SymbolInstance

Disponibilidad

Flash MX 2004.

Descripción

SymbolInstance es una subclase del objeto Instance y representa un símbolo en un fotograma (véase Objeto Instance).

Resumen de propiedades del objeto SymbolInstance

Además de las propiedades del Objeto Instance, el objeto SymbolInstance tiene las siguientes:

Propiedad	Descripción
`symbolInstance.accName`	Una cadena que equivale al campo Nombre del panel Accesibilidad.
`symbolInstance.actionScript`	Una cadena que especifica las acciones asignadas al símbolo.
`symbolInstance.blendMode`	Una cadena que especifica el modo de mezcla que se aplica a un símbolo de clip de película.
`symbolInstance.buttonTracking`	Una cadena que define (sólo para símbolos de botón) la misma propiedad que el menú emergente para Seguimiento como botón o Seguimiento como elemento de menú en el Inspector de propiedades.
`symbolInstance.cacheAsBitmap`	Un valor booleano que especifica si está activada la caché de mapa de bits en tiempo de ejecución.
`symbolInstance.colorAlphaAmount`	Un entero que forma parte de la transformación de color de la instancia, especificando la configuración de Efecto avanzado Alfa; equivale a utilizar la opción Color › Avanzado en el Inspector de propiedades y ajustar los controles de la derecha del cuadro de diálogo.
`symbolInstance.colorAlphaPercent`	Un entero que especifica parte de la transformación de color para la instancia; equivale a utilizar la opción Color › Avanzado en el inspector de propiedades de la instancia (los controles de porcentaje de la parte izquierda del cuadro de diálogo).

Propiedad	Descripción
symbolInstance.colorBlueAmount	Un entero que forma parte de la transformación de color de la instancia; equivale a utilizar la opción Color › Avanzado en el inspector de propiedades de la instancia.
symbolInstance.colorBluePercent	Un entero que forma parte de la transformación de color para la instancia; equivale a utilizar la opción Color › Avanzado en el inspector de propiedades de la instancia (los controles de porcentaje de la parte izquierda del cuadro de diálogo).
symbolInstance.colorGreenAmount	Un entero que forma parte de la transformación de color de la instancia; equivale a utilizar la opción Color › Avanzado en el inspector de propiedades de la instancia. Los valores válidos van de -255 a 255.
symbolInstance.colorGreenPercent	Parte de la transformación de color para la instancia; equivale a utilizar la opción Color › Avanzado en el inspector de propiedades de la instancia (los controles de porcentaje de la parte izquierda del cuadro de diálogo).
symbolInstance.colorMode	Una cadena que especifica el modo de color identificado en el menú emergente Color del inspector de propiedades de símbolo.
symbolInstance.colorRedAmount	Un entero que forma parte de la transformación de color de la instancia, equivale a utilizar la opción Color › Avanzado en el inspector de propiedades de la instancia.
symbolInstance.colorRedPercent	Parte de la transformación de color para la instancia; equivale a utilizar la opción Color › Avanzado en el inspector de propiedades de la instancia (los controles de porcentaje de la parte izquierda del cuadro de diálogo).
symbolInstance.description	Una cadena que equivale al campo Descripción del panel Accesibilidad.
symbolInstance.filters	Una matriz de objetos Filter (véase Objeto Filter).
symbolInstance.firstFrame	Un entero basado en cero que especifica el primer fotograma que aparecerá en la línea de tiempo del gráfico.

Propiedad	Descripción
symbolInstance.forceSimple	Un valor booleano que activa o desactiva la accesibilidad de los elementos secundarios del objeto; equivale a la lógica inversa de la opción Hacer que los objetos secundarios sean accesibles del panel Accesibilidad.
symbolInstance.loop	Una cadena que define (para símbolos gráficos) la misma propiedad que el menú emergente Reproducir indefinidamente en el Inspector de propiedades.
symbolInstance.shortcut	Una cadena que equivale a la tecla de método abreviado asociada al símbolo; equivale al campo Método abreviado del panel Accesibilidad.
symbolInstance.silent	Un valor booleano que activa o desactiva la accesibilidad del objeto; equivale a la lógica inversa de la opción Hacer que el objeto sea accesible del panel Accesibilidad.
symbolInstance.symbolType	Una cadena que especifica el tipo de símbolo; equivale al valor de Comportamiento en los cuadros de diálogo Crear nuevo símbolo y Convertir en símbolo.
symbolInstance.tabIndex	Un entero que equivale al campo Índice de fichas del panel Accesibilidad.

symbolInstance.accName

Disponibilidad

Flash MX 2004.

Uso

```
symbolInstance.accName
```

Descripción

Propiedad; una cadena que equivale al campo Nombre del panel Accesibilidad. Los lectores de pantalla identifican los objetos mediante la lectura del nombre en voz alta. Esta propiedad no está disponible para símbolos gráficos.

Ejemplo

El ejemplo siguiente almacena el valor del nombre del panel Accesibilidad del objeto en la variable `theName`:

```
var theName = fl.getDocumentDOM().selection[0].accName;
```

El ejemplo siguiente define el valor del nombre del panel Accesibilidad del objeto como `"Home Button"`:

```
fl.getDocumentDOM().selection[0].accName = "Home Button";
```

symbolInstance.actionScript

Disponibilidad

Flash MX 2004.

Uso

```
symbolInstance.actionScript
```

Descripción

Propiedad; una cadena que especifica las acciones asignadas al símbolo. Sólo se aplica a instancias de botones y clips de película. Para una instancia de símbolos gráficos, el valor devuelve no definido.

Ejemplo

El ejemplo siguiente asigna una acción `onClipEvent` al primer elemento del primer fotograma de la primera capa de la línea de tiempo:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].actionScr
  ipt
  = "onClipEvent(enterFrame) {trace('movie clip enterFrame');}";
```

symbolInstance.blendMode

Disponibilidad

Flash 8.

Uso

```
symbolInstance.blendMode
```

Descripción

Propiedad; una cadena que especifica el modo de mezcla que se aplica a un símbolo de clip de película. Los valores aceptables son: `"normal"`, `"layer"`, `"multiply"`, `"screen"`, `"overlay"`, `"hardlight"`, `"lighten"`, `"darken"`, `"difference"`, `"add"`, `"subtract"`, `"invert"`, `"alpha"` y `"erase"`.

Ejemplo

El ejemplo siguiente establece como `"add"` el modo de mezcla para el primer símbolo de clip de película en el primer fotograma del primer nivel:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].blendMode
  = 'add';
```

Véase también

`document.setBlendMode()`

symbolInstance.buttonTracking

Disponibilidad

Flash MX 2004.

Uso

```
symbolInstance.buttonTracking
```

Descripción

Propiedad; una cadena que define (sólo para símbolos de botón) la misma propiedad que el menú emergente para Seguimiento como botón o Seguimiento como elemento de menú en el Inspector de propiedades. Esta propiedad se ignora con otros tipos de símbolos. Los valores aceptables son: `"button"` o `"menu"`.

Ejemplo

El ejemplo siguiente define el primer símbolo del primer fotograma de la primera capa de la línea de tiempo como Seguimiento como elemento de menú, siempre que ese símbolo sea un botón:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].buttonTra
  cking = "menu";
```

symbolInstance.cacheAsBitmap

Disponibilidad

Flash 8.

Uso

```
symbolInstance.cacheAsBitmap
```

Descripción

Propiedad; un valor booleano que especifica si está activada la caché de mapa de bits en tiempo de ejecución.

Ejemplo

El ejemplo siguiente activa la caché de mapa de bits en tiempo de ejecución para el primer elemento del primer fotograma de la primera capa:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].cacheAsBi
  tmap = true;
```

symbolInstance.colorAlphaAmount

Disponibilidad

Flash MX 2004.

Uso

```
symbolInstance.colorAlphaAmount
```

Descripción

Propiedad; un entero que forma parte de la transformación de color de la instancia, especificando la configuración de Efecto avanzado Alfa. Esta propiedad equivale a utilizar la opción Color > Avanzado en el Inspector de propiedades y ajustar los controles de la derecha del cuadro de diálogo. Este valor reduce o aumenta los valores de tinta y alfa en una cantidad constante. Este valor se añade al valor actual. Esta propiedad resulta más útil si se utiliza con `symbolInstance.colorAlphaPercent`. Los valores válidos van de -255 a 255.

Ejemplo

El ejemplo siguiente resta 100 del valor de alfa de la instancia de símbolo seleccionada:

```
fl.getDocumentDOM().selection[0].colorAlphaAmount = -100;
```

symbolInstance.colorAlphaPercent

Disponibilidad

Flash MX 2004.

Uso

```
symbolInstance.colorAlphaPercent
```

Descripción

Propiedad; un entero que especifica parte de la transformación de color de la instancia. Esta propiedad equivale a utilizar la opción Color > Avanzado en el inspector de propiedades de la instancia (los controles de porcentaje de la izquierda del cuadro de diálogo). Este valor cambia los valores de tinta y alfa al porcentaje especificado. Los valores válidos van de -100 a 100. Véase también `symbolInstance.colorAlphaAmount`.

Ejemplo

El ejemplo siguiente define el `colorAlphaPercent` de la instancia de símbolo seleccionada como 80:

```
fl.getDocumentDOM().selection[0].colorAlphaPercent = 80;
```

symbolInstance.colorBlueAmount

Disponibilidad

Flash MX 2004.

Uso

```
symbolInstance.colorBlueAmount
```

Descripción

Propiedad; un entero que forma parte de la transformación de color de la instancia. Esta propiedad equivale a utilizar la opción Color > Avanzado en el inspector de propiedades de la instancia. Los valores válidos van de -255 a 255.

symbolInstance.colorBluePercent

Disponibilidad

Flash MX 2004.

Uso

```
symbolInstance.colorBluePercent
```

Descripción

Propiedad; un entero que forma parte de la transformación de color de la instancia. Esta propiedad equivale a utilizar la opción Color > Avanzado en el inspector de propiedades de la instancia (los controles de porcentaje de la izquierda del cuadro de diálogo). Este valor define los valores de azul en un porcentaje especificado. Los valores válidos van de -100 a 100.

Ejemplo

El ejemplo siguiente define el `colorBluePercent` de la instancia de símbolo seleccionada como 80:

```
fl.getDocumentDOM().selection[0].colorBluePercent = 80;
```

symbolInstance.colorGreenAmount

Disponibilidad

Flash MX 2004.

Uso

```
symbolInstance.colorGreenAmount
```

Descripción

Propiedad; un entero que forma parte de la transformación de color de la instancia. Esta propiedad equivale a utilizar la opción Color > Avanzado en el inspector de propiedades de la instancia. Los valores válidos van de -255 a 255.

symbolInstance.colorGreenPercent

Disponibilidad

Flash MX 2004.

Uso

```
symbolInstance.colorGreenPercent
```

Descripción

Propiedad; forma parte de la transformación de color de la instancia. Esta propiedad equivale a utilizar la opción Color > Avanzado en el inspector de propiedades de la instancia (los controles de porcentaje de la izquierda del cuadro de diálogo). Este valor define los valores de verde en un porcentaje especificado. Los valores válidos van de -100 a 100.

Ejemplo

El ejemplo siguiente define el `colorGreenPercent` de la instancia de símbolo seleccionada como 70:

```
fl.getDocumentDOM().selection[0].colorGreenPercent = 70;
```

symbolInstance.colorMode

Disponibilidad

Flash MX 2004.

Uso

```
symbolInstance.colorMode
```

Descripción

Propiedad; una cadena que especifica el modo de color identificado en el menú emergente Color del inspector de propiedades de símbolo. Los valores válidos son: `"none"`, `"brightness"`, `"tint"`, `"alpha"` y `"advanced"`.

Ejemplo

El ejemplo siguiente cambia la propiedad `colorMode` del primer elemento del primer fotograma de la primera capa de la línea de tiempo a `"alpha"`:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].colorMode
  = "alpha";
```

symbolInstance.colorRedAmount

Disponibilidad

Flash MX 2004.

Uso

```
symbolInstance.colorRedAmount
```

Descripción

Propiedad; un entero que forma parte de la transformación de color de la instancia. Esta propiedad equivale a utilizar la opción Color > Avanzado en el inspector de propiedades de la instancia. Los valores válidos van de -255 a 255.

Ejemplo

El ejemplo siguiente define el `colorRedAmount` de la instancia de símbolo seleccionada como 255:

```
fl.getDocumentDOM().selection[0].colorRedAmount = 255;
```

symbolInstance.colorRedPercent

Disponibilidad

Flash MX 2004.

Uso

```
symbolInstance.colorRedPercent
```

Descripción

Propiedad; forma parte de la transformación de color de la instancia. Esta propiedad equivale a utilizar la opción Color > Avanzado en el inspector de propiedades de la instancia (los controles de porcentaje de la izquierda del cuadro de diálogo). Este valor define los valores de rojo en un porcentaje especificado. Los valores válidos van de -100 a 100.

Ejemplo

El ejemplo siguiente define el `colorRedPercent` de la instancia de símbolo seleccionada como 10:

```
fl.getDocumentDOM().selection[0].colorRedPercent = 10;
```

symbolInstance.description

Disponibilidad

Flash MX 2004.

Uso

```
symbolInstance.description
```

Descripción

Propiedad; una cadena que equivale al campo Descripción del panel Accesibilidad. El lector de pantalla lee esta descripción. Esta propiedad no está disponible para símbolos gráficos.

Ejemplo

El ejemplo siguiente almacena el valor de la descripción del panel Accesibilidad del objeto en la variable `theDescription`:

```
var theDescription = fl.getDocumentDOM().selection[0].description;
```

El ejemplo siguiente define el valor de la descripción del panel Accesibilidad como “Click the home button to go to home”:

```
fl.getDocumentDOM().selection[0].description= "Click the home button to go
  to home";
```

symbolInstance.filters

Disponibilidad

Flash 8.

Uso

```
symbolInstance.filters
```

Descripción

Propiedad; una matriz de objetos Filter (véase Objeto Filter). Para modificar las propiedades de filtro, no se escribe en esta matriz directamente, sino que se debe recuperar la matriz, definir las propiedades individuales y después definir la matriz para que refleje las nuevas propiedades.

Ejemplo

El siguiente ejemplo traza el nombre del filtro en el índice 0. Si es un filtro de iluminado, la propiedad `blurX` se define como 100 y el nuevo valor se escribe en la matriz de filtros.

```
var filterName =
  fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].filter
  s[0].name;
fl.trace(filterName);
var filterArray =
  fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].filter
  s;
if (filterName == 'glowFilter'){
  filterArray[0].blurX = 100;
}
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].filters =
  filterArray;
```

symbolInstance.firstFrame

Disponibilidad

Flash MX 2004.

Uso

```
symbolInstance.firstFrame
```

Descripción

Propiedad; un entero basado en cero que especifica el primer fotograma que aparecerá en la línea de tiempo del gráfico. Esta propiedad sólo se aplica a símbolos gráficos y define la misma propiedad que el campo Primero del inspector de propiedades. Para otros tipos de símbolos, esta propiedad es `undefined`.

Ejemplo

El ejemplo siguiente especifica que el Fotograma 11 debe ser el primer fotograma que aparezca en la línea de tiempo del elemento especificado:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].firstFram
  e = 10;
```

symbolInstance.forceSimple

Disponibilidad

Flash MX 2004.

Uso

```
symbolInstance.forceSimple
```

Descripción

Propiedad; un valor booleano que activa o desactiva la accesibilidad para los elementos secundarios del objeto. Esta propiedad equivale a la lógica inversa de la opción Hacer que los objetos secundarios sean accesibles del panel Accesibilidad. Por ejemplo, si `forceSimple` es `true`, equivale a la opción desactivada Hacer que los objetos secundarios sean accesibles. Si `forceSimple` es `false`, equivale a la opción activada Hacer que los objetos secundarios sean accesibles.

Esta propiedad sólo está disponible para objetos de clip de película.

Ejemplo

El ejemplo siguiente comprueba si los elementos secundarios del objeto son accesibles; un valor devuelto de `false` significa que los elementos secundarios son accesibles:

```
var areChildrenAccessible = fl.getDocumentDOM().selection[0].forceSimple;
```

El ejemplo siguiente permite que los elementos secundarios del objeto sean accesibles:

```
fl.getDocumentDOM().selection[0].forceSimple = false;
```

symbolInstance.loop

Disponibilidad

Flash MX 2004.

Uso

```
symbolInstance.loop
```

Descripción

Propiedad; una cadena que define (para símbolos gráficos) la misma propiedad que el menú emergente Reproducir indefinidamente en el Inspector de propiedades. Para otros tipos de símbolos, esta propiedad es `undefined`. Los valores aceptables son: `"loop"`, `"play once"` y `"single frame"` para definir la animación del gráfico en consonancia.

Ejemplo

El ejemplo siguiente define el primer símbolo del primer fotograma de la primera capa de la línea de tiempo como Fotograma único (muestra un fotograma especificado de la línea de tiempo gráfica), siempre que ese símbolo sea un gráfico:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].loop =
  'single frame';
```

symbolInstance.shortcut

Disponibilidad

Flash MX 2004.

Uso

```
symbolInstance.shortcut
```

Descripción

Propiedad; una cadena que equivale a la tecla de método abreviado asociada al símbolo. Este propiedad equivale al campo Método abreviado del panel Accesibilidad. El lector de pantalla lee esta tecla. Esta propiedad no está disponible para símbolos gráficos.

Ejemplo

El ejemplo siguiente almacena el valor de la tecla de método abreviado del objeto en la variable `theShortcut`:

```
var theShortcut = fl.getDocumentDOM().selection[0].shortcut;
```

El ejemplo siguiente define la tecla de método abreviado del objeto como `"Ctrl+i"`:

```
fl.getDocumentDOM().selection[0].shortcut = "Ctrl+i";
```

symbolInstance.silent

Disponibilidad

Flash MX 2004.

Uso

```
symbolInstance.silent
```

Descripción

Propiedad; un valor booleano que activa o desactiva la accesibilidad del objeto. Esta propiedad equivale a la lógica inversa de la opción Hacer que el objeto sea accesible del panel Accesibilidad. Por ejemplo, si `silent` es `true`, equivale a la opción desactivada Hacer que el objeto sea accesible. Si `silent` es `false`, equivale a la opción activada Hacer que el objeto sea accesible.

Esta propiedad no está disponible para objetos gráficos.

Ejemplo

El ejemplo siguiente comprueba si el objeto es accesible; un valor devuelto de `false` significa que el objeto es accesible:

```
var isSilent = fl.getDocumentDOM().selection[0].silent;
```

El ejemplo siguiente define el objeto como accesible:

```
fl.getDocumentDOM().selection[0].silent = false;
```

symbolInstance.symbolType

Disponibilidad

Flash MX 2004.

Uso

```
symbolInstance.symbolType
```

Descripción

Propiedad; una cadena que especifica el tipo de símbolo. Esta propiedad equivale al valor de Comportamiento de los cuadros de diálogo Crear nuevo símbolo y Convertir en símbolo. Los valores aceptables son: `"button"`, `"movie clip"` y `"graphic"`.

Ejemplo

El ejemplo siguiente define el primer símbolo del primer fotograma de la primera capa de la línea de tiempo del documento actual para que se comporte como un símbolo gráfico:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].symbolTyp
  e = "graphic";
```

symbolInstance.tabIndex

Disponibilidad

Flash MX 2004.

Uso

```
symbolInstance.tabIndex
```

Descripción

Propiedad; un entero que equivale al campo Índice de fichas del panel Accesibilidad. Crea un orden de tabulación con el que se accede a los objetos cuando el usuario presiona la tecla Tabulador. Esta propiedad no está disponible para símbolos gráficos.

Ejemplo

El ejemplo siguiente define la propiedad `tabIndex` del objeto mySymbol como 3 y muestra ese valor en el panel Salida:

```
var mySymbol = fl.getDocumentDOM().selection[0];
mySymbol.tabIndex = 3;
fl.trace(mySymbol.tabIndex);
```

Objeto SymbolItem

Herencia Objeto Item > Objeto SymbolItem

Disponibilidad

Flash MX 2004.

Descripción

El objeto SymbolItem es una subclase del Objeto Item.

Resumen de métodos del objeto SymbolItem

Además de los métodos del Objeto Item, puede emplear los siguientes con el objeto SymbolItem:

Método	Descripción
`symbolItem.convertToCompiledClip()`	Convierte un elemento de símbolo de la biblioteca en un clip de película compilado.
`symbolItem.exportSWC()`	Exporta el elemento de símbolo a un archivo SWC.
`symbolItem.exportSWF()`	Exporta el elemento de símbolo a un archivo SWF.

Resumen de propiedades del objeto SymbolItem

Además de las propiedades de Objeto Item, el objeto SymbolItem dispone de las siguientes:

Propiedad	Descripción
`symbolItem.scalingGrid`	Un valor booleano que especifica si se activa la escala en 9 divisiones para el elemento.
`symbolItem.scalingGridRect`	Un valor booleano que especifica si se activa la escala en 9 divisiones para el elemento.
`symbolItem.sourceAutoUpdate`	Un valor booleano que especifica si el elemento se actualizará cuando se publique el archivo FLA.
`symbolItem.sourceFilePath`	Una cadena que especifica la ruta del archivo FLA de origen como archivo:/// URI.
`symbolItem.sourceLibraryName`	Una cadena que especifica el nombre del elemento de la biblioteca de archivos de origen.
`symbolItem.symbolType`	Una cadena que especifica el tipo de símbolo.
`symbolItem.timeline`	De sólo lectura; un Objeto Timeline.

symbolItem.convertToCompiledClip()

Disponibilidad

Flash MX 2004.

Uso

```
symbolItem.convertToCompiledClip()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; convierte un elemento de símbolo de la biblioteca en un clip de película compilado.

Ejemplo

El ejemplo siguiente convierte un elemento de la biblioteca en un clip de película compilado:

```
fl.getDocumentDOM().library.items[3].convertToCompiledClip();
```

symbolItem.exportSWC()

Disponibilidad

Flash MX 2004.

Uso

```
symbolItem.exportSWC( outputURI )
```

Parámetros

outputURI Una cadena, expresada como archivo:/// URI, que especifica el archivo SWC al que el método exportará el símbolo. El *outputURI* debe hacer referencia a un archivo local. Flash no crea una carpeta si no existe *outputURI*.

Valor devuelto

Ninguno.

Descripción

Método; exporta el elemento de símbolo a un archivo SWC.

Ejemplo

El ejemplo siguiente exporta un elemento de la biblioteca al archivo SWC llamado my.swc de la carpeta tests:

```
fl.getDocumentDOM().library.items[0].exportSWC("file:///c|/tests/my.swc");
```

symbolItem.exportSWF()

Disponibilidad

Flash MX 2004.

Uso

```
symbolItem.exportSWF( outputURI )
```

Parámetros

outputURI Una cadena, expresada como archivo:/// URI, que especifica el archivo SWF al que el método exportará el símbolo. El *outputURI* debe hacer referencia a un archivo local. Flash no crea una carpeta si no existe *outputURI*.

Valor devuelto

Ninguno.

Descripción

Método; exporta el elemento de símbolo a un archivo SWF.

Ejemplo

El ejemplo siguiente exporta un elemento de la biblioteca al archivo my.swf de la carpeta tests:

```
fl.getDocumentDOM().library.items[0].exportSWF("file:///c|/tests/my.swf");
```

symbolItem.scalingGrid

Disponibilidad

Flash 8.

Uso

```
symbolItem.scalingGrid
```

Descripción

Propiedad; un valor booleano que especifica si se activa la escala en 9 divisiones para el elemento.

Ejemplo

El siguiente ejemplo activa la escala en 9 divisiones para un elemento de la biblioteca:

```
fl.getDocumentDOM().library.items[0].scalingGrid = true;
```

Véase también

`symbolItem.scalingGridRect`

symbolItem.scalingGridRect

Disponibilidad

Flash 8.

Uso

```
symbolItem.scalingGridRect
```

Descripción

Propiedad; un objeto de rectángulo que especifica las ubicaciones de las cuatro guías de 9 divisiones. Para más información sobre el formato del rectángulo, consulte `document.addNewRectangle()`.

Ejemplo

El ejemplo siguiente especifica las ubicaciones de las guías en 9 divisiones.

```
fl.getDocumentDOM().library.items[0].scalingGridRect = {left:338, top:237,
  right:3859, bottom:713};
```

Véase también

`symbolItem.scalingGrid`

symbolItem.sourceAutoUpdate

Disponibilidad

Flash MX 2004.

Uso

```
symbolItem.sourceAutoUpdate
```

Descripción

Propiedad; un valor booleano que especifica si el elemento se actualizará cuando se publique el archivo FLA. El valor predeterminado es `false`. Se utiliza para símbolos de bibliotecas compartidas.

Ejemplo

El ejemplo siguiente define la propiedad `sourceAutoUpdate` para un elemento de biblioteca:

```
fl.getDocumentDOM().library.items[0].sourceAutoUpdate = true;
```

symbolItem.sourceFilePath

Disponibilidad

Flash MX 2004.

Uso

```
symbolItem.sourceFilePath
```

Descripción

Propiedad; una cadena que especifica la ruta del archivo FLA de origen como archivo:/// URI. La ruta debe ser una ruta absoluta, no relativa. Esta propiedad se utiliza para símbolos de bibliotecas compartidas.

Ejemplo

El ejemplo siguiente muestra el valor de la propiedad `sourceFilePath` en el panel Salida:

```
fl.trace(fl.getDocumentDOM().library.items[0].sourceFilePath);
```

symbolItem.sourceLibraryName

Disponibilidad

Flash MX 2004.

Uso

```
symbolItem.sourceLibraryName
```

Descripción

Propiedad; una cadena que especifica el nombre del elemento de la biblioteca de archivos de origen. Se utiliza para símbolos de bibliotecas compartidas.

Ejemplo

El ejemplo siguiente muestra el valor de la propiedad `sourceLibraryName` en el panel Salida:

```
fl.trace(fl.getDocumentDOM().library.items[0].sourceLibraryName);
```

symbolItem.symbolType

Disponibilidad

Flash MX 2004.

Uso

```
symbolItem.symbolType
```

Descripción

Propiedad; una cadena que especifica el tipo de símbolo. Los valores aceptables son: `"movie clip"`, `"button"` y `"graphic"`.

Ejemplo

El ejemplo siguiente muestra el valor actual de la propiedad `symbolType`, lo cambia a `"button"` y vuelve a mostrarlo:

```
alert(fl.getDocumentDOM().library.items[0].symbolType);
fl.getDocumentDOM().library.items[0].symbolType = "button";
alert(fl.getDocumentDOM().library.items[0].symbolType);
```

symbolItem.timeline

Disponibilidad

Flash MX 2004.

Uso

```
symbolItem.timeline
```

Descripción

Propiedad de sólo lectura; un Objeto Timeline.

Ejemplo

El ejemplo siguiente obtiene y muestra el número de capas que contiene el clip de película seleccionado en la biblioteca:

```
var tl = fl.getDocumentDOM().library.getSelectedItems()[0].timeline;
alert(tl.layerCount);
```

Objeto Text

Herencia Objeto Element > Objeto Text

Disponibilidad

Flash MX 2004.

Descripción

El objeto Text representa un elemento de texto único en un documento. Todas las propiedades del texto pertenecen al bloque de texto completo.

Para definir las propiedades de una extensión de texto en el campo de texto, consulte "Resumen de propiedades del objeto TextRun" en la página 474. Para cambiar las propiedades de una selección en un campo de texto, puede utilizar `document.setElementTextAttr()` y especificar un rango de texto o emplear la selección actual.

Para definir las propiedades de texto del campo de texto seleccionado, utilice `document.setElementProperty()`. El ejemplo siguiente asigna el campo de texto seleccionado actualmente a la variable `textVar`:

```
fl.getDocumentDOM().setElementProperty("variableName", "textVar");
```

Resumen de métodos para el objeto Text

Además de los métodos del Objeto Element, puede emplear los métodos siguientes con el objeto Text:

Método	Descripción
`text.getTextAttr()`	Recupera el atributo especificado para el texto identificado por los parámetros `startIndex` y `endIndex` opcionales.
`text.getTextString()`	Recupera el rango de texto especificado.
`text.setTextAttr()`	Define el atributo especificado asociado al texto identificado por `startIndex` y `endIndex`.
`text.setTextString()`	Cambia la cadena de texto en este objeto de texto.

Resumen de propiedades del objeto Text

Además de las propiedades del Objeto Element, el objeto Text dispone de las siguientes:

Propiedad	Descripción
`text.accName`	Una cadena que equivale al campo Nombre del panel Accesibilidad.
`text.antiAliasSharpness`	Un valor flotante que especifica la nitidez de suavizado del texto.
`text.antiAliasThickness`	Un valor flotante que especifica el grosor de suavizado del texto.
`text.autoExpand`	Un valor booleano que controla la expansión de la anchura de delimitación para campos de texto estático o la anchura y la altura de delimitación para texto dinámico o de entrada.
`text.border`	Un valor booleano que controla si Flash muestra (`true`) u oculta (`false`) un borde alrededor del texto dinámico o de entrada.
`text.description`	Una cadena que equivale al campo Descripción del panel Accesibilidad.
`text.embeddedCharacters`	Una cadena que especifica caracteres para incorporar. Equivale a introducir texto en el cuadro de diálogo Opciones de caracteres.
`text.embedRanges`	Una cadena compuesta por enteros delimitados que corresponde a los elementos que se pueden seleccionar en el cuadro de diálogo Opciones de caracteres.
`text.fontRenderingMode`	Una cadena que especifica el modo de presentación del texto.
`text.length`	De sólo lectura; un entero que representa el número de caracteres del objeto de texto.
`text.lineType`	Una cadena que define el tipo de línea como `"single line"`, `"multiline"`, `"multiline no wrap"` o `"password"`.
`text.maxCharacters`	Un entero que especifica los caracteres máximos que el usuario puede introducir en este objeto de texto.
`text.orientation`	Una cadena que especifica la orientación del campo de texto.
`text.renderAsHTML`	Un valor booleano que controla si Flash dibuja el texto como HTML e interpreta etiquetas HTML incorporadas.
`text.scrollable`	Un valor booleano que controla si el texto se puede desplazar (`true`) o no (`false`).

Propiedad	Descripción
text.selectable	Un valor booleano que controla si el texto se puede seleccionar (true) o no (false). El texto de entrada siempre se puede seleccionar.
text.selectionEnd	Un entero basado en cero que especifica el desplazamiento del final de una subselección de texto.
text.selectionStart	Un entero basado en cero que especifica el desplazamiento del principio de una subselección de texto.
text.shortcut	Una cadena que equivale al campo Método abreviado del panel Accesibilidad.
text.silent	Un valor booleano que especifica si el objeto es accesible.
text.tabIndex	Un entero que equivale al campo Índice de fichas del panel Accesibilidad.
text.textRuns	De sólo lectura; una matriz de objetos TextRun.
text.textType	Una cadena que especifica el tipo del campo de texto. Los valores válidos son: "static", "dynamic" e "input".
text.useDeviceFonts	Valor booleano. Un valor de true hace que Flash dibuje el texto utilizando fuentes del dispositivo.
text.variableName	Una cadena que alberga el contenido del objeto de texto.

text.antiAliasSharpness

Disponibilidad

Flash 8.

Uso

```
text.antiAliasSharpness
```

Descripción

Propiedad; un valor flotante que especifica la nitidez de suavizado del texto. Esta propiedad controla la nitidez con la que se dibuja el texto; los valores más altos especifican texto de mayor nitidez. Un valor de 0 especifica nitidez normal. Esta propiedad sólo está disponible si text.fontRenderingMode se define como "customThicknessSharpness".

Ejemplo

Véase text.fontRenderingMode.

Véase también

text.antiAliasThickness, text.fontRenderingMode

text.antiAliasThickness

Disponibilidad

Flash 8.

Uso

```
text.antiAliasThickness
```

Descripción

Propiedad; un valor flotante que especifica el grosor de suavizado del texto. Esta propiedad controla el grosor con el que se dibuja el texto, los valores más altos especifican texto de mayor grosor. Un valor de 0 especifica grosor normal. Esta propiedad sólo está disponible si `text.fontRenderingMode` se define como `"customThicknessSharpness"`.

Ejemplo

Véase `text.fontRenderingMode`.

Véase también

`text.antiAliasSharpness`, `text.fontRenderingMode`

text.accName

Disponibilidad

Flash MX 2004.

Uso

```
text.accName
```

Descripción

Propiedad; una cadena que equivale al campo Nombre del panel Accesibilidad. Los lectores de pantalla identifican los objetos mediante la lectura del nombre en voz alta. Esta propiedad no se puede utilizar con texto dinámico.

Ejemplo

El ejemplo siguiente recupera el nombre del objeto:

```
var theName =
  fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].accNam
  e;
```

El ejemplo siguiente define el nombre del objeto seleccionado actualmente:

```
fl.getDocumentDOM().selection[0].accName = "Home Button";
```

text.autoExpand

Disponibilidad

Flash MX 2004.

Uso

```
text.autoExpand
```

Descripción

Propiedad; un valor booleano. En campos de texto estático, un valor de `true` hace que la anchura de delimitación se expanda para mostrar todo el texto. En campos de texto dinámico o de entrada, un valor de `true` hace que la anchura y la altura de delimitación se expandan para mostrar todo el texto.

Ejemplo

El ejemplo siguiente define la propiedad `autoExpand` con un valor de `true`:

```
fl.getDocumentDOM().selection[0].autoExpand = true;
```

text.border

Disponibilidad

Flash MX 2004.

Uso

```
text.border
```

Descripción

Propiedad; un valor booleano. Un valor de `true` hace que Flash muestre un borde alrededor del texto.

Ejemplo

El ejemplo siguiente define la propiedad `border` con un valor de `true`:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].border =
  true;
```

text.description

Disponibilidad

Flash MX 2004.

Uso

```
text.description
```

Descripción

Propiedad; una cadena que equivale al campo Descripción del panel Accesibilidad. El lector de pantalla lee esta descripción.

Ejemplo

El ejemplo siguiente recupera la descripción del objeto:

```
var theDescription =
  fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].descri
  ption;
```

El ejemplo siguiente establece la descripción del objeto:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].descripti
  on= "Enter your name here";
```

text.embeddedCharacters

Disponibilidad

Flash MX 2004.

Uso

```
text.embeddedCharacters
```

Descripción

Propiedad; una cadena que especifica caracteres para incorporar. Equivale a introducir texto en el cuadro de diálogo Opciones de caracteres.

Esta propiedad sólo funciona con texto dinámico o de entrada; genera una advertencia si se utiliza con otros tipos de texto.

Ejemplo

El ejemplo siguiente define la propiedad `embeddedCharacters` como `"abc"`:

```
fl.getDocumentDOM().selection[0].embeddedCharacters = "abc";
```

text.embedRanges

Disponibilidad

Flash MX 2004.

Uso

```
text.embedRanges
```

Descripción

Propiedad; una cadena compuesta por enteros delimitados que corresponde a los elementos que se pueden seleccionar en el cuadro de diálogo Opciones de caracteres. Esta propiedad sólo funciona con texto dinámico o de entrada; se ignora si se utiliza con texto estático.

NOTA	Esta propiedad corresponde al archivo XML de la carpeta Configuration/Font Embedding.

Ejemplo

El ejemplo siguiente define la propiedad `embedRanges` como `"1|3|7"`:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].embedRang
  es = "1|3|7";
```

El ejemplo siguiente restablece la propiedad:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].embedRang
  es = "";
```

text.fontRenderingMode

Disponibilidad

Flash 8.

Uso

```
text.fontRenderingMode
```

Descripción

Propiedad; una cadena que especifica el modo de presentación del texto. Esta propiedad afecta al modo en que se muestra el texto tanto en el escenario como en Flash Player. Los valores aceptables se describen en la siguiente tabla.

Valor de propiedad	Presentación del texto
`device`	Presenta el texto con fuentes de dispositivo.
`bitmap`	Presenta el texto suavizado como un mapa de bits, o como lo haría una fuente de píxel.

`standard`	Presenta texto con el método de suavizado que emplea Flash MX 2004. Se trata de la mejor configuración que se puede utilizar para texto animado.
`advanced`	Presenta el texto a través de la tecnología de representación de fuentes FlashType implementada en Flash 8, que produce un mejor suavizado y aumenta la legibilidad, especialmente la del texto pequeño.
`customThicknessSharpness`	Permite especificar configuraciones personalizadas para la nitidez y el grosor del texto al utilizar la tecnología de representación de fuentes FlashType implementada en Flash 8.

Ejemplo

El ejemplo siguiente muestra cómo se puede utilizar el valor `customThicknessSharpness` para especificar la nitidez y el grosor del texto:

```
fl.getDocumentDOM().setElementProperty("fontRenderingMode",
  "customThicknessSharpness");
fl.getDocumentDOM().setElementProperty("antiAliasSharpness", 400);
fl.getDocumentDOM().setElementProperty("antiAliasThickness", -200);
```

Véase también

`text.antiAliasSharpness`, `text.antiAliasThickness`

text.getTextAttr()

Disponibilidad

Flash MX 2004.

Uso

`text.getTextAttr(`*`attrName`* `[, ` *`startIndex`* `[, ` *`endIndex`*`]])`

Parámetros

`attrName` Una cadena que especifica el nombre de la propiedad del objeto TextAttrs que se va a devolver.

> **NOTA** Para obtener una lista de posibles valores de *`attrName`*, consulte Resumen de propiedades del objeto TextAttrs.

`startIndex` Un entero que es el índice del primer carácter. Este parámetro es opcional.

`endIndex` Un entero que especifica el final del rango de texto, que comienza por *`startIndex`* y llega hasta *`endIndex`*, no incluido. Este parámetro es opcional.

Valor devuelto

El valor del atributo especificado en el parámetro *attrName*.

Descripción

Método; recupera el atributo especificado por el parámetro *attrName* para el texto identificado por los parámetros *startIndex* y *endIndex* opcionales. Si el atributo no coincide con el rango especificado, Flash devuelve `undefined`. Si omite los parámetros opcionales *startIndex* y *endIndex*, el método utilizará el rango de texto completo. Si sólo especifica *startIndex*, el rango utilizado será el carácter que se encuentra en esa posición. Si especifica *startIndex* y *endIndex*, el rango comenzará en *startIndex* y llegará hasta *endIndex*, no incluido.

Ejemplo

El ejemplo siguiente obtiene el tamaño de fuente del campo de texto seleccionado actualmente y lo muestra:

```
var TheTextSize = fl.getDocumentDOM().selection[0].getTextAttr("size");
fl.trace(TheTextSize);
```

El ejemplo siguiente obtiene el color de relleno del texto del campo de texto seleccionado:

```
var TheFill = fl.getDocumentDOM().selection[0].getTextAttr("fillColor");
fl.trace(TheFill);
```

El ejemplo siguiente obtiene el tamaño del tercer carácter:

```
var Char2 = fl.getDocumentDOM().selection[0].getTextAttr("size", 2);
fl.trace(Char2);
```

El ejemplo siguiente obtiene el color del campo de texto seleccionado del tercer al octavo carácter:

```
fl.getDocumentDOM().selection[0].getTextAttr("fillColor", 2, 8);
```

text.getTextString()

Disponibilidad

Flash MX 2004.

Uso

```
text.getTextString([startIndex [, endIndex] ])
```

Parámetros

startIndex Un entero que especifica el índice (basado en cero) del primer carácter. Este parámetro es opcional.

endIndex Un entero que especifica el final del rango de texto, que comienza por *startIndex* y llega hasta *endIndex*, no incluido. Este parámetro es opcional.

Valor devuelto

Una cadena del texto del rango especificado.

Descripción

Método; recupera el rango de texto especificado. Si omite los parámetros opcionales *startIndex* y *endIndex*, se devolverá la cadena de texto completa. Si sólo especifica *startIndex*, el método devolverá la cadena que comienza en la ubicación de índice y termina al final del campo. Si especifica *startIndex* y *endIndex*, el método devolverá la cadena que comienza en *startIndex* y llega hasta *endIndex*, no incluido.

Ejemplo

El ejemplo siguiente obtiene el carácter o caracteres desde el quinto carácter hasta el final del campo de texto seleccionado:

```
var myText = fl.getDocumentDOM().selection[0].getTextString(4);
fl.trace(myText);
```

El ejemplo siguiente obtiene del cuarto al noveno carácter comenzando en el campo de texto seleccionado:

```
var myText = fl.getDocumentDOM().selection[0].getTextString(3, 9);
fl.trace(myText);
```

text.length

Disponibilidad

Flash MX 2004.

Uso

```
text.length
```

Descripción

Propiedad de sólo lectura; un entero que representa el número de caracteres del objeto de texto.

Ejemplo

El ejemplo siguiente devuelve el número de caracteres del texto seleccionado:

```
var textLength = fl.getDocumentDOM().selection[0].length;
```

text.lineType

Disponibilidad

Flash MX 2004.

Uso

```
text.lineType
```

Descripción

Propiedad; una cadena que define el tipo de línea. Los valores válidos son: `"single line"`, `"multiline"`, `"multiline no wrap"` y `"password"`.

Esta propiedad sólo funciona con texto dinámico o de entrada y genera una advertencia si se utiliza con texto estático. El valor de `"password"` sólo funciona con texto de entrada.

Ejemplo

El ejemplo siguiente define la propiedad `lineType` con el valor `"multiline no wrap"`:

```
fl.getDocumentDOM().selection[0].lineType = "multiline no wrap";
```

text.maxCharacters

Disponibilidad

Flash MX 2004.

Uso

```
text.maxCharacters
```

Descripción

Propiedad; un entero que especifica el número máximo de caracteres que el usuario puede introducir en este objeto de texto.

Esta propiedad sólo funciona con texto de entrada; si se utiliza con otros tipos de texto, la propiedad genera una advertencia.

Ejemplo

El ejemplo siguiente define el valor de la propiedad `maxCharacters` como `30`:

```
fl.getDocumentDOM().selection[0].maxCharacters = 30;
```

text.orientation

Disponibilidad

Flash MX 2004.

Uso

```
text.orientation
```

Descripción

Propiedad; una cadena que especifica la orientación del campo de texto. Los valores válidos son: `"horizontal"`, `"vertical left to right"` y `"vertical right to left"`.

Esta propiedad sólo funciona con texto estático; genera una advertencia si se utiliza con otros tipos de texto.

Ejemplo

El ejemplo siguiente define la propiedad de orientación como `"vertical right to left"`:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].orientati
  on = "vertical right to left";
```

text.renderAsHTML

Disponibilidad

Flash MX 2004.

Uso

```
text.renderAsHTML
```

Descripción

Propiedad; un valor booleano. Si el valor es `true`, Flash dibuja el texto como HTML e interpreta etiquetas HTML incorporadas.

Esta propiedad sólo funciona con texto dinámico o de entrada; genera una advertencia si se utiliza con otros tipos de texto.

Ejemplo

El ejemplo siguiente define la propiedad `renderAsHTML` como `true`:

```
fl.getDocumentDOM().selection[0].renderAsHTML = true;
```

text.scrollable

Disponibilidad

Flash MX 2004.

Uso

```
text.scrollable
```

Descripción

Propiedad; un valor booleano. Si el valor es `true`, el texto se puede desplazar.

Esta propiedad sólo funciona con texto dinámico o de entrada; genera una advertencia si se utiliza con texto estático.

Ejemplo

El ejemplo siguiente define la propiedad `scrollable` como `false`:

```
fl.getDocumentDOM().selection[0].scrollable = false;
```

text.selectable

Disponibilidad

Flash MX 2004.

Uso

```
text.selectable
```

Descripción

Propiedad; un valor booleano. Si el valor es `true`, el texto se puede seleccionar.

El texto de entrada siempre se puede seleccionar. Genera una advertencia cuando se define como `false` y se utiliza con texto de entrada.

Ejemplo

El ejemplo siguiente define la propiedad `selectable` como `true`:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].selectabl
  e = true;
```

text.selectionEnd

Disponibilidad

Flash MX 2004.

Uso

```
text.selectionEnd
```

Descripción

Propiedad; un entero basado en cero que especifica el final de una subselección de texto. Para más información, consulte `text.selectionStart`.

text.selectionStart

Disponibilidad

Flash MX 2004.

Uso

```
text.selectionStart
```

Descripción

Propiedad; un entero basado en cero que especifica el principio de una subselección de texto. Puede utilizar esta propiedad con `text.selectionEnd` para seleccionar un rango de caracteres. Se seleccionarán los caracteres hasta `text.selectionEnd`, no incluido. Véase `text.selectionEnd`.

- Si no hay un punto de inserción o una selección, `text.selectionEnd` es igual a `text.selectionStart`.
- Si se define `text.selectionStart` con un valor mayor que `text.selectionEnd`, `text.selectionEnd` se definirá como `text.selectionStart` y no se seleccionará texto.

Ejemplo

El ejemplo siguiente define el comienzo de la subselección de texto en el sexto carácter:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].selection
  Start = 5;
```

El ejemplo siguiente selecciona los caracteres “Barbara” de un campo de texto que contiene el texto “My name is Barbara” y les aplica formato de negrita y verde:

```
fl.getDocumentDOM().selection[0].selectionStart = 11;
fl.getDocumentDOM().selection[0].selectionEnd = 18;
var s = fl.getDocumentDOM().selection[0].selectionStart;
var e = fl.getDocumentDOM().selection[0].selectionEnd;
fl.getDocumentDOM().setElementTextAttr('bold', true, s, e);
fl.getDocumentDOM().setElementTextAttr("fillColor", "#00ff00", s, e);
```

text.setTextAttr()

Disponibilidad

Flash MX 2004.

Uso

```
text.setTextAttr(attrName, attrValue [, startIndex [, endIndex]])
```

Parámetros

attrName Una cadena que especifica el nombre de la propiedad del objeto TextAttrs que se va a cambiar.

attrValue El valor de la propiedad del objeto TextAttrs.

NOTA

Para obtener una lista de posibles valores de *attrName* y *attrValue*, consulte “Resumen de propiedades del objeto TextAttrs” en la página 464.

startIndex Un entero que es el índice (basado en cero) del primer carácter de la matriz. Este parámetro es opcional.

endIndex Un entero que especifica el índice del punto final de la cadena de texto seleccionada, que comienza por *startIndex* y llega hasta *endIndex*, no incluido. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; define el atributo especificado por el parámetro *attrName* asociado al texto identificado por *startIndex* y *endIndex* con el valor especificado por *attrValue*. Este método puede emplearse para cambiar atributos de texto que pueden comprender elementos TextRun (véase Objeto TextRun) o que son partes de elementos TextRun existentes. Su uso puede cambiar la posición y el número de elementos TextRun en la matriz `text.textRuns` de este objeto (véase `text.textRuns`).

Si omite los parámetros opcionales, el método utilizará el rango de caracteres de todo el objeto de texto. Si sólo especifica *startIndex*, el rango será el carácter que se encuentra en esa posición. Si especifica *startIndex* y *endIndex*, el rango comenzará en *startIndex* y llegará hasta el carácter situado en *endIndex*, no incluido.

Ejemplo

El ejemplo siguiente define el campo de texto seleccionado como cursiva:

```
fl.getDocumentDOM().selection[0].setTextAttr("italic", true);
```

El ejemplo siguiente define el tamaño del tercer carácter como 10:

```
fl.getDocumentDOM().selection[0].setTextAttr("size", 10, 2);
```

El ejemplo siguiente define el color como rojo para el rango del tercer al octavo carácter del texto seleccionado:

```
fl.getDocumentDOM().selection[0].setTextAttr("fillColor", 0xff0000, 2, 8);
```

text.setTextString()

Disponibilidad

Flash MX 2004.

Uso

```
text.setTextString(text [, startIndex [, endIndex]])
```

Parámetros

text Una cadena compuesta por los caracteres que se van a insertar en este objeto de texto.

startIndex Un entero que especifica el índice (basado en cero) del carácter en la cadena donde se insertará el texto. Este parámetro es opcional.

endIndex Un entero que especifica el índice el punto final en la cadena de texto seleccionada. El nuevo texto sobrescribe el texto de *startIndex* hasta *endIndex*, no incluido. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Propiedad; cambia la cadena de texto en este objeto de texto. Si omite los parámetros opcionales, se reemplazará todo el objeto de texto. Si sólo especifica *startIndex*, la cadena especificada se insertará en la posición *startIndex*. Si especifica *startIndex* y *endIndex*, la cadena especificada reemplazará el segmento de texto que comienza en *startIndex* y llega a *endIndex* (no incluido).

Ejemplo

El ejemplo siguiente asigna la cadena `"this is a string"` al campo de texto seleccionado:

```
fl.getDocumentDOM().selection[0].setTextString("this is a string");
```

El ejemplo siguiente inserta la cadena `"abc"` que comienza en el quinto carácter del campo de texto seleccionado:

```
fl.getDocumentDOM().selection[0].setTextString("01234567890");
fl.getDocumentDOM().selection[0].setTextString("abc", 4);
// el campo de texto es ahora "0123abc4567890"
```

El ejemplo siguiente reemplaza el texto del rango que va del tercer al octavo carácter de la cadena de texto seleccionada por la cadena `"abcdefghij"`. Se sobrescribirán los caracteres entre *`startIndex`* y *`endIndex`*. Los caracteres que comienzan con *`endIndex`* siguen a la cadena insertada.

```
fl.getDocumentDOM().selection[0].setTextString("01234567890");
fl.getDocumentDOM().selection[0].setTextString("abcdefghij", 2, 8);
// el campo de texto es ahora 01abcdefghij890"
```

text.shortcut

Disponibilidad

Flash MX 2004.

Uso

```
text.shortcut
```

Descripción

Propiedad; una cadena que equivale al campo Método abreviado del panel Accesibilidad. El lector de pantalla lee este método abreviado. Esta propiedad no se puede utilizar con texto dinámico.

Ejemplo

El ejemplo siguiente obtiene la tecla de método abreviado del objeto seleccionado y muestra el valor:

```
var theShortcut = fl.getDocumentDOM().selection[0].shortcut;
fl.trace(theShortcut);
```

El ejemplo siguiente define la tecla de método abreviado del objeto seleccionado:

```
fl.getDocumentDOM().selection[0].shortcut = "Ctrl+i";
```

text.silent

Disponibilidad

Flash MX 2004.

Uso

```
text.silent
```

Descripción

Propiedad; un valor booleano que especifica si el objeto es accesible. Equivale a la lógica inversa de la opción Hacer que el objeto sea accesible del panel Accesibilidad. Es decir, si `silent` es `true`, estará desactivada la opción Hacer que el objeto sea accesible. Si es `false`, estará activada la opción Hacer que el objeto sea accesible.

Ejemplo

El ejemplo siguiente determina si el objeto es accesible (un valor de `false` significa que es accesible):

```
var isSilent =
  fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].silent
  ;
```

El ejemplo siguiente define el objeto como accesible:

```
fl.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].silent =
  false;
```

text.tabIndex

Disponibilidad

Flash MX 2004.

Uso

```
text.tabIndex
```

Descripción

Propiedad; un entero que equivale al campo Índice de fichas del panel Accesibilidad. This value lets you determine the order in which objects are accessed when the user presses the Tab key.

Ejemplo

El ejemplo siguiente obtiene el `tabIndex` del objeto seleccionado actualmente:

```
var theTabIndex = fl.getDocumentDOM().selection[0].tabIndex;
```

El ejemplo siguiente define el `tabIndex` del objeto seleccionado actualmente:

```
fl.getDocumentDOM().selection[0].tabIndex = 1;
```

text.textRuns

Disponibilidad

Flash MX 2004.

Uso

```
text.textRuns
```

Descripción

Propiedad de sólo lectura; una matriz de objetos TextRun (véase Objeto TextRun).

Ejemplo

El ejemplo siguiente almacena el valor de la propiedad `textRuns` en la variable `myTextRuns`:

```
var myTextRuns = fl.getDocumentDOM().selection[0].textRuns;
```

text.textType

Disponibilidad

Flash MX 2004.

Uso

```
text.textType
```

Descripción

Propiedad; una cadena que especifica el tipo del campo de texto. Los valores válidos son: `"static"`, `"dynamic"` e `"input"`.

Ejemplo

El ejemplo siguiente define la propiedad `textType` como `"input"`:

```
fl.getDocumentDOM().selection[0].textType = "input";
```

text.useDeviceFonts

Disponibilidad

Flash MX 2004.

Uso

```
text.useDeviceFonts
```

Descripción

Propiedad; un valor booleano. Un valor de `true` hace que Flash dibuje el texto utilizando fuentes del dispositivo.

Ejemplo

El ejemplo siguiente hace que Flash utilice fuentes del dispositivo al dibujar texto.

```
fl.getDocumentDOM().selection[0].useDeviceFonts = true;
```

text.variableName

Disponibilidad

Flash MX 2004.

Uso

```
text.variableName
```

Descripción

Propiedad; una cadena que contiene el nombre de la variable asociada al objeto de texto. Esta propiedad sólo funciona con texto dinámico o de entrada; genera una advertencia si se utiliza con otros tipos de texto.

Objeto TextAttrs

Disponibilidad

Flash MX 2004.

Descripción

El objeto TextAttrs contiene todas las propiedades de texto que se pueden aplicar a una subselección. Este objeto es una propiedad del objeto TextRun (`textRun.textAttrs`).

Resumen de propiedades del objeto TextAttrs

Las propiedades siguientes están disponibles para el objeto TextAttrs.

Propiedad	Descripción
`textAttrs.aliasText`	Un valor booleano que especifica que Flash debe dibujar el texto empleando un método optimizado para aumentar la legibilidad del texto pequeño.
`textAttrs.alignment`	Una cadena que especifica la justificación del párrafo. Los valores válidos son: `"left"`, `"center"`, `"right"` y `"justify"`.
`textAttrs.autoKern`	Un valor booleano que determina si Flash utiliza (`true`) o ignora (`false`) la información de ajuste entre caracteres en las fuentes para justificar el texto.
`textAttrs.bold`	Valor booleano. Un valor de `true` hace que el texto aparezca con la versión en negrita de la fuente.
`textAttrs.characterPosition`	Una cadena que determina la línea base del texto.
`textAttrs.characterSpacing`	No admitida a favor de `textAttrs.letterSpacing`. Un entero que representa el espacio entre caracteres.
`textAttrs.face`	Una cadena que representa el nombre de la fuente, por ejemplo, `"Arial"`.
`textAttrs.fillColor`	Una cadena, valor hexadecimal o entero que representa el color de relleno.
`textAttrs.indent`	Un entero que especifica el sangrado del párrafo.
`textAttrs.italic`	Valor booleano. Un valor de `true` hace que el texto aparezca con la versión en cursiva de la fuente.
`textAttrs.leftMargin`	Un entero que especifica el margen izquierdo del párrafo.
`textAttrs.letterSpacing`	Un entero que representa el espacio entre caracteres.

Propiedad	Descripción
textAttrs.lineSpacing	Un entero que especifica el espacio de línea (inicial) del párrafo.
textAttrs.rightMargin	Un entero que especifica el margen derecho del párrafo.
textAttrs.rotation	Valor booleano. Un valor de `true` hace que Flash gire los caracteres del texto 90º. El valor predeterminado es `false`.
textAttrs.size	Un entero que especifica el tamaño de la fuente.
textAttrs.target	Una cadena que representa la propiedad `target` del campo de texto.
textAttrs.url	Una cadena que representa la propiedad `URL` del campo de texto.

textAttrs.aliasText

Disponibilidad

Flash MX 2004.

Uso

```
textAttrs.aliasText
```

Descripción

Propiedad; un valor booleano que especifica que Flash debe dibujar el texto empleando un método optimizado para aumentar la legibilidad del texto pequeño.

Ejemplo

El ejemplo siguiente define la propiedad `aliasText` como `true` para todo el texto del campo de texto seleccionado actualmente:

```
fl.getDocumentDOM().setElementTextAttr('aliasText', true);
```

textAttrs.alignment

Disponibilidad

Flash MX 2004.

Uso

```
textAttrs.alignment
```

Descripción

Propiedad; una cadena que especifica la justificación del párrafo. Los valores válidos son: `"left"`, `"center"`, `"right"` y `"justify"`.

Ejemplo

El ejemplo siguiente define la justificación de los párrafos que contienen caracteres entre el índice 0 y el índice 3, no incluido. Puede afectar a los caracteres fuera del rango especificado si se encuentran en el mismo párrafo.

```
fl.getDocumentDOM().setTextSelection(0, 3);
fl.getDocumentDOM().setElementTextAttr('alignment', 'justify');
```

textAttrs.autoKern

Disponibilidad

Flash MX 2004.

Uso

```
textAttrs.autoKern
```

Descripción

Propiedad; un valor booleano que determina si Flash utiliza (`true`) o ignora (`false`) la información de ajuste entre caracteres en las fuentes cuando se justifica el texto.

Ejemplo

El ejemplo siguiente selecciona los caracteres desde el índice 2 hasta el índice 6, no incluido, y define la propiedad `autoKern` como `true`:

```
fl.getDocumentDOM().setTextSelection(3, 6);
fl.getDocumentDOM().setElementTextAttr('autoKern', true);
```

textAttrs.bold

Disponibilidad

Flash MX 2004.

Uso

```
textAttrs.bold
```

Descripción

Propiedad; un valor booleano. Un valor de `true` hace que el texto aparezca con la versión en negrita de la fuente.

Ejemplo

El ejemplo siguiente selecciona el primer carácter del objeto de texto seleccionado y define la propiedad `bold` como `true`:

```
fl.getDocumentDOM().setTextSelection(0, 1);
fl.getDocumentDOM().setElementTextAttr('bold', true);
```

textAttrs.characterPosition

Disponibilidad

Flash MX 2004.

Uso

```
textAttrs.characterPosition
```

Descripción

Propiedad; una cadena que determina la línea base del texto. Los valores válidos son: `"normal"`, `"subscript"` y `"superscript"`. Esta propiedad sólo se aplica a texto estático.

Ejemplo

El ejemplo siguiente selecciona los caracteres desde el índice 2 hasta el índice 6, no incluido, del texto seleccionado y define la propiedad `characterPosition` como `"subscript"`:

```
fl.getDocumentDOM().setTextSelection(2, 6);
fl.getDocumentDOM().setElementTextAttr("characterPosition", "subscript");
```

textAttrs.characterSpacing

Disponibilidad

Flash MX 2004. No admitido en Flash 8 a favor de `textAttrs.letterSpacing`.

Uso

```
textAttrs.characterSpacing
```

Descripción

Propiedad; un entero que representa el espacio entre caracteres. Los valores válidos van de -60 a 60.

Esta propiedad sólo se aplica a texto estático; genera una advertencia si se utiliza con otros tipos de texto.

Ejemplo

El ejemplo siguiente define el espaciado de caracteres del campo de texto seleccionado como 10:

```
fl.getDocumentDOM().setElementTextAttr("characterSpacing", 10);
```

textAttrs.face

Disponibilidad

Flash MX 2004.

Uso

```
textAttrs.face
```

Descripción

Propiedad; una cadena que representa el nombre de la fuente, por ejemplo, `"Arial"`.

Ejemplo

El ejemplo siguiente define como `"Arial"` la fuente del campo de texto seleccionado desde el carácter del índice 2 hasta el carácter del índice 8, no incluido:

```
fl.getDocumentDOM().selection[0].setTextAttr("face", "Arial", 2, 8);
```

textAttrs.fillColor

Disponibilidad

Flash MX 2004.

Uso

```
textAttrs.fillColor
```

Descripción

Propiedad; el color del relleno, en uno de los formatos siguientes:

- Una cadena con el formato `"#RRGGBB"` o `"#RRGGBBAA"`
- Un número hexadecimal con el formato `0xRRGGBB`
- Un entero que representa el equivalente decimal del número hexadecimal

Ejemplo

El ejemplo siguiente define como rojo el color del campo de texto seleccionado desde el carácter del índice 2 hasta el carácter del índice 8, no incluido:

```
fl.getDocumentDOM().selection[0].setTextAttr("fillColor", 0xff0000, 2, 8);
```

textAttrs.indent

Disponibilidad

Flash MX 2004.

Uso

```
textAttrs.indent
```

Descripción

Propiedad; un entero que especifica el sangrado del párrafo. Los valores válidos van de -720 a 720.

Ejemplo

El ejemplo siguiente define como 100 el sangrado del campo de texto seleccionado desde el carácter del índice 2 hasta el carácter del índice 8, no incluido. Puede afectar a los caracteres fuera del rango especificado si se encuentran en el mismo párrafo.

```
fl.getDocumentDOM().selection[0].setTextAttr("indent", 100, 2, 8);
```

textAttrs.italic

Disponibilidad

Flash MX 2004.

Uso

```
textAttrs.italic
```

Descripción

Propiedad; un valor booleano. Un valor de `true` hace que el texto aparezca con la versión en cursiva de la fuente.

Ejemplo

El ejemplo siguiente define el campo de texto seleccionado como cursiva:

```
fl.getDocumentDOM().selection[0].setTextAttr("italic", true);
```

textAttrs.leftMargin

Disponibilidad

Flash MX 2004.

Uso

```
textAttrs.leftMargin
```

Descripción

Propiedad; un entero que especifica el margen izquierdo del párrafo. Los valores válidos van de 0 a 720.

Ejemplo

El ejemplo siguiente define como 100 la propiedad `leftMargin` del campo de texto seleccionado desde el carácter del índice 2 hasta el carácter del índice 8, no incluido. Puede afectar a los caracteres fuera del rango especificado si se encuentran en el mismo párrafo.

```
fl.getDocumentDOM().selection[0].setTextAttr("leftMargin", 100, 2, 8);
```

textAttrs.letterSpacing

Disponibilidad

Flash 8.

Uso

```
textAttrs.letterSpacing
```

Descripción

Propiedad; un entero que representa el espacio entre caracteres. Los valores válidos van de -60 a 60.

Esta propiedad sólo se aplica a texto estático; genera una advertencia si se utiliza con otros tipos de texto.

Ejemplo

El código siguiente selecciona los caracteres desde el índice 0 hasta el índice 10, éste no incluido, y define el espaciado de caracteres en 60:

```
fl.getDocumentDOM().setTextSelection(0, 10);
fl.getDocumentDOM().setElementTextAttr("letterSpacing", 60);
```

textAttrs.lineSpacing

Disponibilidad

Flash MX 2004.

Uso

```
textAttrs.lineSpacing
```

Descripción

Propiedad; un entero que especifica el espacio de línea (*inicial*) del párrafo. Los valores válidos van de -360 a 720.

Ejemplo

El ejemplo siguiente define como 100 la propiedad `lineSpacing` del campo de texto seleccionado:

```
fl.getDocumentDOM().selection[0].setTextAttr("lineSpacing", 100);
```

textAttrs.rightMargin

Disponibilidad

Flash MX 2004.

Uso

```
textAttrs.rightMargin
```

Descripción

Propiedad; un entero que especifica el margen derecho del párrafo. Los valores válidos van de 0 a 720.

Ejemplo

El ejemplo siguiente define como 100 la propiedad `rightMargin` del campo de texto seleccionado desde el carácter del índice 2 hasta el carácter del índice 8, no incluido. Puede afectar a los caracteres fuera del rango especificado si se encuentran en el mismo párrafo.

```
fl.getDocumentDOM().selection[0].setTextAttr("rightMargin", 100, 2, 8);
```

textAttrs.rotation

Disponibilidad

Flash MX 2004.

Uso

```
textAttrs.rotation
```

Descripción

Propiedad; un valor booleano. Un valor de `true` hace que Flash gire los caracteres del texto 90º. El valor predeterminado es `false`. Esta propiedad sólo se aplica a texto estático con orientación vertical; genera una advertencia si se utiliza con otros tipos de texto.

Ejemplo

El ejemplo siguiente define la rotación del campo de texto seleccionado como `true`:

```
fl.getDocumentDOM().setElementTextAttr("rotation", true);
```

textAttrs.size

Disponibilidad

Flash MX 2004.

Uso

```
textAttrs.size
```

Descripción

Propiedad; un entero que especifica el tamaño de la fuente.

Ejemplo

El ejemplo siguiente recupera el tamaño del carácter en el índice 2 y muestra el resultado en el panel Salida:

```
fl.outputPanel.trace(fl.getDocumentDOM().selection[0].getTextAttr("size",
  2));
```

textAttrs.target

Disponibilidad

Flash MX 2004.

Uso

```
textAttrs.target
```

Descripción

Propiedad; una cadena que representa la propiedad `target` del campo de texto.
Esta propiedad sólo funciona con texto estático.

Ejemplo

El ejemplo siguiente obtiene la propiedad `target` del campo de texto del primer fotograma de la capa superior de la escena actual y la muestra en el panel Salida:

```
fl.outputPanel.trace(fl.getDocumentDOM().getTimeline().layers[0].frames[0].
  elements[0].getTextAttr("target"));
```

textAttrs.url

Disponibilidad

Flash MX 2004.

Uso

```
textAttrs.url
```

Descripción

Propiedad; una cadena que representa la propiedad `URL` del campo de texto. Esta propiedad sólo funciona con texto estático.

Ejemplo

El ejemplo siguiente define la URL del campo de texto seleccionado como http://www.macromedia.com/es:

```
fl.getDocumentDOM().setElementTextAttr("url", "http://www.macromedia.com/
  es");
```

Objeto TextRun

Disponibilidad

Flash MX 2004.

Descripción

El objeto TextRun representa una serie de caracteres que tienen atributos que coinciden con todas las propiedades del Objeto TextAttrs. Este objeto es una propiedad del objeto Text (`text.textRuns`).

Resumen de propiedades del objeto TextRun

Además de las propiedades disponibles para utilizar con el Objeto Text, el objeto TextRun suministra las siguientes.

Propiedad	Descripción
`textRun.characters`	Una cadena que representa el texto que contiene el objeto TextRun.
`textRun.textAttrs`	El objeto TextAttrs que contiene los atributos de la extensión de texto.

textRun.characters

Disponibilidad

Flash MX 2004.

Uso

```
textRun.characters
```

Descripción

Propiedad; el texto que contiene el objeto TextRun.

Ejemplo

El ejemplo siguiente muestra los caracteres que componen la primera extensión de caracteres del campo de texto seleccionado en el panel Salida.

```
fl.trace(fl.getDocumentDOM().selection[0].textRuns[0].characters);
```

textRun.textAttrs

Disponibilidad

Flash MX 2004.

Uso

```
textRun.textAttrs
```

Descripción

Propiedad; el Objeto TextAttrs que contiene los atributos de la extensión de texto.

Ejemplo

El ejemplo siguiente muestra las propiedades de la primera extensión de caracteres del campo de texto seleccionado en el panel Salida.

```
var curTextAttrs = fl.getDocumentDOM().selection[0].textRuns[0].textAttrs;
for (var prop in curTextAttrs) {
  fl.trace(prop + " = " + curTextAttrs[prop]);
}
```

Objeto Timeline

Disponibilidad

Flash MX 2004.

Descripción

El objeto Timeline representa la línea de tiempo de Flash, a la que puede acceder mediante `fl.getDocumentDOM().getTimeline()` para el documento actual. Este método devuelve la línea de tiempo de la escena o el símbolo actual que se está editando.

Cuando se trabaja con escenas, la línea de tiempo de cada escena tiene un valor de índice al que puede acceder para el documento actual `fl.getDocumentDOM().timelines[i]`. (En este ejemplo, `i` es el índice del valor de la línea de tiempo.)

Cuando trabaje con fotogramas empleando los métodos y las propiedades del objeto Timeline, recuerde que el valor del fotograma es un índice basado en cero (y no el número del fotograma en la secuencia de fotogramas de la línea de tiempo). Es decir, el primer fotograma tiene un índice de fotograma de 0.

Resumen de métodos del objeto Timeline

Los métodos siguientes están disponibles para el objeto Timeline.

Método	Descripción
`timeline.addMotionGuide()`	Añade una capa de guía de movimiento sobre la capa actual y asocia la capa actual a la capa de guía añadida.
`timeline.addNewLayer()`	Añade una nueva capa al documento y la convierte en la capa actual.
`timeline.clearFrames()`	Elimina todo el contenido de un fotograma o un intervalo de fotogramas de la capa actual.
`timeline.clearKeyframes()`	Convierte un fotograma clave en un fotograma normal y elimina su contenido en la capa actual.
`timeline.convertToBlankKeyframes()`	Convierte los fotogramas en fotogramas clave en blanco en la capa actual.
`timeline.convertToKeyframes()`	Convierte un intervalo de fotogramas en fotogramas clave (o convierte la selección si no se especifican fotogramas) en la capa actual.
`timeline.copyFrames()`	Copia un intervalo de fotogramas de la capa actual en el Portapapeles.

Método	Descripción
`timeline.createMotionTween()`	Define la propiedad `frame.tweenType` como `motion` para cada fotograma clave seleccionado en la capa actual y, si es necesario, convierte el contenido de cada fotograma en una instancia de símbolo único.
`timeline.cutFrames()`	Corta un intervalo de fotogramas de la capa actual en la línea de tiempo y lo guarda en el Portapapeles.
`timeline.deleteLayer()`	Elimina una capa.
`timeline.expandFolder()`	Expande o contrae la carpeta o carpetas especificadas.
`timeline.findLayerIndex()`	Busca una matriz de índices para las capas con un determinado nombre.
`timeline.getFrameProperty()`	Recupera el valor de la propiedad especificada para los fotogramas seleccionados.
`timeline.getLayerProperty()`	Recupera el valor de la propiedad especificada para las capas seleccionadas.
`timeline.getSelectedFrames()`	Recupera los fotogramas seleccionados en una matriz.
`timeline.getSelectedLayers()`	Recupera los valores de índice basado en cero de las capas seleccionadas actualmente.
`timeline.insertBlankKeyframe()`	Inserta un fotograma clave en blanco en el índice de fotograma especificado; si no se especifica el índice, inserta un fotograma clave en blanco utilizando la cabeza lectora/selección.
`timeline.insertFrames()`	Inserta el número especificado de fotogramas en un determinado número de fotograma.
`timeline.insertKeyframe()`	Inserta un fotograma clave en el fotograma especificado.
`timeline.pasteFrames()`	Pega el intervalo de fotogramas del Portapapeles a los fotogramas especificados.
`timeline.removeFrames()`	Elimina el fotograma.
`timeline.reorderLayer()`	Mueve la primera capa especificada delante o detrás de la segunda capa especificada.
`timeline.reverseFrames()`	Invierte un intervalo de fotogramas.
`timeline.selectAllFrames()`	Selecciona todos los fotogramas de la línea de tiempo actual.

Método	Descripción
`timeline.setFrameProperty()`	Establece la propiedad del objeto Frame para los fotogramas seleccionados.
`timeline.setLayerProperty()`	Define con un valor determinado la propiedad especificada en todas las capas seleccionadas
`timeline.setSelectedFrames()`	Selecciona un intervalo de fotogramas de la capa actual o define los fotogramas seleccionadas con la matriz de selección transferida a este método.
`timeline.setSelectedLayers()`	Define la capa que se va a seleccionar; además, convierte la capa especificada en la capa actual.
`timeline.showLayerMasking()`	Muestra el enmascaramiento de capas durante la edición bloqueando la máscara y las capas enmascaradas.

Resumen de propiedades del objeto Timeline

Los métodos siguientes están disponibles para el objeto Timeline.

Propiedad	Descripción
`timeline.currentFrame`	Un índice basado en cero para el fotograma en la ubicación actual de la cabeza lectora.
`timeline.currentLayer`	Un índice basado en cero para la capa activa actual.
`timeline.frameCount`	De sólo lectura; un entero que representa el número de fotogramas en la capa más larga de esta línea de tiempo.
`timeline.layerCount`	De sólo lectura; un entero que representa el número de capas en la línea de tiempo especificada.
`timeline.layers`	De sólo lectura; una matriz de objetos Layer.
`timeline.name`	Una cadena que representa el nombre de la línea de tiempo actual.

timeline.addMotionGuide()

Disponibilidad

Flash MX 2004.

Uso

```
timeline.addMotionGuide()
```

Parámetros

Ninguno.

Valor devuelto

Un entero que representa el índice basado en cero de la capa de guía añadida. Si la capa actual no es de tipo “Normal”, Flash devuelve -1.

Descripción

Método; añade una capa de guía de movimiento sobre la capa actual y asocia la capa actual a la capa de guía recién añadida, convirtiendo la capa actual en una capa “Con guía”.

Este método sólo funciona en una capa de tipo “Normal”. No tiene ningún efecto sobre una capa de tipo “Carpeta”, “Máscara”, “Enmascarada”, “Guía” o “Con guía”.

Ejemplo

El ejemplo siguiente añade una capa de guía de movimiento sobre la capa actual y convierte la capa actual en `"Con guía"`:

```
fl.getDocumentDOM().getTimeline().addMotionGuide();
```

timeline.addNewLayer()

Disponibilidad

Flash MX 2004.

Uso

```
timeline.addNewLayer([name] [, layerType [, bAddAbove]])
```

Parámetros

`name` Una cadena que especifica el nombre de la nueva capa. Si omite este parámetro, se asignará un nombre de capa nuevo y predeterminado a la nueva capa (“Capa n”, donde *n* es el número total de capas). Este parámetro es opcional.

`layerType` Una cadena que especifica el tipo de capa que se va a añadir. Si omite este parámetro, se creará una capa de tipo “Normal”. Este parámetro es opcional.

bAddAbove Un valor booleano que, si se define como `true` (valor predeterminado), hace que Flash añada la nueva capa sobre la capa actual; `false` hace que Flash añada la capa debajo de la capa actual. Este parámetro es opcional.

Valor devuelto

Un valor entero del índice basado en cero de la capa recién añadida.

Descripción

Método; añade una nueva capa al documento y la convierte en la capa actual.

Ejemplo

El ejemplo siguiente añade una nueva capa a la línea de tiempo con un nombre predeterminado generado por Flash:

```
fl.getDocumentDOM().getTimeline().addNewLayer();
```

El ejemplo siguiente añade una nueva capa de carpeta sobre la capa actual y le asigna el nombre `"Folder1"`:

```
fl.getDocumentDOM().getTimeline().addNewLayer("Folder1", "folder", true);
```

timeline.clearFrames()

Disponibilidad

Flash MX 2004.

Uso

```
timeline.clearFrames([startFrameIndex [, endFrameIndex]])
```

Parámetros

startFrameIndex Un índice basado en cero que define el comienzo del intervalo de fotogramas que se va a borrar. Si omite *startFrameIndex*, el método utilizará la selección actual. Este parámetro es opcional.

endFrameIndex Un índice basado en cero que define el final del intervalo de fotogramas que se va a borrar. El intervalo llega hasta *endFrameIndex*, no incluido. Si sólo especifica *startFrameIndex*, *endFrameIndex* utilizará de forma predeterminada el valor de *startFrameIndex*. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; elimina todo el contenido de un fotograma o un intervalo de fotogramas de la capa actual.

Ejemplo

El ejemplo siguiente borra los fotogramas desde el Fotograma 6 hasta el 11, no incluido (recuerde que los valores de índice son distintos de los valores de número de fotograma):

```
fl.getDocumentDOM().getTimeline().clearFrames(5, 10);
```

El ejemplo siguiente borra el Fotograma 15:

```
fl.getDocumentDOM().getTimeline().clearFrames(14);
```

timeline.clearKeyframes()

Disponibilidad

Flash MX 2004.

Uso

```
timeline.clearKeyframes([startFrameIndex [, endFrameIndex]])
```

Parámetros

`startFrameIndex` Un índice basado en cero que define el comienzo del intervalo de fotogramas que se va a borrar. Si omite *`startFrameIndex`*, el método utilizará la selección actual. Este parámetro es opcional.

`endFrameIndex` Un índice basado en cero que define el final del intervalo de fotogramas que se va a borrar. El intervalo llega hasta *`endFrameIndex`*, no incluido. Si sólo especifica *`startFrameIndex`*, *`endFrameIndex`* utilizará de forma predeterminada el valor de *`startFrameIndex`*. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; convierte un fotograma clave en un fotograma normal y elimina su contenido en la capa actual.

Ejemplo

El ejemplo siguiente borra los fotogramas clave desde el Fotograma 5 hasta el 10, no incluido (recuerde que los valores de índice son distintos de los valores de número de fotograma):

```
fl.getDocumentDOM().getTimeline().clearKeyframes(4, 9);
```

El ejemplo siguiente borra el fotograma clave en el Fotograma 15 y lo convierte en un fotograma normal:

```
fl.getDocumentDOM().getTimeline().clearKeyframes(14);
```

timeline.convertToBlankKeyframes()

Disponibilidad

Flash MX 2004.

Uso

```
timeline.convertToBlankKeyframes([startFrameIndex [, endFrameIndex]])
```

Parámetros

`startFrameIndex` Un índice basado en cero que especifica el fotograma inicial para convertir en fotogramas clave. Si omite *`startFrameIndex`*, el método convertirá los fotogramas seleccionados actualmente. Este parámetro es opcional.

`endFrameIndex` Un índice basado en cero que especifica el fotograma en el que se detendrá la conversión en fotogramas clave. El intervalo de fotogramas para convertir llega hasta *`endFrameIndex`*, no incluido. Si sólo especifica *`startFrameIndex`*, *`endFrameIndex`* utilizará de forma predeterminada el valor de *`startFrameIndex`*. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; convierte los fotogramas en fotogramas clave en blanco en la capa actual.

Ejemplo

El ejemplo siguiente convierte desde el Fotograma 2 hasta el Fotograma 10, no incluido, en fotogramas clave en blanco (recuerde que los valores de índice son distintos de los valores de número de fotograma):

```
fl.getDocumentDOM().getTimeline().convertToBlankKeyframes(1, 9);
```

El ejemplo siguiente convierte el Fotograma 5 en un fotograma clave en blanco:

```
fl.getDocumentDOM().getTimeline().convertToBlankKeyframes(4);
```

timeline.convertToKeyframes()

Disponibilidad

Flash MX 2004.

Uso

```
timeline.convertToKeyframes([startFrameIndex [, endFrameIndex]])
```

Parámetros

startFrameIndex Un índice basado en cero que especifica el primer fotograma para convertir en fotogramas clave. Si omite *startFrameIndex*, el método convertirá los fotogramas seleccionados actualmente. Este parámetro es opcional.

endFrameIndex Un índice basado en cero que especifica el fotograma en el que se detendrá la conversión en fotogramas clave. El intervalo de fotogramas para convertir llega hasta *endFrameIndex*, no incluido. Si sólo especifica *startFrameIndex*, *endFrameIndex* utilizará de forma predeterminada el valor de *startFrameIndex*. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; convierte un intervalo de fotogramas en fotogramas clave (o convierte la selección si no se especifican fotogramas) en la capa actual.

Ejemplo

El ejemplo siguiente convierte los fotogramas seleccionados en fotogramas clave:

```
fl.getDocumentDOM().getTimeline().convertToKeyframes();
```

El ejemplo siguiente convierte los fotogramas clave desde el Fotograma 2 hasta el 10, no incluido (recuerde que los valores de índice son distintos de los valores de número de fotograma):

```
fl.getDocumentDOM().getTimeline().convertToKeyframes(1, 9);
```

El ejemplo siguiente convierte el Fotograma 5 en un fotograma clave:

```
fl.getDocumentDOM().getTimeline().convertToKeyframes(4);
```

timeline.copyFrames()

Disponibilidad

Flash MX 2004.

Uso

```
timeline.copyFrames([startFrameIndex [, endFrameIndex]])
```

Parámetros

startFrameIndex Un índice basado en cero que especifica el comienzo del intervalo de fotogramas que se va a copiar. Si omite *startFrameIndex*, el método utilizará la selección actual. Este parámetro es opcional.

endFrameIndex Un índice basado en cero que especifica el fotograma en el que se dejará de copiar. El intervalo de fotogramas para copiar llega hasta *endFrameIndex*, no incluido. Si sólo especifica *startFrameIndex*, *endFrameIndex* utilizará de forma predeterminada el valor de *startFrameIndex*. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; copia un intervalo de fotogramas de la capa actual en el Portapapeles.

Ejemplo

El ejemplo siguiente copia los fotogramas seleccionados en el Portapapeles:

```
fl.getDocumentDOM().getTimeline().copyFrames();
```

El ejemplo siguiente copia desde el Fotograma 2 hasta el Fotograma 10, no incluido, en el Portapapeles (recuerde que los valores de índice son distintos de los valores de número de fotograma):

```
fl.getDocumentDOM().getTimeline().copyFrames(1, 9);
```

El ejemplo siguiente copia el Fotograma 5 en el Portapapeles:

```
fl.getDocumentDOM().getTimeline().copyFrames(4);
```

timeline.createMotionTween()

Disponibilidad

Flash MX 2004.

Uso

```
timeline.createMotionTween([startFrameIndex [, endFrameIndex]])
```

Parámetros

startFrameIndex Un índice basado en cero que especifica el fotograma inicial en el que se creará una interpolación de movimiento. Si omite *startFrameIndex*, el método utilizará la selección actual. Este parámetro es opcional.

endFrameIndex Un índice basado en cero que especifica el fotograma en el que se detendrá la interpolación de movimiento. El intervalo de fotogramas llega hasta *endFrameIndex*, no incluido. Si sólo especifica *startFrameIndex*, *endFrameIndex* utilizará de forma predeterminada el valor de *startFrameIndex*. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Define la propiedad `frame.tweenType` como `motion` para cada fotograma clave seleccionado en la capa actual y, si es necesario, convierte el contenido de cada fotograma en una instancia de símbolo único. Esta propiedad equivale al elemento de menú Crear interpolación de movimiento de la herramienta de edición de Flash.

Ejemplo

El ejemplo siguiente convierte la forma del primer fotograma hasta el Fotograma 10, no incluido, en una instancia de un símbolo gráfico y define `frame.tweenType` como `motion` (recuerde que los valores de índice son distintos de los valores de número de fotograma):

```
fl.getDocumentDOM().getTimeline().createMotionTween(0, 9);
```

timeline.currentFrame

Disponibilidad

Flash MX 2004.

Uso

```
timeline.currentFrame
```

Descripción

Propiedad; un índice basado en cero para el fotograma en la ubicación actual de la cabeza lectora.

Ejemplo

El ejemplo siguiente define la cabeza lectora de la línea de tiempo actual en el Fotograma 10 (recuerde que los valores de índice son distintos de los valores de número de fotograma):

```
fl.getDocumentDOM().getTimeline().currentFrame = 9;
```

El ejemplo siguiente almacena el valor de la ubicación de la cabeza lectora actual en la variable `curFrame`:

```
var curFrame = fl.getDocumentDOM().getTimeline().currentFrame;
```

timeline.currentLayer

Disponibilidad

Flash MX 2004.

Uso

```
timeline.currentLayer
```

Descripción

Propiedad; el índice basado en cero para la capa activa actual. Un valor de 0 especifica la capa superior, un valor de 1 especifica la capa que se encuentra por debajo, y así sucesivamente.

Ejemplo

El ejemplo siguiente convierte en activa la capa superior:

```
fl.getDocumentDOM().getTimeline().currentLayer = 0;
```

El ejemplo siguiente almacena el índice de la capa activa actualmente en la variable `curLayer`:

```
var curLayer = fl.getDocumentDOM().getTimeline().currentLayer;
```

timeline.cutFrames()

Disponibilidad

Flash MX 2004.

Uso

```
timeline.cutFrames([startFrameIndex [, endFrameIndex]])
```

Parámetros

startFrameIndex Un índice basado en cero que especifica el comienzo del intervalo de fotogramas que se va a cortar. Si omite *startFrameIndex*, el método utilizará la selección actual. Este parámetro es opcional.

endFrameIndex Un índice basado en cero que especifica el fotograma en el que se dejará de cortar. El intervalo de fotogramas llega hasta *endFrameIndex*, no incluido. Si sólo especifica *startFrameIndex*, *endFrameIndex* utilizará de forma predeterminada el valor de *startFrameIndex*. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; corta un intervalo de fotogramas de la capa actual en la línea de tiempo y lo guarda en el Portapapeles.

Ejemplo

El ejemplo siguiente corta los fotogramas seleccionados en la línea de tiempo y los guarda en el Portapapeles:

```
fl.getDocumentDOM().getTimeline().cutFrames();
```

El ejemplo siguiente corta desde el Fotograma 2 hasta el Fotograma 10, no incluido, en la línea de tiempo y los guarda en el Portapapeles (recuerde que los valores de índice son distintos de los valores de número de fotograma):

```
fl.getDocumentDOM().getTimeline().cutFrames(1, 9);
```

El ejemplo siguiente corta el Fotograma 5 en la línea de tiempo y lo guarda en el Portapapeles:

```
fl.getDocumentDOM().getTimeline().cutFrames(4);
```

timeline.deleteLayer()

Disponibilidad

Flash MX 2004.

Uso

```
timeline.deleteLayer([index])
```

Parámetros

index Un índice basado en cero que especifica la capa que se va a eliminar. Si sólo hay una capa en la línea de tiempo, este método no tiene ningún efecto. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; elimina una capa. Si la capa es una carpeta, se eliminarán todas las capas dentro de la carpeta. Si no especifica el índice de capa, Flash eliminará las capas seleccionadas actualmente.

Ejemplo

El ejemplo siguiente elimina la segunda capa desde la parte superior:

```
fl.getDocumentDOM().getTimeline().deleteLayer(1);
```

El ejemplo siguiente elimina las capas seleccionadas actualmente:

```
fl.getDocumentDOM().getTimeline().deleteLayer();
```

timeline.expandFolder()

Disponibilidad

Flash MX 2004.

Uso

```
timeline.expandFolder(bExpand [, bRecurseNestedParents [, index]])
```

Parámetros

bExpand Un valor booleano que, si se define como `true`, hace que el método expanda la carpeta; `false` hace que el método contraiga la carpeta.

bRecurseNestedParents Un valor booleano que, si se define como `true`, hace que todas las capas de la carpeta especificada se abran o cierren, según el parámetro *bExpand*. Este parámetro es opcional.

index Un índice basado en cero de la carpeta que se va a expandir o contraer. Utilice -1 para aplicar a todas las capas (deberá definir también *bRecurseNestedParents* como `true`). Esta propiedad equivale a los elementos de menú Expandir todas/Contraer todas de la herramienta de edición de Flash. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; expande o contrae la carpeta o carpetas especificadas. Si no especifica una carpeta, este método actúa sobre la capa actual.

Ejemplo

Los ejemplos siguientes utilizan esta estructura de carpetas:

```
Folder 1 ***
--layer 7
--Folder 2 ****
----Layer 5
```

El ejemplo siguiente expande sólo la Carpeta 1:

```
fl.getDocumentDOM().getTimeline().currentLayer = 1;
fl.getDocumentDOM().getTimeline().expandFolder(true);
```

El ejemplo siguiente expande sólo la Carpeta 1 (suponiendo que la Carpeta 2 se contrajo al contraerse por última vez la Carpeta 1; en caso contrario, la Carpeta 2 aparece expandida):

```
fl.getDocumentDOM().getTimeline().expandFolder(true, false, 0);
```

El ejemplo siguiente contrae todas las carpetas de la línea de tiempo actual:

```
fl.getDocumentDOM().getTimeline().expandFolder(false, true, -1);
```

timeline.findLayerIndex()

Disponibilidad

Flash MX 2004.

Uso

```
timeline.findLayerIndex(name)
```

Parámetros

name Una cadena que especifica el nombre de la carpeta que se va a buscar.

Valor devuelto

Una matriz de valores de índice para la capa especificada. Si no se encuentra la capa especificada, Flash devuelve `undefined`.

Descripción

Método; busca una matriz de índices para las capas con un determinado nombre. El índice de capa es plano, por lo que las carpetas se consideran parte del índice principal.

Ejemplo

El ejemplo siguiente muestra los valore de índice de todas las capas llamadas Capa 7 en el panel Salida:

```
var layerIndex = fl.getDocumentDOM().getTimeline().findLayerIndex("Layer
  7");
fl.trace(layerIndex);
```

El ejemplo siguiente ilustra cómo se transfieren los valores devueltos desde este método hasta `timeline.setSelectedLayers()`:

```
var layerIndex = fl.getDocumentDOM().getTimeline().findLayerIndex("Layer
  1");
fl.getDocumentDOM().getTimeline().setSelectedLayers(layerIndex[0], true);
```

timeline.frameCount

Disponibilidad

Flash MX 2004.

Uso

```
timeline.frameCount
```

Descripción

Propiedad de sólo lectura; un entero que representa el número de fotogramas en la capa más larga de esta línea de tiempo.

Ejemplo

El ejemplo siguiente utiliza una variable `countNum` para almacenar el número de fotogramas de la capa más larga del documento actual:

```
var countNum = fl.getDocumentDOM().getTimeline().frameCount;
```

timeline.getFrameProperty()

Disponibilidad

Flash MX 2004.

Uso

```
timeline.getFrameProperty(property [, startframeIndex [, endFrameIndex]])
```

Parámetros

property Una cadena que especifica el nombre de la propiedad para la que se va a obtener el valor. Consulte la lista completa de propiedades en "Resumen de propiedades del objeto Frame" en la página 279.

startFrameIndex Un índice basado en cero que especifica el número de fotograma inicial para el que se obtendrá el valor. Si omite *startFrameIndex*, el método utilizará la selección actual. Este parámetro es opcional.

endFrameIndex Un índice basado en cero que especifica el final del intervalo de fotogramas que se va a seleccionar. El intervalo llega hasta *endFrameIndex*, no incluido. Si sólo especifica *startFrameIndex*, *endFrameIndex* utilizará de forma predeterminada el valor de *startFrameIndex*. Este parámetro es opcional.

Valor devuelto

Un valor para la propiedad especificada o `undefined` si todos los fotogramas seleccionados no tienen el mismo valor de propiedad.

Descripción

Método; recupera el valor de la propiedad especificada para los fotogramas seleccionados.

Ejemplo

El ejemplo siguiente recupera el nombre del primer fotograma de la capa superior del documento actual y muestra el nombre en el panel Salida:

```
fl.getDocumentDOM().getTimeline().currentLayer = 0;
fl.getDocumentDOM().getTimeline().setSelectedFrames(0, 0, true);
var frameName = fl.getDocumentDOM().getTimeline().getFrameProperty("name");
fl.trace(frameName);
```

timeline.getLayerProperty()

Disponibilidad

Flash MX 2004.

Uso

```
timeline.getLayerProperty(property)
```

Parámetros

property Una cadena que especifica el nombre de la propiedad cuyo valor desea recuperar. Para ver una lista de propiedades, consulte "Resumen de propiedades del objeto Layer" en la página 311.

Valor devuelto

Valor de la propiedad especificada. Flash examina las propiedades de la capa para determinar el tipo. Si no todas las capas especificadas tienen el mismo valor de propiedad, Flash devuelve `undefined`.

Descripción

Método; recupera el valor de la propiedad especificada para las capas seleccionadas.

Ejemplo

El ejemplo siguiente recupera el nombre de la capa superior del documento actual y lo muestra en el panel Salida:

```
fl.getDocumentDOM().getTimeline().currentLayer = 0;
var layerName = fl.getDocumentDOM().getTimeline().getLayerProperty("name");
fl.trace(layerName);
```

timeline.getSelectedFrames()

Disponibilidad

Flash MX 2004.

Parámetros

Ninguno.

Valor devuelto

Una matriz que contiene 3*n* enteros, donde *n* es el número de regiones seleccionadas. El primer entero de cada grupo es el índice de capa, el segundo entero es el fotograma inicial del comienzo de la selección y el tercer entero especifica el fotograma final del intervalo de esa selección. El fotograma final no se incluye en la selección.

Descripción

Método; recupera los fotogramas seleccionados actualmente en una matriz.

Ejemplo

Con la capa superior como capa actual, el ejemplo siguiente muestra `0,5,10,0,20,25` en el panel Salida:

```
var timeline = fl.getDocumentDOM().getTimeline();
timeline.setSelectedFrames(5,10);
timeline.setSelectedFrames(20,25,false);
var theSelectedFrames = timeline.getSelectedFrames();
fl.trace(theSelectedFrames);
```

Véase también

`timeline.setSelectedFrames()`

timeline.getSelectedLayers()

Disponibilidad

Flash MX 2004.

Parámetros

Ninguno.

Valor devuelto

Una matriz de valores de índice basado en cero de las capas seleccionadas.

Descripción

Método; recupera los valores de índice basado en cero de las capas seleccionadas actualmente.

Ejemplo

El ejemplo siguiente muestra `1,0` en el panel Salida:

```
fl.getDocumentDOM().getTimeline().setSelectedLayers(0);
fl.getDocumentDOM().getTimeline().setSelectedLayers(1, false);
var layerArray = fl.getDocumentDOM().getTimeline().getSelectedLayers();
fl.trace(layerArray);
```

Véase también

`timeline.setSelectedLayers()`

timeline.insertBlankKeyframe()

Disponibilidad

Flash MX 2004.

Uso

```
timeline.insertBlankKeyframe([frameNumIndex])
```

Parámetros

frameNumIndex Un índice basado en cero que especifica el fotograma en el que se insertará el fotograma clave. Si omite *frameNumIndex*, el método utilizará el número de fotograma de la cabeza lectora actual. Este parámetro es opcional.

Si el fotograma especificado o seleccionado es normal, el fotograma clave se insertará en él. Por ejemplo, si tiene un intervalo de 10 fotogramas con números 1-10 y selecciona el Fotograma 5, este método convertirá el Fotograma 5 en un fotograma clave en blanco y la extensión del intervalo de fotogramas seguirá siendo 10 fotogramas. Si se selecciona el Fotograma 5 y es un fotograma clave con uno normal junto al mismo, este método inserta un fotograma clave en blanco en el Fotograma 6. Si el Fotograma 5 es un fotograma clave y el fotograma junto al mismo ya es uno clave, no se inserta ningún fotograma clave pero la cabeza lectora se desplaza al Fotograma 6.

Valor devuelto

Ninguno.

Descripción

Método; inserta un fotograma clave en blanco en el índice de fotograma especificado; si no se especifica el índice, el método inserta el fotograma clave en blanco utilizando la cabeza lectora/selección. Véase también `timeline.insertKeyframe()`.

Ejemplo

El ejemplo siguiente inserta un fotograma clave en blanco en el Fotograma 20 (recuerde que los valores de índice son distintos de los valores de número de fotograma):

```
fl.getDocumentDOM().getTimeline().insertBlankKeyframe(19);
```

El ejemplo siguiente inserta un fotograma clave en blanco en el fotograma seleccionado actualmente (o la ubicación de la cabeza lectora si no hay ningún fotograma seleccionado):

```
fl.getDocumentDOM().getTimeline().insertBlankKeyframe();
```

timeline.insertFrames()

Disponibilidad

Flash MX 2004.

Uso

```
timeline.insertFrames([numFrames [, bAllLayers [, frameNumIndex]]])
```

Parámetros

numFrames Un entero que especifica el número de fotogramas que se van a insertar. Si omite este parámetro, el método insertará fotogramas en la selección actual de la capa actual. Este parámetro es opcional.

bAllLayers Un valor booleano que, si se define como `true` (valor predeterminado), hace que el método inserte en todas las capas el número especificado de fotogramas en el parámetro *numFrames*; si se define como `false`, el método insertará fotogramas en la capa actual. Este parámetro es opcional.

frameNumIndex Un índice basado en cero que especifica el fotograma en el que se insertará un nuevo fotograma. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; inserta el número especificado de fotogramas en el índice especificado.

Si no se especifican parámetros, este método funciona de este modo:

- Si hay uno o varios fotogramas seleccionados, el método inserta el número de fotogramas seleccionado en la ubicación del primer fotograma seleccionado en la capa actual. Es decir, si se seleccionan los fotogramas 6 a 10 (un total de cinco fotogramas), el método añade cinco fotogramas al Fotograma 6 de la capa que contiene los fotogramas seleccionados.
- Si no hay fotogramas seleccionados, el método inserta un fotograma en el fotograma actual en todas las capas.

Si se especifican parámetros, el método funciona de este modo:

- Si sólo se especifica *numFrames*, inserta el número especificado de fotogramas en el fotograma actual en la capa actual.
- Si se especifica *numFrames* y *bAllLayers* es `true`, inserta el número especificado de fotogramas en el fotograma actual en todas las capas.

- Si se especifican los tres parámetros, inserta el número especificado de fotogramas en el índice especificado (*frameIndex*); el valor transferido para *bAllLayers* determina si los fotogramas sólo se añaden a la capa actual o a todas las capas.

Si el fotograma especificado o seleccionado es normal, el fotograma se insertará en él. Por ejemplo, si tiene un intervalo de 10 fotogramas con números 1-10 y selecciona el Fotograma 5 (o transfiere un valor de 4 para *frameIndex*), este método añadirá un fotograma en el Fotograma 5 y la extensión del intervalo de fotogramas será de 11. Si selecciona el Fotograma 5 y es un fotograma clave, este método insertará un fotograma en el Fotograma 6 con independencia de si el fotograma situado junto a él también es un fotograma clave.

Ejemplo

El ejemplo siguiente inserta un fotograma (o fotogramas, según la selección) en la ubicación actual de la capa actual:

```
fl.getDocumentDOM().getTimeline().insertFrames();
```

El ejemplo siguiente inserta cinco fotogramas en el fotograma actual en todas las capas:

```
fl.getDocumentDOM().getTimeline().insertFrames(5);
```

NOTA	Si tiene varias capas que contienen fotogramas y selecciona un fotograma en una capa cuando utiliza el comando anterior, Flash sólo insertará los fotogramas en la capa seleccionada. Si tiene varias capas sin fotogramas seleccionados en ellas, Flash insertará los fotogramas en todas las capas.

El ejemplo siguiente inserta tres fotogramas en la capa actual únicamente:

```
fl.getDocumentDOM().getTimeline().insertFrames(3, false);
```

El ejemplo siguiente inserta cuatro fotogramas en todas las capas, comenzando desde el primer fotograma:

```
fl.getDocumentDOM().getTimeline().insertFrames(4, true, 0);
```

timeline.insertKeyframe()

Disponibilidad

Flash MX 2004.

Uso

```
timeline.insertKeyframe([frameNumIndex])
```

Parámetros

frameNumIndex Un índice basado en cero que especifica el índice de fotograma en el que se insertará el fotograma clave en la capa actual. Si omite *frameNumIndex*, el método utilizará el número de fotograma de la cabeza lectora actual o el fotograma seleccionado. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; inserta un fotograma clave en el fotograma especificado. Si omite el parámetro, el método insertará un fotograma clave utilizando la ubicación de la cabeza lectora o de la selección.

Este método funciona igual que `timeline.insertBlankKeyframe()`, salvo que el fotograma clave insertado alberga el contenido del fotograma que ha convertido (es decir, no está en blanco).

Ejemplo

El ejemplo siguiente inserta un fotograma clave en la ubicación de la cabeza lectora o la selección:

```
fl.getDocumentDOM().getTimeline().insertKeyframe();
```

El ejemplo siguiente inserta un fotograma clave en el Fotograma 10 de la segunda capa (recuerde que los valores de índice son distintos de los valores de número de fotograma o capa):

```
fl.getDocumentDOM().getTimeline().currentLayer = 1;
fl.getDocumentDOM().getTimeline().insertKeyframe(9);
```

timeline.layerCount

Disponibilidad

Flash MX 2004.

Uso

```
timeline.layerCount
```

Descripción

Propiedad de sólo lectura; un entero que representa el número de capas en la línea de tiempo especificada.

Ejemplo

El ejemplo siguiente utiliza la variable `NumLayer` para almacenar el número de capas de la escena actual:

```
var NumLayer = fl.getDocumentDOM().getTimeline().layerCount;
```

timeline.layers

Disponibilidad

Flash MX 2004.

Uso

```
timeline.layers
```

Descripción

Propiedad de sólo lectura; una matriz de objetos Layer.

Ejemplo

El ejemplo siguiente utiliza la variable `currentLayers` para almacenar la matriz de objetos Layer del documento actual:

```
var currentLayers = fl.getDocumentDOM().getTimeline().layers;
```

timeline.name

Disponibilidad

Flash MX 2004.

Uso

```
timeline.name
```

Descripción

Propiedad; una cadena que especifica el nombre de la línea de tiempo actual. Este nombre es el nombre de la escena, pantalla (diapositiva o formulario) o símbolo que se está editando.

Ejemplo

El ejemplo siguiente recupera el nombre de la primera escena:

```
var sceneName = fl.getDocumentDOM().timelines[0].name;
```

El ejemplo siguiente define el nombre de la primera escena como `FirstScene`:

```
fl.getDocumentDOM().timelines[0].name = "FirstScene";
```

timeline.pasteFrames()

Disponibilidad

Flash MX 2004.

Uso

```
timeline.pasteFrames([startFrameIndex [, endFrameIndex]])
```

Parámetros

startFrameIndex Un índice basado en cero que especifica el comienzo del intervalo de fotogramas que se va a pegar. Si omite *startFrameIndex*, el método utilizará la selección actual. Este parámetro es opcional.

endFrameIndex Un índice basado en cero que especifica el fotograma en el que se dejará de pegar fotogramas. El método pega hasta *endFrameIndex*, no incluido. Si sólo especifica *startFrameIndex*, *endFrameIndex* utilizará de forma predeterminada el valor de *startFrameIndex*. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; pega el intervalo de fotogramas del Portapapeles a los fotogramas especificados.

Ejemplo

El ejemplo siguiente pega los fotogramas del Portapapeles al fotograma seleccionado o la ubicación de la cabeza lectora:

```
fl.getDocumentDOM().getTimeline().pasteFrames();
```

El ejemplo siguiente pega los fotogramas desde el Fotograma 2 hasta el 10, no incluido (recuerde que los valores de índice son distintos de los valores de número de fotograma):

```
fl.getDocumentDOM().getTimeline().pasteFrames(1, 9);
```

El ejemplo siguiente pega los fotogramas del Portapapeles comenzando en el Fotograma 5:

```
fl.getDocumentDOM().getTimeline().pasteFrames(4);
```

timeline.removeFrames()

Disponibilidad

Flash MX 2004.

Uso

```
timeline.removeFrames([startFrameIndex [, endFrameIndex]])
```

Parámetros

startFrameIndex Un índice basado en cero que especifica el primer fotograma en el que se iniciará la eliminación de fotogramas. Si omite *startFrameIndex*, el método utiliza la selección actual; si no hay selección, se eliminarán todos los fotogramas en la cabeza lectora actual en todas las capas. Este parámetro es opcional.

endFrameIndex Un índice basado en cero que especifica el fotograma en el que se dejará de eliminar fotogramas; el intervalo de fotogramas llega hasta *endFrameIndex*, no incluido. Si sólo especifica *startFrameIndex*, *endFrameIndex* utilizará de forma predeterminada el valor de *startFrameIndex*. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; elimina el fotograma.

Ejemplo

El ejemplo siguiente convierte desde el Fotograma 5 hasta el Fotograma 10 (no incluido) de la capa superior en la escena actual (recuerde que los valores de índice son distintos de los valores de número de fotograma):

```
fl.getDocumentDOM().getTimeline().currentLayer = 0;
fl.getDocumentDOM().getTimeline().removeFrames(4, 9);
```

El ejemplo siguiente elimina el Fotograma 8 en la capa superior de la escena actual:

```
fl.getDocumentDOM().getTimeline().currentLayer = 0;
fl.getDocumentDOM().getTimeline().removeFrames(7);
```

timeline.reorderLayer()

Disponibilidad

Flash MX 2004.

Uso

```
timeline.reorderLayer(layerToMove, layerToPutItBy [, bAddBefore])
```

Parámetros

layerToMove Un índice basado en cero que especifica qué capa se va a mover.

layerToPutItBy Un índice basado en cero que especifica junto a qué capa desea mover la capa. Por ejemplo, si especifica 1 para *layerToMove* y 0 para *layerToPutItBy*, la segunda capa se situará junto a la primera capa.

bAddBefore Especifica si se va a mover la capa delante o detrás de *layerToPutItBy*. Si especifica `false`, la capa se moverá detrás de *layerToPutItBy*. El valor predeterminado es `true`. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; mueve la primera capa especificada delante o detrás de la segunda capa especificada.

Ejemplo

El ejemplo siguiente mueve la capa en el índice 2 a la parte superior (sobre la capa en el índice 0):

```
fl.getDocumentDOM().getTimeline().reorderLayer(2, 0);
```

El ejemplo siguiente sitúa la capa en el índice 3 detrás de la capa en el índice 5:

```
fl.getDocumentDOM().getTimeline().reorderLayer(3, 5, false);
```

timeline.reverseFrames()

Disponibilidad

Flash MX 2004.

Uso

```
timeline.reverseFrames([startFrameIndex [, endFrameIndex]])
```

Parámetros

startFrameIndex Un índice basado en cero que especifica el primer fotograma en el que se iniciará la inversión de fotogramas. Si omite *startFrameIndex*, el método utilizará la selección actual. Este parámetro es opcional.

endFrameIndex Un índice basado en cero que especifica el primer fotograma en el que se dejará de invertir fotogramas; el intervalo de fotogramas llega hasta *endFrameIndex*, no incluido. Si sólo especifica *startFrameIndex*, *endFrameIndex* utilizará de forma predeterminada el valor de *startFrameIndex*. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; invierte un intervalo de fotogramas.

Ejemplo

El ejemplo siguiente invierte las posiciones de los fotogramas seleccionados actualmente:

```
fl.getDocumentDOM().getTimeline().reverseFrames();
```

El ejemplo siguiente invierte los fotogramas desde el Fotograma 10 hasta el 15, no incluido (recuerde que los valores de índice son distintos de los valores de número de fotograma):

```
fl.getDocumentDOM().getTimeline().reverseFrames(9, 14);
```

timeline.selectAllFrames()

Disponibilidad

Flash MX 2004.

Uso

```
timeline.selectAllFrames()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; selecciona todos los fotogramas de la línea de tiempo actual.

Ejemplo

El ejemplo siguiente selecciona todos los fotogramas de la línea de tiempo actual.

```
fl.getDocumentDOM().getTimeline().selectAllFrames();
```

timeline.setFrameProperty()

Disponibilidad

Flash MX 2004.

Uso

```
timeline.setFrameProperty(property, value [, startFrameIndex [,
  endFrameIndex]])
```

Parámetros

property Una cadena que especifica el nombre de la propiedad que se va a modificar. Para ver una lista completa de propiedades y valores, consulte "Resumen de propiedades del objeto Frame" en la página 279.

NOTA

No es posible utilizar este método para definir valores para propiedades de sólo lectura, como `frame.duration` y `frame.elements`.

value Especifica el valor con el que desea definir la propiedad. Para determinar los valores y el tipo adecuados, consulte "Resumen de propiedades del objeto Frame" en la página 279.

startFrameIndex Un índice basado en cero que especifica el número del fotograma inicial que desea modificar. Si omite *startFrameIndex*, el método utilizará la selección actual. Este parámetro es opcional.

endFrameIndex Un índice basado en cero que especifica el primer fotograma en el que se detendrá. El intervalo de fotogramas llega hasta *endFrameIndex*, no incluido. Si especifica *startFrameIndex* pero omite *endFrameIndex*, *endFrameIndex* utilizará de forma predeterminada el valor de *startFrameIndex*. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; establece la propiedad del objeto Frame para los fotogramas seleccionados.

Ejemplo

El ejemplo siguiente asigna el comando ActionScript `stop()` al primer fotograma de la capa superior del documento actual:

```
fl.getDocumentDOM().getTimeline().currentLayer = 0;
fl.getDocumentDOM().getTimeline().setSelectedFrames(0,0,true);
fl.getDocumentDOM().getTimeline().setFrameProperty("actionScript",
  "stop();");
```

El ejemplo siguiente establece una interpolación de movimiento desde el Fotograma 2 hasta el 5, no incluido (recuerde que los valores de índice son distintos de los valores de número de fotograma):

```
fl.getDocumentDOM().getTimeline().setFrameProperty("tweenType","motion",1,4
  );
```

timeline.setLayerProperty()

Disponibilidad

Flash MX 2004.

Uso

`timeline.setLayerProperty(`*`property`*`, `*`value`*` [, `*`layersToChange`*`])`

Parámetros

`property` Una cadena que especifica la propiedad que se va a definir. Para ver una lista de propiedades, consulte "Objeto Layer" en la página 311.

`value` El valor con el que desea definir la propiedad. Utilice el mismo tipo de valor que utilizaría para definir la propiedad en el objeto Layer.

`layersToChange` Una cadena que identifica qué capas deben modificarse. Los valores válidos son: `"selected"`, `"all"` y `"others"`. El valor predeterminado es `"selected"` si omite este parámetro. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; define con un valor determinado la propiedad especificada en todas las capas seleccionadas

Ejemplo

El ejemplo siguiente hace que la capa o capas seleccionadas sean invisibles:

```
fl.getDocumentDOM().getTimeline().setLayerProperty("visible", false);
```

El ejemplo siguiente define el nombre de la capa o capas seleccionadas como `"selLayer"`:

```
fl.getDocumentDOM().getTimeline().setLayerProperty("name", "selLayer");
```

timeline.setSelectedFrames()

Disponibilidad

Flash MX 2004.

Uso

```
timeline.setSelectedFrames(startFrameIndex, endFrameIndex [,
  bReplaceCurrentSelection])
timeline.setSelectedFrames(selectionList [, bReplaceCurrentSelection])
```

Parámetros

`startFrameIndex` Un índice basado en cero que especifica el fotograma inicial que desea definir.

`endFrameIndex` Un índice basado en cero que especifica el final de la selección; *`endFrameIndex`* es el fotograma que va detrás del último fotograma del intervalo que se va a seleccionar.

`bReplaceCurrentSelection` Un valor booleano que, si se define como `true`, hace que se anule la selección los fotogramas seleccionados actualmente antes de seleccionar los fotogramas especificados. El valor predeterminado es `true`.

`selectionList` Una matriz de tres enteros que devuelve `timeline.getSelectedFrames()`.

Valor devuelto

Ninguno.

Descripción

Método; selecciona un intervalo de fotogramas de la capa actual o define los fotogramas seleccionadas con la matriz de selección transferida a este método.

Ejemplo

El ejemplo siguiente selecciona la capa superior, desde el Fotograma 1 hasta el Fotograma 10 (no incluido), y a continuación, añade a la selección actual desde el Fotograma 12 hasta el Fotograma 15 (no incluido) de la misma capa (recuerde que los valores de índice son distintos de los valores de número de fotograma):

```
fl.getDocumentDOM().getTimeline().setSelectedFrames(0, 9);
fl.getDocumentDOM().getTimeline().setSelectedFrames(11, 14, false);
```

El ejemplo siguiente almacena en primer lugar la matriz de fotogramas seleccionados en la variable `savedSelectionList` y, a continuación, utiliza posteriormente la matriz en el código para volver a seleccionar esos fotogramas cuando un comando o la interacción del usuario ha cambiado la selección:

```
var savedSelectionList =
  fl.getDocumentDOM().getTimeline().getSelectedFrames();
// Hace algo que cambia la selección.
fl.getDocumentDOM().getTimeline().setSelectedFrames(savedSelectionList);
```

El ejemplo siguiente selecciona la capa superior, desde el Fotograma 1 hasta el Fotograma 10 (no incluido) y, a continuación, añade a la selección actual desde el Fotograma 12 hasta el Fotograma 15 (no incluido) de la misma capa:

```
fl.getDocumentDOM().getTimeline().setSelectedFrames([0, 0, 9]);
fl.getDocumentDOM().getTimeline().setSelectedFrames([0, 11, 14], false);
```

Véase también

`timeline.getSelectedFrames()`

timeline.setSelectedLayers()

Disponibilidad

Flash MX 2004.

Uso

```
timeline.setSelectedLayers(index [, bReplaceCurrentSelection])
```

Parámetros

`index` Un índice basado en cero para la capa que desea seleccionar.

`bReplaceCurrentSelection` Un valor booleano que, si se define como `true`, hace que el método reemplace la selección actual; `false` hace que el método amplíe la selección actual. El valor predeterminado es `true`. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; define la capa que se va a seleccionar y, además, convierte la capa especificada en la capa actual. Al seleccionar una capa se seleccionan todos los fotogramas de la capa.

Ejemplo

El ejemplo siguiente selecciona la capa superior:

```
fl.getDocumentDOM().getTimeline().setSelectedLayers(0);
```

El ejemplo siguiente añade la capa siguiente a la selección:

```
fl.getDocumentDOM().getTimeline().setSelectedLayers(1, false);
```

Véase también

`timeline.getSelectedLayers()`

timeline.showLayerMasking()

Disponibilidad

Flash MX 2004.

Uso

```
timeline.showLayerMasking([layer])
```

Parámetros

`layer` Un índice basado en cero de una máscara o capa enmascarada para mostrar enmascaramiento durante la edición. Este parámetro es opcional.

Valor devuelto

Ninguno.

Descripción

Método; muestra el enmascaramiento de capas durante la edición bloqueando la máscara y las capas enmascaradas. Este método utiliza la capa actual si no hay ninguna capa especificada. Si utiliza este método en una capa que no es de tipo Máscara o Enmascarada, Flash muestra un error en el panel Salida.

Ejemplo

El ejemplo siguiente especifica que el enmascaramiento de capa de la primera capa debe mostrarse durante la edición.

```
fl.getDocumentDOM().getTimeline().showLayerMasking(0);
```

Objeto ToolObj

Disponibilidad

Flash MX 2004.

Descripción

Un objeto ToolObj representa una herramienta individual en el panel Herramientas. Para acceder a un objeto ToolObj, utilice las propiedades del Objeto Tools: la matriz de `tools.toolObjs` o `tools.activeTool`.

Resumen de métodos del objeto ToolObj

Los métodos siguientes están disponibles para el objeto ToolObj.

NOTA	Los métodos siguientes sólo se utilizan para crear herramientas ampliables.

Método	Descripción
`toolObj.enablePIControl()`	Activa o desactiva el control especificado en un inspector de propiedades. Sólo se utiliza para crear herramientas ampliables.
`toolObj.setIcon()`	Identifica un archivo PNG para utilizarlo como icono de herramienta en el panel Herramientas de Flash.
`toolObj.setMenuString()`	Define la cadena que aparece en el menú emergente como nombre de la herramienta.
`toolObj.setOptionsFile()`	Asocia un archivo XML con la herramienta.
`toolObj.setPI()`	Define un determinado inspector de propiedades para utilizarlo cuando se active la herramienta.
`toolObj.setToolName()`	Asigna un nombre a la herramienta para la configuración del panel Herramientas.
`toolObj.setToolTip()`	Define la sugerencia que aparece cuando el ratón se mantiene sobre el icono de herramienta.
`toolObj.showPIControl()`	Muestra u oculta un control en el inspector de propiedades.
`toolObj.showTransformHandles()`	Se llama en el método `configureTool()` de un archivo JavaScript de una herramienta ampliable para indicar que los controladores de transformación libre deben aparecer cuando la herramienta está activa.

Resumen de propiedades del objeto ToolObj

La propiedad siguiente está disponible para el objeto ToolObj:

Propiedad	Descripción
`toolObj.depth`	Un entero que especifica la profundidad de la herramienta en el menú emergente del panel Herramientas.
`toolObj.iconID`	Un entero que especifica el ID de recurso de la herramienta.
`toolObj.position`	De sólo lectura; un entero que especifica la posición de la herramienta en el panel Herramientas.

toolObj.depth

Disponibilidad

Flash MX 2004.

Uso

```
toolObj.depth
```

Descripción

Propiedad de sólo lectura; un entero que especifica la profundidad de la herramienta en el menú emergente del panel Herramientas. Esta propiedad sólo se utiliza para crear herramientas ampliables.

Ejemplo

El siguiente ejemplo especifica que la herramienta tiene una profundidad de 1, lo que indica un nivel por debajo de una herramienta del panel Herramientas.

```
fl.tools.activeTool.depth = 1;
```

toolObj.enablePIControl()

Disponibilidad

Flash MX 2004.

Uso

```
toolObj.enablePIControl( control, bEnable )
```

Parámetros

control Una cadena que especifica el nombre del control que se va a activar o desactivar. Los valores válidos dependen del inspector de propiedades que invoque esta herramienta (véase `toolObj.setPI()`).

Un inspector de propiedades de forma cuenta con los controles siguientes:

stroke	fill

Un inspector de propiedades de texto cuenta con los controles siguientes:

type	font	pointsize
color	bold	italic
direction	alignLeft	alignCenter
alignRight	alignJustify	spacing
position	autoKern	small
rotation	format	lineType
selectable	html	border
deviceFonts	varEdit	options
link	maxChars	target

Un inspector de propiedades de película cuenta con los controles siguientes:

size	publish	background
framerate	player	profile

bEnable Un valor booleano que determina si se activa (`true`) o desactiva (`false`) el control.

Valor devuelto

Ninguno.

Descripción

Método; activa o desactiva el control especificado en un inspector de propiedades. Sólo se utiliza para crear herramientas ampliables.

Ejemplo

El comando siguiente en un archivo JavaScript de una herramienta ampliable configurará Flash para que no muestre las opciones de trazo en el inspector de propiedades de esa herramienta:

```
theTool.enablePIControl( "stroke",  false);
```

toolObj.iconID

Disponibilidad

Flash MX 2004.

Uso

```
toolObj.iconID
```

Descripción

Propiedad de sólo lectura; un entero con un valor de -1. Esta propiedad sólo se utiliza cuando se crean herramientas ampliables. Un valor de `iconID` de -1 indica que Flash no intentará buscar un icono para la herramienta. En su lugar, el script de la herramienta deberá especificar el icono que se mostrará en el panel Herramientas; véase `toolObj.setIcon()`.

Ejemplo

El siguiente ejemplo asigna un valor de -1 (el ID de icono de la herramienta actual) a la variable `toolIconID`:

```
var toolIconID = fl.tools.activeTool.iconID
```

toolObj.position

Disponibilidad

Flash MX 2004.

Uso

```
toolObj.position
```

Descripción

Propiedad de sólo lectura; un entero que especifica la posición de la herramienta en el panel Herramientas. Esta propiedad sólo se utiliza cuando se crean herramientas ampliables.

Ejemplo

Los comandos siguientes del método `mouseDown()` de un archivo JavaScript de una herramienta mostrarán la posición de esa herramienta en el panel Herramientas como un entero en el panel Salida:

```
myToolPos = fl.tools.activeTool.position;
fl.trace(myToolPos);
```

toolObj.setIcon()

Disponibilidad

Flash MX 2004.

Uso

```
toolObj.setIcon( file )
```

Parámetros

file Una cadena que especifica el nombre del archivo PNG que se va a utilizar como icono. El archivo PNG se debe colocar en la misma carpeta que el archivo JSFL.

Valor devuelto

Ninguno.

Descripción

Método; identifica un archivo PNG para utilizarlo como icono de herramienta en el panel Herramientas. Este método sólo se utiliza cuando se crean herramientas ampliables.

Ejemplo

El ejemplo siguiente especifica que la imagen del archivo PolyStar.png debe utilizarse como icono para la herramienta llamada `PolyStar`. Este código se toma del archivo de muestra PolyStar.jsfl (véase "Herramienta de muestra PolyStar" en la página 21):

```
theTool = fl.tools.activeTool;
theTool.setIcon("PolyStar.png");
```

toolObj.setMenuString()

Disponibilidad

Flash MX 2004.

Uso

```
toolObj.setMenuString( menuStr )
```

Parámetros

menuStr Una cadena que especifica el nombre que aparece en el menú emergente como nombre de la herramienta.

Valor devuelto

Ninguno.

Descripción

Método; define la cadena que aparece en el menú emergente como nombre de la herramienta. Este método sólo se utiliza cuando se crean herramientas ampliables.

Ejemplo

El ejemplo siguiente especifica que la herramienta llamada `theTool` debe mostrar el nombre "PolyStar Tool" en su menú emergente. Este código se toma del archivo de muestra PolyStar.jsfl (véase "Herramienta de muestra PolyStar" en la página 21):

```
theTool = fl.tools.activeTool;
theTool.setMenuString("PolyStar Tool");
```

toolObj.setOptionsFile()

Disponibilidad

Flash MX 2004.

Uso

```
toolObj.setOptionsFile( xmlFile )
```

Parámetros

`xmlFile` Una cadena que especifica el nombre del archivo XML que tiene la descripción de las opciones de la herramienta. El archivo XML se debe colocar en la misma carpeta que el archivo JSFL.

Valor devuelto

Ninguno.

Descripción

Método; asocia un archivo XML con la herramienta. El archivo especifica las opciones que van a aparecer en un panel modal que invoca el botón Opciones del inspector de propiedades. Normalmente este método se utilizaría en la función `configureTool()` dentro del archivo JSFL. Véase `configureTool()`.

Por ejemplo, el archivo PolyStar.xml especifica tres opciones asociadas a la herramienta Polígono:

```
<properties>
  <property name="Style"
    variable="style"
    list="polygon,star"
    defaultValue="0"
    type="Strings"  />

  <property name="Number of Sides"
    variable="nsides"
    min="3"
    max="32"
    defaultValue="5"
    type="Number"   />

  <property name="Star point size"
    variable="pointParam"
    min="0"
    max="1"
    defaultValue=".5"
    type="Double"   />

</properties>
```

Ejemplo

El ejemplo siguiente especifica que el archivo llamado PolyStar.xml está asociado con la herramienta que se encuentra activa. Este código se toma del archivo de muestra PolyStar.jsfl (véase "Herramienta de muestra PolyStar" en la página 21):

```
theTool = fl.tools.activeTool;
theTool.setOptionsFile( "PolyStar.xml" );
```

toolObj.setPI()

Disponibilidad

Flash MX 2004.

Uso

```
toolObj.setPI( pi )
```

Parámetros

pi Una cadena que especifica el inspector de propiedades que se va a invocar para esta herramienta.

Valor devuelto

Ninguno.

Descripción

Método; especifica qué inspector de propiedades debe utilizarse cuando se active la herramienta. Este método sólo se utiliza cuando se crean herramientas ampliables. Los valores aceptables son: `"shape"` (predeterminado), `"text"` y `"movie"`.

Ejemplo

El ejemplo siguiente especifica que debe utilizarse el inspector de propiedades de forma cuando se active la herramienta. Este código se toma del archivo de muestra PolyStar.jsfl (véase "Herramienta de muestra PolyStar" en la página 21):

```
theTool = fl.tools.activeTool;
theTool.setPI( "shape" );
```

toolObj.setToolName()

Disponibilidad

Flash MX 2004.

Uso

```
toolObj.setToolName( name )
```

Parámetros

name Cadena que especifica el nombre de la herramienta.

Valor devuelto

Ninguno.

Descripción

Método; asigna un nombre a la herramienta para la configuración del panel Herramientas. Este método sólo se utiliza cuando se crean herramientas ampliables. El nombre sólo lo utiliza el archivo de diseño XML que lee Flash para crear el panel Herramientas. El nombre no aparece en la interfaz de usuario de Flash.

Ejemplo

El ejemplo siguiente asigna el nombre "polystar" a la herramienta llamada `theTool`. Este código se toma del archivo de muestra PolyStar.jsfl (véase "Herramienta de muestra PolyStar" en la página 21):

```
theTool = fl.tools.activeTool;
theTool.setToolName("polystar");
```

toolObj.setToolTip()

Disponibilidad

Flash MX 2004.

Uso

```
toolObj.setToolTip( toolTip )
```

Parámetros

toolTip Una cadena que especifica la sugerencia que se va a utilizar para la herramienta.

Valor devuelto

Ninguno.

Descripción

Método; define la sugerencia que aparece cuando el ratón se mantiene sobre el icono de herramienta. Este método sólo se utiliza cuando se crean herramientas ampliables.

Ejemplo

El ejemplo siguiente especifica que la sugerencia de la herramienta debe ser "PolyStar Tool". Este código se toma del archivo de muestra PolyStar.jsfl (véase "Herramienta de muestra PolyStar" en la página 21):

```
theTool = fl.tools.activeTool;
theTool.setToolTip("PolyStar Tool");
```

toolObj.showPIControl()

Disponibilidad

Flash MX 2004.

Uso

```
toolObj.showPIControl( control, bShow )
```

Parámetros

control Una cadena que especifica el nombre del control que se va a mostrar u ocultar. Este método sólo se utiliza cuando se crean herramientas ampliables. Los valores válidos dependen del inspector de propiedades que invoque esta herramienta (véase `toolObj.setPI()`).

Un inspector de propiedades de forma cuenta con los controles siguientes:

stroke	fill

Un inspector de propiedades de texto cuenta con los controles siguientes:

type	font	pointsize
color	bold	italic
direction	alignLeft	alignCenter
alignRight	alignJustify	spacing
position	autoKern	small
rotation	format	lineType
selectable	html	border
deviceFonts	varEdit	options
link	maxChars	target

El inspector de propiedades de película cuenta con los controles siguientes:

size	publish	background
framerate	player	profile

bShow Un valor booleano que determina si se muestra u oculta el control especificado (`true` muestra el control; `false` oculta el control).

Valor devuelto

Ninguno.

Descripción

Método; muestra u oculta un control en el inspector de propiedades. Este método sólo se utiliza cuando se crean herramientas ampliables.

Ejemplo

El comando siguiente en un archivo JavaScript de una herramienta ampliable configurará Flash para que no muestre las opciones de relleno en el inspector de propiedades de esa herramienta:

```
fl.tools.activeTool.showPIControl( "fill", false );
```

toolObj.showTransformHandles()

Disponibilidad

Flash MX 2004.

Uso

```
toolObj.showTransformHandles( bShow )
```

Parámetros

bShow Un valor booleano que determina si se muestran u ocultan los controladores de transformación libre para la herramienta actual (`true` muestra lo controladores; `false` los oculta).

Valor devuelto

Ninguno.

Descripción

Método; se llama en el método `configureTool()` de un archivo JavaScript de una herramienta ampliable para indicar que los controladores de transformación libre deben aparecer cuando la herramienta está activa. Este método sólo se utiliza cuando se crean herramientas ampliables.

Ejemplo

Véase `configureTool()`.

Objeto Tools

Disponibilidad

Flash MX 2004.

Descripción

Se puede acceder al objeto Tools desde el objeto Flash (`fl.tools`). La propiedad `tools.toolObjs` contiene una matriz de objetos ToolObj y la propiedad `tools.activeTool` devuelve el objeto ToolObj para la herramienta activa actualmente. (Véase también "Objeto ToolObj" en la página 508 y "Herramientas ampliables" en la página 25.)

NOTA	Los métodos y las propiedades siguientes sólo se utilizan para crear herramientas extensibles.

Resumen de métodos del objeto Tools

Los métodos siguientes están disponibles para el objeto Tools.

Método	Descripción
`tools.constrainPoint()`	Toma dos puntos y devuelve un nuevo punto ajustado o *restringido*.
`tools.getKeyDown()`	Devuelve la tecla presionada más recientemente.
`tools.setCursor()`	Define el puntero con una apariencia especificada.
`tools.snapPoint()`	Toma un punto como entrada y devuelve un punto nuevo que se puede encajar o *ajustar* al objeto geométrico más próximo.

Resumen de propiedades del objeto Tools

Las propiedades siguientes están disponibles para el objeto Tools.

Propiedad	Descripción
`tools.activeTool`	De sólo lectura; devuelve el Objeto ToolObj para la herramienta que se encuentra activa.
`tools.altIsDown`	De sólo lectura; un valor booleano que identifica si se está presionando la tecla Alt.
`tools.ctlIsDown`	De sólo lectura; un valor booleano que identifica si se está presionando la tecla Control.

Propiedad	Descripción
tools.mouseIsDown	De sólo lectura; un valor booleano que identifica si se está presionando el botón izquierdo del ratón.
tools.penDownLoc	De sólo lectura; un punto que representa la posición del último evento de pulsación del ratón en el escenario.
tools.penLoc	De sólo lectura; un punto que representa la ubicación actual del ratón.
tools.shiftIsDown	De sólo lectura; un valor booleano que identifica si se está presionando la tecla Mayús.
tools.toolObjs	De sólo lectura; una matriz de objetos ToolObj.

tools.activeTool

Disponibilidad

Flash MX 2004.

Uso

```
tools.activeTool
```

Descripción

Propiedad de sólo lectura; devuelve el Objeto ToolObj para la herramienta que se encuentra activa.

Ejemplo

El ejemplo siguiente guarda un objeto que representa la herramienta que se encuentra activa en la variable `theTool`.

```
var theTool = fl.tools.activeTool;
```

tools.altIsDown

Disponibilidad

Flash MX 2004.

Uso

```
tools.altIsDown
```

Descripción

Propiedad de sólo lectura; un valor booleano que identifica si se está presionando la tecla Alt. El valor es `true` si la tecla Alt está presionada, y `false` en caso contrario.

Ejemplo

El siguiente ejemplo determina si se está presionando la tecla Alt.

```
var isAltDown = fl.tools.altIsDown;
```

tools.constrainPoint()

Disponibilidad

Flash MX 2004.

Uso

```
tools.constrainPoint(pt1, pt2)
```

Parámetros

pt1 y *pt2* especifican el punto de inicio del clic y el punto hasta donde se arrastra.

Valor devuelto

Un nuevo punto ajustado o restringido.

Descripción

Método; toma dos puntos y devuelve un nuevo punto ajustado o restringido. Si está presionada la tecla Mayús cuando se ejecuta el comando, el punto devuelto queda restringido a seguir una limitación de 45º (resulta útil para elementos como una línea con flecha) o para restringir un objeto de modo que mantenga su relación de aspecto (como extraer un cuadrado perfecto con la herramienta Rectángulo).

Ejemplo

El ejemplo siguiente devuelve un punto restringido:

```
pt2 = fl.tools.constrainPoint(pt1, tempPt);
```

tools.ctlIsDown

Disponibilidad

Flash MX 2004.

Uso

```
tools.ctlIsDown
```

Descripción

Propiedad de sólo lectura; un valor booleano que es `true` si se presiona la tecla Control y `false` en caso contrario.

Ejemplo

El siguiente ejemplo determina si se está presionando la tecla Control.

```
var isCtrldown = fl.tools.ctrlIsDown;
```

tools.getKeyDown()

Disponibilidad

Flash MX 2004.

Uso

```
tools.getKeyDown()
```

Parámetros

Ninguno.

Valor devuelto

El valor entero de la tecla.

Descripción

Método; devuelve la tecla presionada más recientemente.

Ejemplo

El ejemplo siguiente muestra el valor de entero de la tecla presionada más recientemente en el panel Salida:

```
var theKey = fl.tools.getKeyDown();
fl.trace(theKey);
```

tools.mouseIsDown

Disponibilidad

Flash MX 2004.

Uso

```
tools.mouseIsDown
```

Descripción

Propiedad de sólo lectura; un valor booleano que es `true` si se está presionando el botón izquierdo del ratón y `false` en caso contrario.

Ejemplo

El siguiente ejemplo determina si se presiona el botón izquierdo del ratón.

```
var isMouseDown = fl.tools.mouseIsDown;
```

tools.penDownLoc

Disponibilidad

Flash MX 2004.

Uso

```
tools.penDownLoc
```

Descripción

Propiedad de sólo lectura; un punto que representa la posición del último evento de pulsación del ratón en el escenario. La propiedad `tools.penDownLoc` incluye dos propiedades, *x* e *y*, correspondientes a la posición *x*, *y* del puntero del ratón.

Ejemplo

El ejemplo siguiente determina la posición del último evento de pulsación del ratón en el escenario y muestra los valores *x* e *y* en el panel Salida.

```
var pt1 = fl.tools.penDownLoc;
fl.trace("x,y location of last mouseDown event was " + pt1.x + ", " + pt1.y)
```

Véase también

`tools.penLoc`

tools.penLoc

Disponibilidad

Flash MX 2004.

Uso

```
tools.penLoc
```

Descripción

Propiedad de sólo lectura; un punto que representa la ubicación actual del puntero del ratón. La propiedad `tools.penLoc` incluye dos propiedades, *x* e *y*, correspondientes a la posición *x*, *y* del puntero del ratón.

Ejemplo

El ejemplo siguiente determina la posición actual del ratón.

```
var tempPt = fl.tools.penLoc;
```

Véase también

tools.penDownLoc

tools.setCursor()

Disponibilidad

Flash MX 2004.

Uso

```
tools.setCursor( cursor )
```

Parámetros

cursor Un entero que define la apariencia del puntero, tal como se describe en la lista siguiente:

- 0 Cursor de signo más (+)
- 1 Flecha negra
- 2 Flecha blanca
- 3 Flecha de cuatro puntas
- 4 Flecha horizontal de dos puntas
- 5 Flecha vertical de dos puntas
- 6 X
- 7 Cursor de mano

Valor devuelto

Ninguno.

Descripción

Método; define el puntero con una apariencia especificada.

Ejemplo

El ejemplo siguiente define el puntero como una flecha negra.

```
fl.tools.setCursor(1);
```

tools.shiftIsDown

Disponibilidad

Flash MX 2004.

Uso

```
tools.shiftIsDown
```

Descripción

Propiedad de sólo lectura; un valor booleano que es `true` si se presiona la tecla Mayús y `false` en caso contrario.

Ejemplo

El siguiente ejemplo determina si se está presionando la tecla Mayús.

```
var isShiftDown = fl.tools.shiftIsDown;
```

tools.snapPoint()

Disponibilidad

Flash MX 2004.

Uso

```
tools.snapPoint(pt)
```

Parámetros

pt Especifica la ubicación del punto para el que desea devolver un punto de ajuste.

Valor devuelto

Un nuevo punto que se puede encajar o ajustar al objeto geométrico más próximo.

Descripción

Método; toma un punto como entrada y devuelve un punto nuevo que se puede encajar o *ajustar* al objeto geométrico más próximo. Si el ajuste está desactivado en el menú Ver de la interfaz de usuario de Flash, el punto devuelto es el punto original.

Ejemplo

El ejemplo siguiente devuelve un nuevo punto que se puede ajustar al objeto geométrico más próximo.

```
var theSnapPoint = fl.tools.snapPoint(pt1);
```

tools.toolObjs

Disponibilidad

Flash MX 2004.

Uso

```
tools.toolObjs
```

Descripción

Propiedad de sólo lectura; una matriz de objetos ToolObj (véase Objeto ToolObj).

Objeto Vertex

Disponibilidad

Flash MX 2004.

Descripción

El objeto Vertex forma parte de la estructura de datos de formas que contiene los datos de coordenadas.

Resumen de métodos del objeto Vertex

Puede emplear los métodos siguientes con el objeto Vertex.

Método	Descripción
`vertex.getHalfEdge()`	Obtiene un Objeto HalfEdge que comparte este vértice.
`vertex.setLocation()`	Define la ubicación del vértice.

Resumen de propiedades del objeto Vertex

Las propiedades siguientes están disponibles para el objeto Vertex:

Propiedad	Descripción
`vertex.x`	De sólo lectura; la ubicación *x* del vértice en píxeles.
`vertex.y`	De sólo lectura; la ubicación *y* del vértice en píxeles.

vertex.getHalfEdge()

Disponibilidad

Flash MX 2004.

Uso

```
vertex.getHalfEdge()
```

Parámetros

Ninguno.

Valor devuelto

Un Objeto HalfEdge.

Descripción

Método; obtiene un Objeto HalfEdge que comparte este vértice.

Ejemplo

El ejemplo siguiente muestra cómo obtener otros lados dirigidos que compartan el mismo vértice.

```
var shape = fl.getDocumentDOM().selection[0];
var hEdge = shape.edges[0].getHalfEdge(0);
var theVertex = hEdge.getVertex();
var someHEdge = theVertex.getHalfEdge(); // No necesariamente el mismo lado
  dirigido
var theSameVertex = someHEdge.getVertex();
fl.trace('the same vertex: ' + theSameVertex);
```

vertex.setLocation()

Disponibilidad

Flash MX 2004.

Uso

```
vertex.setLocation( x, y )
```

Parámetros

x Un valor de coma flotante que especifica la coordenada *x* de dónde deberá situarse el vértice, en píxeles.

y Un valor de coma flotante que especifica la coordenada *y* de dónde deberá situarse el vértice, en píxeles.

Valor devuelto

Ninguno.

Descripción

Método; define la ubicación del vértice. Deberá llamar a `shape.beginEdit()` antes de utilizar este método.

Ejemplo

El ejemplo siguiente define el vértice en el punto de origen.

```
var shape = fl.getDocumentDOM().selection[0];
var hEdge = shape.edges[0].getHalfEdge(0);
var vertex = hEdge.getVertex();

// Mueve el vértice al origen.
vertex.setLocation(0.0, 0.0);
```

vertex.x

Disponibilidad

Flash MX 2004.

Uso

```
vertex.x
```

Descripción

Propiedad de sólo lectura; la ubicación *x* del vértice en píxeles.

Ejemplo

El ejemplo siguiente muestra la ubicación de los valores *x* e *y* del vértice en el panel Salida.

```
var shape = fl.getDocumentDOM().selection[0];
var hEdge = shape.edges[0].getHalfEdge(0);
var vertex = hEdge.getVertex();

fl.trace('x location of vertex is: ' +  vertex.x);
fl.trace('y location of vertex is: ' + vertex.y);
```

vertex.y

Disponibilidad

Flash MX 2004.

Uso

```
vertex.y
```

Descripción

Propiedad de sólo lectura; la ubicación *y* del vértice en píxeles.

Ejemplo

Véase `vertex.x`.

Objeto XMLUI

Disponibilidad

Flash MX 2004.

Descripción

Flash 8 admite cuadros de diálogo personalizados escritos en un subconjunto del lenguaje de interfaz de usuario XML (XUL). Hay varias funciones de Flash que pueden utilizar un cuadro de diálogo de interfaz de usuario XML (XMLUI) como, por ejemplo, comandos y comportamientos, para suministrar una interfaz de usuario para funciones que se crean utilizando la extensibilidad. El objeto XMLUI permite obtener y definir propiedades de un cuadro de diálogo XMLUI, así como aceptar o cancelar una. Los métodos de XMLUI pueden utilizarse en devoluciones de llamada, por ejemplo, controladores `oncommand` en los botones.

Puede escribir un archivo dialog.xml e invocarlo desde la API de JavaScript empleando el método `document.xmlPanel()`. Para recuperar un objeto que representa el cuadro de diálogo XMLUI actual, utilice `fl.xmlui`.

Para más información, consulte Apéndice B, "XML a interfaz de usuario" en *Utilización de Flash*.

Resumen de métodos del objeto XMLUI

Los métodos siguientes están disponibles para el objeto XMLUI:

Método	Descripción
`xmlui.accept()`	Cierra el cuadro de diálogo XMLUI actual con un estado "accept".
`xmlui.cancel()`	Cierra el cuadro de diálogo XMLUI actual con un estado "cancel".
`xmlui.get()`	Recupera el valor de la propiedad especificada del cuadro de diálogo XMLUI actual.
`xmlui.getControlItemElement()`	Devuelve el elemento de control actual para el control especificado.
`xmlui.getEnabled()`	Devuelve un valor booleano que especifica si el control está activo o inactivo (atenuado).
`xmlui.getVisible()`	Devuelve un valor booleano que especifica si el control está visible u oculto.

Método	Descripción
`xmlui.set()`	Modifica el valor de la propiedad especificada del cuadro de diálogo XMLUI actual.
`xmlui.setControlItemElement()`	Establece la etiqueta y el valor del elemento actual.
`xmlui.setControlItemElements()`	Establece los pares de etiqueta y valor del elemento actual.
`xmlui.setEnabled()`	Activa o desactiva (atenúa) un control.
`xmlui.setVisible()`	Muestra u oculta un control.

xmlui.accept()

Disponibilidad

Flash MX 2004.

Uso

```
xmlui.accept()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; cierra el cuadro de diálogo XMLUI actual con un estado de aceptar, lo que equivale a que el usuario haga clic en el botón Aceptar.

Véase también

`fl.xmlui`, `document.xmlPanel()`, `xmlui.cancel()`

xmlui.cancel()

Disponibilidad

Flash MX 2004.

Uso

```
xmlui.cancel()
```

Parámetros

Ninguno.

Valor devuelto

Ninguno.

Descripción

Método; cierra el cuadro de diálogo XMLUI actual con un estado de cancelar, lo que equivale a que el usuario haga clic en el botón Cancelar.

Véase también

`fl.xmlui`, `document.xmlPanel()`, `xmlui.accept()`

xmlui.get()

Disponibilidad

Flash MX 2004.

Uso

```
xmlui.get( controlPropertyName )
```

Parámetros

`controlPropertyName` Una cadena que especifica el nombre de la propiedad XMLUI cuyo valor desea recuperar.

Valor devuelto

Una cadena que representa el valor de la propiedad especificada. En los casos en que se podría esperar un valor booleano de `true` o `false`, devuelve la cadena `"true"` o `"false"`.

Descripción

Método; recupera el valor de la propiedad especificada del cuadro de diálogo XMLUI actual.

Ejemplo

El siguiente ejemplo devuelve el valor de una propiedad denominada “URL”:

```
fl.xmlui.get("URL");
```

Véase también

`fl.xmlui`, `document.xmlPanel()`, `xmlui.getControlItemElement()`, `xmlui.set()`

xmlui.getControlItemElement()

Disponibilidad

Flash 8.

Uso

```
xmlui.getControlItemElement( controlPropertyName )
```

Parámetros

controlPropertyName Una cadena que especifica la propiedad cuyo elemento de control desea recuperar.

Valor devuelto

Un objeto que representa el elemento de control actual para el control especificado por *controlPropertyName*.

Descripción

Método; devuelve la etiqueta y el valor de la línea seleccionada en un control ListBox o ComboBox para el control especificado por *controlPropertyName*.

Ejemplo

El siguiente ejemplo devuelve la etiqueta y el valor de la línea seleccionada actualmente para el control myListBox:

```
var elem = new Object();
elem = fl.xmlui.getControlItemElement("myListBox");
fl.trace("label = " + elem.label + " value = " + elem.value);
```

Véase también

fl.xmlui, document.xmlPanel(), xmlui.get(), xmlui.setControlItemElement(), xmlui.setControlItemElements()

xmlui.getEnabled()

Disponibilidad

Flash 8.

Uso

```
xmlui.getEnabled( controlID )
```

Parámetros

controlID Una cadena que especifica el atributo de identificación del control cuyo estado desea recuperar.

Valor devuelto

Un valor booleano de `true` si el control es correcto, y de `false` en caso contrario.

Descripción

Método; devuelve un valor booleano que especifica si el control está activo o inactivo (atenuado).

Ejemplo

El ejemplo siguiente devuelve un valor que indica si está activado el control con el atributo de identificación `myListBox`:

```
var isEnabled = fl.xmlui.getEnabled("myListBox");
fl.trace(isEnabled);
```

Véase también

`fl.xmlui`, `document.xmlPanel()`, `xmlui.setEnabled()`

xmlui.getVisible()

Disponibilidad

Flash 8.

Uso

`xmlui.getVisible(`*controlID*`)`

Parámetros

controlID Una cadena que especifica el atributo de identificación del control cuyo estado de visibilidad desea recuperar.

Valor devuelto

Un valor booleano de `true` si el control está visible o de `false` si no se ve (está oculto).

Descripción

Método; devuelve un valor booleano que especifica si el control está visible u oculto.

Ejemplo

El ejemplo siguiente devuelve un valor que indica si está visible el control con el atributo de identificación `myListBox`:

```
var isVisible = fl.xmlui.getVisible("myListBox");
fl.trace(isVisible);
```

Véase también

`xmlui.setVisible()`

xmlui.set()

Disponibilidad

Flash MX 2004.

Uso

`xmlui.set( controlPropertyName, value )`

Parámetros

`controlPropertyName` Una cadena que especifica el nombre de la propiedad XMLUI que se va a modificar.

`value` Una cadena que especifica el valor para el que desea definir la propiedad XMLUI.

Valor devuelto

Ninguno.

Descripción

Método; modifica el valor de la propiedad especificada del cuadro de diálogo XMLUI actual.

Ejemplo

El ejemplo siguiente define como "www.macromedia.com" el valor de una propiedad denominada "URL":

```
fl.xmlui.set("URL", "www.macromedia.com");
```

Véase también

`fl.xmlui`, `document.xmlPanel()`, `xmlui.get()`, `xmlui.setControlItemElement()`, `xmlui.setControlItemElements()`

xmlui.setControlItemElement()

Disponibilidad

Flash 8.

Uso

`xmlui.setControlItemElement( controlPropertyName, elementItem )`

Parámetros

`controlPropertyName` Una cadena que especifica el elemento de control que se va a definir.

`elementItem` Un objeto JavaScript con una propiedad de cadena llamada `label` y una cadena opcional llamada `value`. Si no existe la propiedad `value`, se creará y se le asignará el mismo valor que `label`.

Valor devuelto

Ninguno.

Descripción

Método; define la etiqueta y el valor de la línea seleccionada actualmente en el control ListBox o ComboBox especificado por *`controlPropertyName`*.

Ejemplo

El siguiente ejemplo define la etiqueta y el valor para el elemento actual de la propiedad de control denominada `"PhoneNumber"`:

```
var elem = new Object();
elem.label = "Fax";
elem.value = "707-555-5555";
fl.xmlui.setControlItemElement("PhoneNumber",elem);
```

Véase también

`fl.xmlui`, `document.xmlPanel()`, `xmlui.getControlItemElement()`, `xmlui.set()`, `xmlui.setControlItemElements()`

xmlui.setControlItemElements()

Disponibilidad

Flash 8.

Uso

`xmlui.setControlItemElements( controlID, elementItemArray )`

Parámetros

`controlID` Una cadena que especifica el atributo de identificación del control que desea definir.

`elementItemArray` Una matriz de objetos JavaScript, donde cada objeto tiene una propiedad de cadena llamada `label` y una propiedad de cadena opcional llamada `value`. Si no existe la propiedad `value`, se creará y se le asignará el mismo valor que `label`.

Valor devuelto

Ninguno.

Descripción

Método; borra los valores del control ListBox o ComboBox especificado por *`controlID`* y reemplaza la lista o elementos de menú con los pares `label`, `value` especificados por *`elementItemArray`*.

Ejemplo

El ejemplo siguiente define la etiqueta y el valor de los elementos del control con el atributo de identificación `myControlID` con los pares `label`, `value` especificados:

```
var nameArray = new Array("January", "February", "March");
var monthArray = new Array();
for (i=0;i<nameArray.length;i++){
  elem = new Object();
  elem.label = nameArray[i];
  elem.value = i;
  monthArray[i] = elem;
}
fl.xmlui.setControlItemElements("myControlID", monthArray);
```

Véase también

`xmlui.getControlItemElement()`, `xmlui.set()`, `xmlui.setControlItemElement()`

xmlui.setEnabled()

Disponibilidad

Flash 8.

Uso

```
xmlui.setEnabled( controlID, enable )
```

Parámetros

controlID Una cadena que especifica el atributo de identificación del control que desea activar o desactivar.

enable Un valor booleano de `true` si desea activar el control o de `false` si desea desactivarlo (atenuarlo).

Valor devuelto

Ninguno.

Descripción

Método; activa o desactiva (atenúa) un control.

Ejemplo

El siguiente ejemplo atenúa el control con el atributo de identificación `myControl`:

```
fl.xmlui.setEnabled("myControl", false);
```

Véase también

`xmlui.getEnabled()`

xmlui.setVisible()

Disponibilidad

Flash 8.

Uso

```
xmlui.setVisible(controlID, visible)
```

Parámetros

controlID Una cadena que especifica el atributo de identificación del control que desea mostrar u ocultar.

visible Un valor booleano de `true` si desea mostrar el control o de `false` si desea ocultarlo.

Valor devuelto

Ninguno.

Descripción

Método; muestra u oculta un control.

Ejemplo

El siguiente ejemplo oculta el control con el atributo de identificación `myControl`:

```
fl.xmlui.setVisible("myControl", false);
```

Véase también

`xmlui.getVisible()`

Objeto VideoItem

Herencia Objeto Item > Objeto VideoItem

Disponibilidad

Flash MX 2004.

Descripción

El objeto VideoItem es una subclase del Objeto Item.

Resumen de propiedades del objeto VideoItem

Además de las propiedades del Objeto Item, puede emplear las propiedades siguientes con el objeto VideoItem:

Propiedad	Descripción
`videoItem.sourceFilePath`	De sólo lectura; una cadena que especifica la ruta al elemento de vídeo.
`videoItem.videoType`	De sólo lectura; una cadena que especifica el tipo de vídeo que representa el elemento.

videoItem.sourceFilePath

Disponibilidad

Flash 8.

Uso

```
videoItem.sourceFilePath
```

Descripción

Propiedad de sólo lectura; una cadena, expresada como archivo:/// URI, que especifica la ruta al elemento de vídeo.

Ejemplo

El ejemplo siguiente muestra el nombre y la ruta del archivo de origen de todos los elementos de la biblioteca que sean del tipo `"video"`:

```
for ( idx in fl.getDocumentDOM().library.items ) {
  if ( fl.getDocumentDOM().library.items[idx].itemType == "video" ) {
      var myItem = fl.getDocumentDOM().library.items[idx];
      fl.trace( myItem.name + " source is " + myItem.sourceFilePath );
  }
}
```

Véase también

`library.items`

videoItem.videoType

Disponibilidad

Flash 8.

Uso

```
videoItem.videoType
```

Descripción

Propiedad de sólo lectura; una cadena que especifica el tipo de vídeo que representa el elemento. Los valores posibles son: `"embedded video"`, `"linked video"` y `"video"`.

Ejemplo

El ejemplo siguiente muestra el nombre y el tipo de todos los elementos de la biblioteca que sean del tipo `"video"`:

```
for ( idx in fl.getDocumentDOM().library.items ) {
  if ( fl.getDocumentDOM().library.items[idx].itemType == "video" ) {
      var myItem = fl.getDocumentDOM().library.items[idx];
      fl.trace( myItem.name + " is " + myItem.videoType );
  }
}
```

Véase también

`library.items`

CAPÍTULO 3

Extensibilidad de nivel C

3

El mecanismo de extensibilidad de nivel C permite implementar los archivos de extensibilidad de Macromedia Flash empleando una combinación de código JavaScript y C personalizado. Las funciones se definen utilizando C, se integran en una biblioteca de vínculos dinámicos (DLL) o en una biblioteca compartida, se guarda la biblioteca en el directorio adecuado y, a continuación, se llama a las funciones desde JavaScript mediante la API JavaScript de Macromedia Flash.

Por ejemplo, puede definir una función que realice cálculos intensos con mayor eficiencia que JavaScript, que mejore el rendimiento o cuando desee crear herramientas o efectos más avanzados.

Este mecanismo de extensibilidad es un subconjunto de la API Macromedia Dreamweaver. Si está familiarizado con esa API, puede reconocer las funciones de esta API. Sin embargo, esta API se diferencia de la API de Dreamweaver en lo siguiente:

- Esta API no contiene todos los comandos de la API de Dreamweaver.
- Todas las declaraciones del tipo `wchar_t` y `char` de la API de Dreamweaver se implementan como declaraciones `unsigned short` en esta API por compatibilidad con Unicode cuando se transfieren cadenas.
- La función `JSVal JS_BytesToValue()` de esta API no forma parte de la API de Dreamweaver.
- La ubicación donde se deben almacenar los archivos DLL o de biblioteca compartida es distinta (consulte “Integración de las funciones de C” en la página 544).

Integración de las funciones de C

El mecanismo de extensibilidad de nivel C permite implementar los archivos de extensibilidad de Flash empleando una combinación de código JavaScript y C. El proceso para implementar este mecanismo se resume en los pasos siguientes:

1. Defina funciones utilizando el lenguaje C o C++.
2. Intégrelas en un archivo DLL (Windows) o en una biblioteca compartida (Macintosh).
3. Guarde el archivo DLL o la biblioteca en la ubicación correspondiente:
   - Windows 2000 o Windows XP:

     unidad de inicio\Documents and Settings*usuario*\Configuración local\Datos de programa\Macromedia\ Flash 8*idioma*\Configuration\External Libraries
   - Macintosh OS X:

     Disco duro de Macintosh/Users/*nombreUsuario*/Library/Application Support/Macromedia/Flash 8/*idioma*/Configuration/External Libraries
4. Cree un archivo JSFL que llame a las funciones.
5. Ejecute el archivo JSFL desde el menú Comandos en el entorno de edición de Flash.

Para más información, consulte "Implementación de DLL de muestra" en la página 550.

Extensibilidad de nivel C y el intérprete de JavaScript

El código C de la DLL o la biblioteca compartida interactúa con la API JavaScript de Flash en tres momentos distintos:

- Al inicio, para registrar las funciones de la biblioteca
- Cuando se llama a la función C, para desempaquetar los argumentos que se transfieren de JavaScript a C
- Antes de que regrese la función C, para desempaquetar el valor devuelto

Para realizar estas tareas, el intérprete define varios tipos de datos y expone una API. Las definiciones de los tipos de datos y las funciones que aparecen en esta sección se encuentran en el archivo mm_jsapi.h. Para que la biblioteca funcione correctamente deberá incluir el archivo mm_jsapi.h en la parte superior de cada archivo de la biblioteca, con la línea siguiente:

```
#include "mm_jsapi.h"
```

Al incluir el archivo mm_jsapi.h, se incluirá el archivo mm_jsapi_environment.h, que define la estructura de MM_Environment.

Para obtener una copia del archivo mm_jsapi.h, extráigalo del archivo ZIP o SIT de muestra (consulte "Implementación de DLL de muestra" en la página 550) o copie el código siguiente en un archivo con el nombre mm_jsapi.h:

```
#ifndef _MM_JSAPI_H_
#define _MM_JSAPI_H_

/
  ************************************************************************
  *****
 * Tipos de datos públicos

  ************************************************************************
  ****/

typedef struct JSContext JSContext;
typedef struct JSObject JSObject;
typedef long jsval;
#ifndef JSBool
typedef long JSBool;
#endif

typedef JSBool (*JSNative)(JSContext *cx, JSObject *obj, unsigned int argc,
    jsval *argv, jsval *rval);

/* Valores posibles para JSBool */
#define JS_TRUE 1
#define JS_FALSE 0



/
  ************************************************************************
  *****
 * Funciones públicas

  ************************************************************************
  ****/

/* JSBool JS_DefineFunction(unsigned short *name, JSNative call, unsigned
  int nargs) */
#define JS_DefineFunction(n, c, a) \
    (mmEnv.defineFunction ? (*(mmEnv.defineFunction))(mmEnv.libObj, n, c,
  a) \
                          : JS_FALSE)
```

```
/* unsigned short *JS_ValueToString(JSContext *cx, jsval v, unsigned int
  *pLength) */
#define JS_ValueToString(c, v, l) \
    (mmEnv.valueToString  ? (*(mmEnv.valueToString))(c, v, l) : (unsigned
  short *)0)

/* unsigned char *JS_ValueToBytes(JSContext *cx, jsval v, unsigned int
  *pLength) */
#define JS_ValueToBytes(c, v, l) \
   (mmEnv.valueToBytes  ? (*(mmEnv.valueToBytes))(c, v, l) : (unsigned char
  *)0)

/* JSBool JS_ValueToInteger(JSContext *cx, jsval v, long *lp); */
#define JS_ValueToInteger(c, v, l) \
    (mmEnv.valueToInteger ? (*(mmEnv.valueToInteger))(c, v, l) : JS_FALSE)

/* JSBool JS_ValueToDouble(JSContext *cx, jsval v, double *dp); */
#define JS_ValueToDouble(c, v, d) \
    (mmEnv.valueToDouble  ? (*(mmEnv.valueToDouble))(c, v, d) : JS_FALSE)

/* JSBool JS_ValueToBoolean(JSContext *cx, jsval v, JSBool *bp); */
#define JS_ValueToBoolean(c, v, b) \
    (mmEnv.valueToBoolean ? (*(mmEnv.valueToBoolean))(c, v, b) : JS_FALSE)

/* JSBool JS_ValueToObject(JSContext *cx, jsval v, JSObject **op); */
#define JS_ValueToObject(c, v, o) \
    (mmEnv.valueToObject  ? (*(mmEnv.valueToObject))(c, v, o) : JS_FALSE)

/* JSBool JS_StringToValue(JSContext *cx, unsigned short *bytes, uint sz,
  jsval *vp); */
#define JS_StringToValue(c, b, s, v) \
   (mmEnv.stringToValue  ? (*(mmEnv.stringToValue))(c, b, s, v) : JS_FALSE)

/* JSBool JS_BytesToValue(JSContext *cx, unsigned char *bytes, uint sz,
  jsval *vp); */
#define JS_BytesToValue(c, b, s, v) \
    (mmEnv.bytesToValue  ? (*(mmEnv.bytesToValue))(c, b, s, v) : JS_FALSE)

/* JSBool JS_DoubleToValue(JSContext *cx, double dv, jsval *vp); */
#define JS_DoubleToValue(c, d, v) \
    (mmEnv.doubleToValue  ? (*(mmEnv.doubleToValue))(c, d, v) : JS_FALSE)

/* jsval JS_IntegerToValue(long lv); */
#define JS_IntegerToValue(lv)       (((jsval)(lv) << 1) | 0x1)

/* jsval JS_BooleanToValue(JSBool bv); */
#define JS_BooleanToValue(bv)       (((jsval)(bv) << 3) | 0x6)
```

```
/* jsval JS_ObjectToValue(JSObject *obj); */
#define JS_ObjectToValue(ov)        ((jsval)(ov))

/* unsigned short *JS_ObjectType(JSObject *obj); */
#define JS_ObjectType(o) \
    (mmEnv.objectType     ? (*(mmEnv.objectType))(o) : (unsigned short *)0)

/* JSObject *JS_NewArrayObject(JSContext *cx, unsigned int length, jsval
  *v) */
#define JS_NewArrayObject(c, l, v) \
    (mmEnv.newArrayObject ? (*(mmEnv.newArrayObject))(c, l, v) : (JSObject
  *)0)

/* long JS_GetArrayLength(JSContext *cx, JSObject *obj) */
#define JS_GetArrayLength(c, o) \
    (mmEnv.getArrayLength ? (*(mmEnv.getArrayLength))(c, o) : -1)

/* JSBool JS_GetElement(JSContext *cx, JSObject *obj, jsint idx, jsval *vp)
  */
#define JS_GetElement(c, o, i, v) \
    (mmEnv.getElement     ? (*(mmEnv.getElement))(c, o, i, v) : JS_FALSE)

/* JSBool JS_SetElement(JSContext *cx, JSObject *obj, jsint idx, jsval *vp)
  */
#define JS_SetElement(c, o, i, v) \
    (mmEnv.setElement     ? (*(mmEnv.setElement))(c, o, i, v) : JS_FALSE)

/* JSBool JS_ExecuteScript(JSContext *cx, JSObject *obj, unsigned short
  *script,
 *     unsigned int sz, jsval *rval) */
#define JS_ExecuteScript(c, o, s, z, r) \
    (mmEnv.executeScript  ? (*(mmEnv.executeScript))(c, o, s, z,
  _T(__FILE__), \
        __LINE__, r) : JS_FALSE)

/* JSBool JS_ReportError(JSContext *cx, unsigned short *error, unsigned int
  sz) */
#define JS_ReportError(c, e, s) \
    (mmEnv.reportError    ? (*(mmEnv.reportError))(c, e, s) : JS_FALSE)
```

```
/
  ************************************************************************
  *****
 * Tipos de datos privados, macros y globales

  ************************************************************************
  ****/

typedef struct {
    JSObject *libObj;
    JSBool (*defineFunction)(JSObject *libObj, unsigned short *name,
  JSNative call,
        unsigned int nargs);
    unsigned short *(*valueToString)(JSContext *cx, jsval v, unsigned int
  *pLength);
    unsigned char *(*valueToBytes)(JSContext *cx, jsval v, unsigned int
  *pLength);
    JSBool (*valueToInteger)(JSContext *cx, jsval v, long *lp);
    JSBool (*valueToDouble)(JSContext *cx, jsval v, double *dp);
    JSBool (*valueToBoolean)(JSContext *cx, jsval v, JSBool *bp);
    JSBool (*valueToObject)(JSContext *cx, jsval v, JSObject **op);
    JSBool (*stringToValue)(JSContext *cx, unsigned short *b, unsigned int
  sz, jsval *vp);
     JSBool (*bytesToValue)(JSContext *cx, unsigned char *b, unsigned int sz,
  jsval *vp);
    JSBool (*doubleToValue)(JSContext *cx, double dv, jsval *vp);
    unsigned short *(*objectType)(JSObject *obj);
    JSObject *(*newArrayObject)(JSContext *cx, unsigned int length, jsval
  *vp);
    long (*getArrayLength)(JSContext *cx, JSObject *obj);
    JSBool (*getElement)(JSContext *cx, JSObject *obj, unsigned int idx,
        jsval *vp);
    JSBool (*setElement)(JSContext *cx, JSObject *obj, unsigned int idx,
        jsval *vp);
    JSBool (*executeScript)(JSContext *cx, JSObject *obj, unsigned short
  *script,
        unsigned int sz, unsigned short *file, unsigned int lineNum, jsval
  *rval);
     JSBool (*reportError)(JSContext *cx, unsigned short *error, unsigned int
  sz);
} MM_Environment;

extern MM_Environment mmEnv;

// Declara la vinculación y el punto de entrada externo.
#ifdef _WIN32
#   ifndef _MAC
    // Windows
__declspec( dllexport ) void MM_InitWrapper( MM_Environment *env, unsigned
  int envSize );
```

```
#   else
    // Mac con biblioteca de portabilidad MSVC++ Win32
    extern void MM_InitWrapper( MM_Environment *env, unsigned int envSize );
#   endif
#else
    // Codewarrior
#   pragma export on
    extern void MM_InitWrapper( MM_Environment *env, unsigned int envSize );
#   pragma export off
#endif


#define MM_STATE                                                              \
    /* Definiciones de variables globales */
  \
    MM_Environment mmEnv;                                                     \
                                                                              \
    void                                                                      \
   MM_InitWrapper(MM_Environment *env, unsigned int envSize)
  \
    {                                                                         \
        extern void MM_Init();                                                \
                                                                              \
        char **envPtr = (char **)env;
  \
        char **mmPtr =  (char **)(&mmEnv);
  \
       char **envEnd = (char **)((char *)envPtr + envSize);
  \
      char **mmEnd =  (char **)((char *)mmPtr  + sizeof(MM_Environment));
  \
                                                                              \
        /* Copie los campos de env a mmEnv, un puntero por vez */
  \
        while (mmPtr < mmEnd && envPtr < envEnd)
  \
            *mmPtr++ = *envPtr++;                                             \
                                                                              \
     /* Si env no define todos los campos de mmEnv, defina extras como NULL
  */     \
        while (mmPtr < mmEnd)                                                 \
            *mmPtr++ = (char *)0;                                             \
                                                                              \
    /* Llamada a la función MM_Init del usuario */
  \
      MM_Init();                                                              \
    }                                                                         \

#endif /* _MM_JSAPI_H_ */
```

Implementación de DLL de muestra

Una implementación de DLL de muestra se ubica en los archivos ZIP y SIT de la carpeta ExtendingFlash/dllSampleComputeSum (consulte "Implementaciones de muestra" en la página 20). Para ver cómo funciona el proceso sin crear la DLL puede realizar lo siguiente:

- Guarde el archivo Sample.jsfl en el directorio Commands (consulte "Almacenamiento de archivos JSFL" en la página 7).
- Guarde el archivo Sample.dll en el directorio External Libraries (consulte "Integración de las funciones de C" en la página 544).
- En el entorno de edición de Flash, seleccione Comandos > Muestra. La sentencia trace del archivo JSFL envía los resultados de la función definida en Sample.dll al panel Salida.

En esta sección se analiza el desarrollo de la muestra. En este caso, la DLL sólo contiene una función, que suma dos números. El código C se muestra en el siguiente ejemplo:

```
// Código fuente en C
// Guarda la DLL o la biblioteca compartida con el nombre de "Sample".
#include <windows.h>
#include <stdlib.h>

#include "mm_jsapi.h"

// Una función de muestra
// Todas las implementaciones de una función de JavaScript deben tener esta
  firma.
JSBool computeSum(JSContext *cx, JSObject *obj, unsigned int argc, jsval
  *argv, jsval *rval)
{
  long a, b, sum;

  // Compruebe que se ha transferido el número adecuado de argumentos.
  if (argc != 2)
    return JS_FALSE;

  // Convierta los dos argumentos de jsvals a longs.
  if (JS_ValueToInteger(cx, argv[0], &a) == JS_FALSE ||
    JS_ValueToInteger(cx, argv[1], &b) == JS_FALSE)
      return JS_FALSE;

  /* Realice el trabajo real. */
  sum = a + b;

  /* Empaquete el valor devuelto como jsval. */
  *rval = JS_IntegerToValue(sum);

  /* Indica que se ha realizado correctamente. */
  return JS_TRUE;
}
```

Después de escribir este código, cree el archivo DLL o la biblioteca compartida y guárdelo en el directorio External Libraries correspondiente (consulte “Integración de las funciones de C” en la página 544). A continuación, cree un archivo JSFL con el código siguiente y guárdelo en el directorio Commands (consulte “Almacenamiento de archivos JSFL” en la página 7).

```
// Archivo JSFL para ejecutar la función C definida anteriormente.
var a = 5;
var b = 10;
var sum = Sample.computeSum(a, b);
fl.trace("The sum of " + a + " and " + b + " is " + sum );
```

Para ejecutar la función definida en la DLL, seleccione Comandos > Muestra en el entorno de edición de Flash.

Tipos de datos

El intérprete de JavaScript define los siguientes tipos de datos:

- `JSContext`
- `JSObject`
- `jsval`
- `JSBool`

typedef struct JSContext JSContext

Se pasa un puntero a este tipo de datos opacos a la función de nivel C. Algunas funciones de la API aceptan este puntero como uno de sus argumentos.

typedef struct JSObject JSObject

Se pasa un puntero a este tipo de datos opacos a la función de nivel C. Este tipo de datos representa un objeto, que puede ser un objeto de matriz o algún otro tipo de objeto.

typedef struct jsval jsval

Una estructura de datos opacos que puede contener un entero, o un puntero a un valor flotante, una cadena o un objeto. Algunas funciones de la API pueden leer los valores de argumentos de función leyendo el contenido de una estructura `jsval` y algunas pueden emplearse para escribir el valor devuelto de la función escribiendo una estructura `jsval`.

typedef enum { JS_FALSE = 0, JS_TRUE = 1 } JSBool

Un tipo de datos simples que almacena un valor booleano.

La API de nivel C

La API de extensibilidad de nivel C se compone de la firma de función `JSBool (*JSNative)` y las funciones siguientes:

- `JSBool JS_DefineFunction()`
- `unsigned short *JS_ValueToString()`
- `JSBool JS_ValueToInteger()`
- `JSBool JS_ValueToDouble()`
- `JSBool JS_ValueToBoolean()`
- `JSBool JS_ValueToObject()`
- `JSBool JS_StringToValue()`
- `JSBool JS_DoubleToValue()`
- `JSVal JS_BooleanToValue()`
- `JSVal JS_BytesToValue()`
- `JSVal JS_IntegerToValue()`
- `JSVal JS_ObjectToValue()`
- `unsigned short *JS_ObjectType()`
- `JSObject *JS_NewArrayObject()`
- `long JS_GetArrayLength()`
- `JSBool JS_GetElement()`
- `JSBool JS_SetElement()`
- `JSBool JS_ExecuteScript()`

typedef JSBool (*JSNative)(JSContext *cx, JSObject *obj, unsigned int argc, jsval *argv, jsval *rval)

Descripción

Método; describe implementaciones de nivel C de funciones de JavaScript en las situaciones siguientes:

- El puntero *cx* es un puntero a una estructura `JSContext` opaca que debe transferirse a algunas de las funciones de la API JavaScript. Esta variable contiene el contexto de ejecución del intérprete.
- El puntero *obj* es un puntero al objeto en cuyo contexto se ejecuta el script. Mientras se ejecuta el script, la palabra clave `this` es igual a este objeto.
- El entero *argc* es el número de argumentos que se transfieren a la función.
- El puntero *argv* es un puntero a una matriz de estructuras `jsval`. La matriz es elementos `argc` en longitud.
- El puntero *rval* es un puntero a una estructura `jsval` única. El valor devuelto de la función debe escribirse en `*rval`.

La función devuelve `JS_TRUE` si se ha ejecutado correctamente, y `JS_FALSE` en caso contrario. Si la función devuelve `JS_FALSE`, el script actual dejará de ejecutarse y aparecerá un mensaje de error.

JSBool JS_DefineFunction()

Uso

```
JSBool JS_DefineFunction(unsigned short *name, JSNative call, unsigned int
  nargs)
```

Descripción

Método; registra una función de nivel C con el intérprete de JavaScript en Flash. Cuando la función `JS_DefineFunction()` registre la función de nivel C que usted especifica en el argumento *call*, puede invocarla en un script de JavaScript haciendo referencia a ella con el nombre que especifique en el argumento *name*. El argumento *name* tiene en cuenta el uso de mayúsculas y minúsculas.

Esta función suele llamarse desde la función `MM_Init()`, a la que Flash llama en el inicio.

Argumentos

`unsigned short *name`, `JSNative call`, `unsigned int nargs`

- El argumento `name` es el nombre de la función tal como se expone a JavaScript.
- El argumento `call` es un puntero a una función de nivel C. La función debe devolver `JSBool`, que indica si se ha ejecutado correcta o incorrectamente.
- El argumento `nargs` es el número de argumentos que la función espera recibir.

Valor devuelto

Un valor booleano: `JS_TRUE` indica que se ha ejecutado correctamente; `JS_FALSE` indica que se ha ejecutado incorrectamente.

unsigned short *JS_ValueToString()

Uso

```
unsigned short *JS_ValueToString(JSContext *cx, jsval v,
  unsigned int *pLength)
```

Descripción

Método; extrae un argumento de función de una estructura `jsval`, lo convierte en una cadena, si es posible, y devuelve el valor convertido al llamador.

NOTA	Si modifica el puntero del búfer devuelto puede dañar las estructuras de datos del intérprete de JavaScript. Para cambiar la cadena deberá copiar los caracteres en otro búfer y crear una nueva cadena de JavaScript.

Argumentos

`JSContext *cx`, `jsval v`, `unsigned int *pLength`

- El argumento `cx` es el puntero `JSContext` opaco que se transfiere a la función de JavaScript.
- El argumento `v` es la estructura `jsval` de la que va a extraerse la cadena.
- El argumento `pLength` es un puntero a un entero sin signo. Esta función define `*plength` con la longitud de la cadena en bytes.

Valor devuelto

Un puntero que señala a una cadena terminada en cero si se ha ejecutado correctamente o a un valor `null` si se ha ejecutado incorrectamente. La rutina de llamada no debe liberar esta cadena cuando termina.

JSBool JS_ValueToInteger()

Uso

```
JSBool JS_ValueToInteger(JSContext *cx, jsval v, long *lp);
```

Descripción

Método; extrae un argumento de función de una estructura `jsval`, lo convierte en un entero (si es posible) y devuelve el valor convertido al llamador.

Argumentos

`JSContext *cx`, `jsval v`, `long *lp`

- El argumento *cx* es el puntero `JSContext` opaco que se transfiere a la función de JavaScript.
- El argumento *v* es la estructura `jsval` de la que va a extraerse el entero.
- El argumento *lp* es un puntero a un entero de 4 bytes. Esta función almacena el valor convertido en `*lp`.

Valor devuelto

Un valor booleano: `JS_TRUE` indica que se ha ejecutado correctamente; `JS_FALSE` indica que se ha ejecutado incorrectamente.

JSBool JS_ValueToDouble()

Uso

```
JSBool JS_ValueToDouble(JSContext *cx, jsval v, double *dp);
```

Descripción

Método; extrae un argumento de función de una estructura `jsval`, lo convierte en un valor double (si es posible) y devuelve el valor convertido al llamador.

Argumentos

`JSContext *cx`, `jsval v`, `double *dp`

- El argumento *cx* es el puntero `JSContext` opaco que se transfiere a la función de JavaScript.
- El argumento *v* es la estructura `jsval` de la que va a extraerse el valor double.
- El argumento *dp* es un puntero a un valor double de 8 bytes. Esta función almacena el valor convertido en `*dp`.

Valor devuelto

Un valor booleano: `JS_TRUE` indica que se ha ejecutado correctamente; `JS_FALSE` indica que se ha ejecutado incorrectamente.

JSBool JS_ValueToBoolean()

Uso

```
JSBool JS_ValueToBoolean(JSContext *cx, jsval v, JSBool *bp);
```

Descripción

Método; extrae un argumento de función de una estructura `jsval`, lo convierte en un valor booleano (si es posible) y devuelve el valor convertido al llamador.

Argumentos

`JSContext` *`*cx`*, `jsval` *`v`*, `JSBool` *`*bp`*

- El argumento *`cx`* es el puntero `JSContext` opaco que se transfiere a la función de JavaScript.
- El argumento *`v`* es la estructura `jsval` de la que va a extraerse el valor booleano.
- El argumento *`bp`* es un puntero a un valor booleano `JSBool`. Esta función almacena el valor convertido en `*bp`.

Valor devuelto

Un valor booleano: `JS_TRUE` indica que se ha ejecutado correctamente; `JS_FALSE` indica que se ha ejecutado incorrectamente.

JSBool JS_ValueToObject()

Uso

```
JSBool JS_ValueToObject(JSContext *cx, jsval v, JSObject **op);
```

Descripción

Método; extrae un argumento de función de una estructura `jsval`, lo convierte en un objeto (si es posible) y devuelve el valor convertido al llamador. Si el objeto es una matriz, utilice `JS_GetArrayLength()` y `JS_GetElement()` para leer su contenido.

Argumentos

`JSContext *cx`, `jsval v`, `JSObject **op`

- El argumento *cx* es el puntero `JSContext` opaco que se transfiere a la función de JavaScript.
- El argumento *v* es la estructura `jsval` de la que va a extraerse el objeto.
- El argumento *op* es un puntero a un puntero `JSObject`. Esta función almacena el valor convertido en `*op`.

Valor devuelto

Un valor booleano: `JS_TRUE` indica que se ha ejecutado correctamente; `JS_FALSE` indica que se ha ejecutado incorrectamente.

JSBool JS_StringToValue()

Uso

```
JSBool JS_StringToValue(JSContext *cx, unsigned short *bytes, uint sz,
  jsval *vp);
```

Descripción

Método; almacena un valor devuelto de cadena en una estructura `jsval`. Asigna un nuevo objeto de cadena de JavaScript.

Argumentos

`JSContext *cx`, `unsigned short *bytes`, `size_t sz`, `jsval *vp`

- El argumento *cx* es el puntero `JSContext` opaco que se transfiere a la función de JavaScript.
- El argumento *bytes* es la cadena que se va a almacenar en la estructura `jsval`. Los datos de la cadena se copian, por lo que el llamador debe liberar la cadena cuando no es necesaria. Si no se especifica el tamaño de la cadena (consulte el argumento *sz*), la cadena debe terminar en-cero.
- El argumento *sz* es el tamaño de la cadena, expresado en bytes. Si *sz* es 0, la longitud de la cadena terminada-en cero se calcula automáticamente.
- El argumento *vp* es un puntero a la estructura `jsval` en la que debe copiarse el contenido de la cadena.

Valor devuelto

Un valor booleano: `JS_TRUE` indica que se ha ejecutado correctamente; `JS_FALSE` indica que se ha ejecutado incorrectamente.

JSBool JS_DoubleToValue()

Uso

```
JSBool JS_DoubleToValue(JSContext *cx, double dv, jsval *vp);
```

Descripción

Método; almacena un valor devuelto de número de coma flotante en una estructura `jsval`.

Argumentos

`JSContext *cx`, `double dv`, `jsval *vp`

- El argumento *`cx`* es el puntero `JSContext` opaco que se transfiere a la función de JavaScript.
- El argumento *`dv`* es un número de coma flotante de 8 bytes.
- El argumento *`vp`* es un puntero a la estructura `jsval` en la que debe copiarse el contenido del valor double.

Valor devuelto

Un valor booleano: `JS_TRUE` indica que se ha ejecutado correctamente; `JS_FALSE` indica que se ha ejecutado incorrectamente.

JSVal JS_BooleanToValue()

Uso

```
jsval JS_BooleanToValue(JSBool bv);
```

Descripción

Método; almacena un valor devuelto booleano en una estructura `jsval`.

Argumentos

`JSBool bv`

- El argumento *`bv`* es un valor booleano: `JS_TRUE` indica que se ha ejecutado correctamente; `JS_FALSE` indica que se ha ejecutado incorrectamente.

Valor devuelto

Una estructura `JSVal` que contiene el valor booleano que se transfiere a la función como argumento.

JSVal JS_BytesToValue()

Uso

```
JSBool JS_BytesToValue(JSContext *cx, unsigned short *bytes, uint sz, jsval
  *vp);
```

Descripción

Método; convierte los bytes en un valor de JavaScript.

Argumentos

`JSContext *`*`cx`*, `unsigned short` *`bytes`*, `uint` *`sz`*, `jsval *`*`vp`*

- El argumento *`cx`* es el contexto de JavaScript.
- El argumento *`bytes`* es la cadena de bytes que se va a convertir en un objeto JavaScript.
- El argumento *`sz`* es el número de bytes que se va a convertir.
- El argumento *`vp`* es el valor de JavaScript.

Valor devuelto

Un valor booleano: `JS_TRUE` indica que se ha ejecutado correctamente; `JS_FALSE` indica que se ha ejecutado incorrectamente.

JSVal JS_IntegerToValue()

Uso

```
jsval JS_IntegerToValue(long lv);
```

Descripción

Método; convierte un valor entero largo en una estructura `JSVal`.

Argumentos

`lv`

- El argumento *`lv`* es el valor entero largo que desea convertir en una estructura `jsval`.

Valor devuelto

Una estructura `JSVal` que contiene el entero que se transfiere a la función como argumento.

JSVal JS_ObjectToValue()

Uso

```
jsval JS_ObjectToValue(JSObject *obj);
```

Descripción

Método; almacena un valor devuelto de objeto en una estructura `JSVal`. Utilice `JS_NewArrayObject()` para crear un objeto de matriz; utilice `JS_SetElement()` para definir su contenido.

Argumentos

`JSObject *obj`

- El argumento *`obj`* es un puntero al objeto `JSObject` que desea convertir en una estructura `JSVal`.

Valor devuelto

Una estructura `JSVal` que contiene el objeto que ha transferido a la función como argumento.

unsigned short *JS_ObjectType()

Uso

```
unsigned short *JS_ObjectType(JSObject *obj);
```

Descripción

Método; si se suministra una referencia a un objeto, devuelve el nombre de clase del objeto. Por ejemplo, si el objeto es un objeto DOM, la función devuelve "`Document`". Si el objeto es un nodo del documento, la función devuelve "`Element`". Para un objeto de matriz, la función devuelve "`Array`".

NOTA	Si modifica el puntero del búfer devuelto puede dañar las estructuras de datos del intérprete de JavaScript.

Argumentos

`JSObject *obj`

- Este argumento se suele transferir y convertir empleando la función `JS_ValueToObject()`.

Valor devuelto

Un puntero a una cadena terminada en cero. El llamador no debe liberar esta cadena al terminar.

JSObject *JS_NewArrayObject()

Uso

```
JSObject *JS_NewArrayObject( JSContext *cx, unsigned int length [, jsval *v
  ] )
```

Descripción

Método; crea un objeto nuevo que contiene una matriz de `JSVals`.

Argumentos

`JSContext *cx`, `unsigned int length`, `jsval *v`

- El argumento *`cx`* es el puntero `JSContext` opaco que se transfiere a la función de JavaScript.
- El argumento *`length`* es el número de elementos que puede contener la matriz.
- El argumento *`v`* es un puntero opcional a `jsvals` que se va a almacenar en la matriz. Si el valor devuelto no es `null`, *`v`* es una matriz que contiene elementos *`length`*. Si el valor devuelto es `null`, el contenido inicial del objeto de matriz es undefined y puede definirse empleando la función `JS_SetElement()`.

Valor devuelto

Un puntero a un nuevo objeto de matriz o al valor `null` si se ejecuta incorrectamente.

long JS_GetArrayLength()

Uso

```
long JS_GetArrayLength(JSContext *cx, JSObject *obj)
```

Descripción

Método; si se suministra un puntero a un objeto de matriz, obtiene el número de elementos de la matriz.

Argumentos

`JSContext *cx`, `JSObject *obj`

- El argumento *`cx`* es el puntero `JSContext` opaco que se transfiere a la función de JavaScript.
- El argumento *`obj`* es un puntero a un objeto de matriz.

Valor devuelto

El número de elementos de la matriz o -1 si se ejecuta incorrectamente.

JSBool JS_GetElement()

Uso

```
JSBool JS_GetElement(JSContext *cx, JSObject *obj, jsint idx, jsval *vp)
```

Descripción

Método; lee un único elemento de un objeto de matriz.

Argumentos

`JSContext *cx`, `JSObject *obj`, `unsigned int index`, `jsval *v`

- El argumento *cx* es el puntero `JSContext` opaco que se transfiere a la función de JavaScript.
- El argumento *obj* es un puntero a un objeto de matriz.
- El argumento *index* es un índice de entero de la matriz. El primer elemento es índice 0 y el último elemento es índice (`length - 1`).
- El argumento *v* es un puntero a un `jsval` donde debe copiarse el contenido de la estructura `jsval` de la matriz.

Valor devuelto

Un valor booleano: `JS_TRUE` indica que se ha ejecutado correctamente; `JS_FALSE` indica que se ha ejecutado incorrectamente.

JSBool JS_SetElement()

Uso

```
JSBool JS_SetElement(JSContext *cx, JSObject *obj, jsint idx, jsval *vp)
```

Descripción

Método; escribe un único elemento de un objeto de matriz.

Argumentos

`JSContext *cx, JSObject *obj, unsigned int index, jsval *v`

- El argumento *cx* es el puntero `JSContext` opaco que se transfiere a la función de JavaScript.
- El argumento *obj* es un puntero a un objeto de matriz.
- El argumento *index* es un índice de entero de la matriz. El primer elemento es índice `0` y el último elemento es índice (`length - 1`).
- El argumento *v* es un puntero a una estructura `jsval` cuyo contenido debe copiarse en el `jsval` de la matriz.

Valor devuelto

Un valor booleano: `JS_TRUE` indica que se ha ejecutado correctamente; `JS_FALSE` indica que se ha ejecutado incorrectamente.

JSBool JS_ExecuteScript()

Uso

```
JS_ExecuteScript (JSContext *cx, JSObject *obj, unsigned short *script,
  unsigned int sz, jsval *rval)
```

Descripción

Método; compila y ejecuta una cadena de JavaScript. Si el script genera un valor devuelto, devuelve `*rval`.

Argumentos

`JSContext` *`*cx`*, `JSObject` *`*obj`*, *`unsigned short *script`*, *`unsigned int sz`*, `jsval` *`*rval`*

- El argumento *`cx`* es el puntero `JSContext` opaco que se transfiere a la función de JavaScript.
- El argumento *`obj`* es un puntero al objeto en cuyo contexto se ejecuta el script. Mientras se ejecuta el script, la palabra clave `this` es igual a este objeto. Generalmente este es el puntero `JSObject` que se transfiere a la función JavaScript.
- El argumento *`script`* es una cadena que contiene código JavaScript. Si no se especifica el tamaño de la cadena (consulte el argumento *`sz`*), la cadena debe terminar en cero.
- El argumento *`sz`* es el tamaño de la cadena, expresado en bytes. Si sz es 0, la longitud de la cadena terminada en cero se calcula automáticamente.
- El argumento *`rval`* es un puntero a una estructura `jsval` única. El valor devuelto de la función se almacena en `*rval`.

Valor devuelto

Un valor booleano: `JS_TRUE` indica que se ha ejecutado correctamente; `JS_FALSE` indica que se ha ejecutado incorrectamente.