macromedia®

FLASH™ 5

Guía de Consulta de ActionScript

macromedia

Nº de pieza ZFL50M200SP

Agradecimientos

Gestión del proyecto: Erick Vera

Redacción: Jody Bleyle, Mary Burger, Louis Dobrozensky, Stephanie Gowin, Marcelle Taylor y Judy Walthers Von Alten

Edición: Peter Fenczik, Rosana Francescato, Ann Szabla

Multimedia: George Brown, John “Zippy” Lehnus y Noah Zilberberg

Diseño de Ayuda e impresión: Chris Basmajian y Noah Zilberberg

Producción: Chris Basmajian y Rebecca Godbois

Gestión del proyecto de localización: Yuko Yagi

Producción de localización: Masayo “Noppe” Noda y Bowne Global Solutions

Agradecimiento especial a: Pablo Francisco Arrieta, Jeremy Clark, Brian Dister y todo el Grupo de Desarrollo de Flash, Michael Dominguez, Margaret Dumas, Rosa Elva Elizondo, Sherri Harte, Yoshika Hedberg, Tim Hussey, Kipling Inscore, Alyn Kelley, Pete Santangeli, Denise Seymour y todo el Grupo de Control de calidad de Flash, Cyn Taylor y Eric Wittman

Primera edición: Septiembre de 2000

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

TABLE DES MATIÈRES

CAPÍTULO 4

CAPÍTULO 5

CAPÍTULO 6

CAPÍTULO 7

INTRODUCTION

Procedimientos iniciales

ActionScript es el lenguaje de creación de scripts de Flash. Puede utilizar ActionScript para controlar objetos en las películas de Flash con el fin de crear elementos interactivos y de navegación y para ampliar Flash con el fin de crear películas altamente interactivas y aplicaciones Web.

Novedades en el lenguaje ActionScript de Flash 5

El lenguaje ActionScript de Flash 5 ofrece nuevas funciones muy interesantes para la creación de sitios Web interactivos con juegos, formularios, encuestas e interactividad en tiempo real sofisticadas como los sistemas de chat.

ActionScript de Flash 5 tiene muchas funciones y convenciones de sintaxis nuevas que lo hacen similar en lo fundamental al lenguaje de programación JavaScript. En este manual se explicarán los conceptos básicos de programación como funciones, variables, sentencias, operadores, condicionales y bucles. El Capítulo 7 de esta manual, “Diccionario de ActionScript”, contiene entradas detalladas para cada elemento de ActionScript.

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

La Asociación de Fabricantes de PCs Europea (ECMA, European Computers Manufacturers Association) publicó un documento llamado ECMA-262 que se deriva de JavaScript y que sirve como el estándar internacional para el lenguaje JavaScript. ActionScript está basado en la especificación ECMA-262 que se encuentra disponible en http://www.ecma.ch.

Netscape DevEdge Online tiene un centro de desarrollo de JavaScript (JavaScript Developer Central http://developer.netscape.com/tech/javascript/index.html) que contiene documentación y artículos útiles para comprender ActionScript. El recurso más valioso es el manual Core JavaScript Guide que se encuentra en la dirección http://developer.netscape.com/docs/manuals/js/core/jsguide/index.htm.

Diferencias entre ActionScript y JavaScript

No es necesario que conozca JavaScript para utilizar y aprender ActionScript. Sin embargo, si ya conoce JavaScript, ActionScript le parecerá familiar. Algunas de las diferencias entre ActionScript y JavaScript se detallan a continuación:

- ActionScript no admite objetos específicos de navegador como Documento, Ventana y Ancla.
- ActionScript no admite por completo todos los objetos predefinidos de JavaScript.
- ActionScript admite las construcciones sintácticas que no se permiten en JavaScript (por ejemplo, las acciones `tellTarget` y `ifFrameLoaded` y la sintaxis de barras).
- ActionScript no admite algunas de las construcciones sintácticas de JavaScript, como los rótulos `switch`, `continue`, `try`, `catch`, `throw` y `statement`.
- ActionScript no admite el constructor de JavaScript `Function`.
- En ActionScript, la acción `eval` solamente puede realizar referencias de variables.
- En JavaScript, `toString` de `undefined` es `undefined`. En Flash 5, para su compatibilidad con Flash 4, `toString` de `undefined` es `" "`.
- En JavaScript, evaluar `undefined` en un contexto numérico da como resultado `NaN`. En Flash 5, por compatibilidad con Flash 4 , evaluar `undefined` da como resultado `0`.
- ActionScript no admite Unicode; admite ISO-8859-1 y conjuntos de caracteres Shift-JIS.

Edición de texto

Puede introducir scripts directamente en el panel Acciones en Modo Experto. También puede elegir elementos de un menú emergente o de una lista de la Caja de herramientas como lo hacía en Flash 4.

Sintaxis de punto

Puede utilizar la sintaxis de punto para obtener y establecer las propiedades y los métodos de un objeto, incluidas instancias y variables de clip de película. Puede utilizar la sintaxis de punto en lugar de la sintaxis de barras que se utilizaba en Flash 4. Ya no se utiliza la sintaxis de barras, pero aún la admite Flash Player.

Tipos de datos

El lenguaje ActionScript de Flash 5 admite los siguientes tipos de datos: cadena, número, booleano, objeto y clip de película. Los distintos tipos de datos le permiten utilizar diferentes tipos de información en ActionScript. Por ejemplo, puede crear matrices y matrices asociativas.

Variables locales

Puede declarar variables locales que expiran al final de la lista de acciones o de la llamada de función. Esto le permite gestionar la memoria y volver a utilizar los nombres de variables. Todas las variables de Flash 4 eran permanentes; incluso las variables temporales como los contadores de bucle que permanecían en la película hasta que finalizaba.

Funciones definidas por el usuario

Puede definir funciones con parámetros que devuelven valores. Esto le permite volver a utilizar los bloques de código en los scripts. En Flash 4, podía volver a utilizar el código mediante la acción `call` pero no podía pasar parámetros ni devolver valores.

Objetos predefinidos

Puede utilizar los objetos predefinidos para acceder y manipular ciertos tipos de información. A continuación se muestran algunos de los objetos predefinidos:

- El objeto Math dispone de un complemento completo de constantes y funciones matemáticas integradas como `E` (constante de Euler), `cos` (Coseno), y `atan` (arco tangente).
- El objeto Date le permite obtener información sobre la fecha y la hora del sistema en el que se esté ejecutando Flash Player.
- El objeto Sound le permite agregar sonidos a una película y controlar los sonidos en una película a medida que se reproduce. Por ejemplo, puede ajustar el volumen (`setVolume`), o el balance (`setPan`).
- El objeto Mouse le permite ocultar el cursor estándar para que pueda utilizar un cursor personalizado.
- El objeto MovieClip le permite controlar clips de película sin utilizar una acción envolvente como `tellTarget`. Puede llamar a un método como `play`, `loadMovie`, o `duplicateMovieClip` desde un nombre de instancia utilizando la sintaxis de punto (por ejemplo, `myMovieClip.play()`).

Acciones de clip

Puede utilizar la acción `onClipEvent` para asignar acciones directamente a instancias de clip de película en el Escenario. La acción onClipEvent dispone de eventos como `load`, `enterFrame`, `mouseMove`, y `data` que le permiten crear nuevas clases de interactividad avanzada.

Acciones nuevas

Puede utilizar acciones nuevas como `do..while` y `for` para crear bucles complejos. Otras acciones nuevas se implementan como métodos del objeto MovieClip; por ejemplo, `getBounds`, `attachMovie`, `hitTest`, `swapDepths`, y `globalToLocal`.

Clips inteligentes

Los Clips inteligentes tienen scripts internos que el usuario, u otro autor, puede cambiar sin utilizar el panel Acciones. Puede pasar valores a un Clip inteligente por medio de los parámetros de clip que puede definir en la Biblioteca.

Depurador

El Depurador le permite ver y cambiar los valores de variables y propiedades cuando está reproduciendo una película en el modo de prueba de película, en el reproductor Flash Player independiente o en un navegador Web. Esto le permite encontrar fácilmente los problemas en su programación ActionScript.

Compatibilidad con XML

El objeto XML predefinido le permite convertir ActionScript en documentos XML y pasarlos a las aplicaciones de servidor. También puede utilizar el objeto XML para cargar documentos XML en una película de Flash e interpretarlos. El objeto predefinido XMLSocket le permite crear una conexión continua con el servidor para pasar datos XML para aplicaciones que se ejecutan en tiempo real.

Optimización

Flash 5 lleva a cabo optimizaciones en el código de ActionScript para mejorar su desempeño y no aumentar el tamaño de archivo. Como resultado de estas optimizaciones, Flash 5 con frecuencia producirá código de bytes de ActionScript más pequeño que en Flash 4.

Utilización de la Ayuda de Flash para acciones

Flash 5 contiene ayuda sensible al contexto para cada acción disponible en el panel Acciones. Mientras crea scripts, puede obtener información sobre las acciones que está utilizando.

Para obtener ayuda sobre las acciones:

1. En el panel Acciones, seleccione una acción en la lista de la Caja de herramientas.
2. Haga clic sobre el botón Ayuda en la parte superior del panel.

   En el navegador aparecerá un tema relacionado con la acción.

CAPÍTULO 1
El lenguaje ActionScript

ActionScript, el lenguaje de creación de scripts de Flash, permite crear películas con elementos interactivos. La película se puede configurar para que ejecute scripts que respondan a eventos de usuario como pueden ser hacer clic en el botón del ratón o presionar una tecla. Por ejemplo, se puede crear un script para indicar a Flash que cargue una película específica en Flash Player como respuesta al botón del navegador seleccionado por el usuario.

La filosofía de ActionScript es la de una herramienta que permite crear películas que se comportan exactamente del modo establecido por el usuario. No es necesario entender todos los posibles usos de la herramienta para empezar a crear scripts; si parte de un objetivo claro, podrá empezar a crear scripts que incluyan acciones sencillas. Puede incorporar nuevos elementos del lenguaje a medida que aprenda a utilizarlos, para realizar tareas más complejas.

Este capítulo presenta ActionScript como un lenguaje de creación de scripts orientado a objetos e introduce diferentes términos del mismo. También analiza un script de ejemplo, de forma que el usuario desarrolle un marco de referencia global respecto al mismo.

Este capítulo introduce además el panel Acciones, que permite construir scripts mediante la selección de elementos de ActionScript o introducir texto en la ventana Script.

Creación de scripts en ActionScript

Para empezar a escribir scripts sencillos no es necesario tener un conocimiento profundo de ActionScript. Lo único que se necesita es un objetivo claro; lo demás es sólo cuestión de seleccionar las acciones adecuadas. La mejor forma de descubrir lo fácil que es ActionScript es creando un script. En el siguiente ejemplo se presenta un script que ha sido asignado a un botón para cambiar el estado de visibilidad de un clip de película.

Para cambiar la visibilidad de un clip de película:

1. Seleccione Ventana > Bibliotecas comunes > Botones y, a continuación, seleccione Ventana > Bibliotecas comunes > Clips de película. Coloque un botón y un clip de película en el Escenario.
2. Seleccione la instancia de clip de película en el Escenario, y seleccione Ventana > Paneles > Propiedades de instancia.
3. Introduzca en el campo Nombre **testMC**.
4. Seleccione el botón en el Escenario, y elija Ventana > Acciones para abrir el panel Acciones.
5. En el panel Acciones de objetos, haga clic en la categoría Acciones para abrirla.
6. Haga doble clic en la acción `setProperty` para agregarla a la lista Acciones.
7. En el menú emergente Propiedad, seleccione `_visible` (Visibilidad).
8. Especifique **testMC** como valor del parámetro Destino.
9. Especifique el valor **0** para el parámetro Valor.

   El código resultante deberá ser similar al siguiente:

```
on (release) {
    setProperty ("testMC", _visible, false);
}
```

10. Seleccione Control > Probar película y haga clic en el botón para ver cómo desaparece el clip de película.

ActionScript es un lenguaje de creación de scripts orientado a objetos. Esto significa que, cuando se dan determinados eventos, las acciones controlan objetos. En este script, el evento es la acción del usuario de soltar el botón del ratón (lo que se conoce como "mouse-up"), el objeto es la instancia de clip de película `MC` y la acción es `setProperty`. Cuando el usuario hace clic en el botón que se muestra en la pantalla, el evento `release` ejecuta un script que establece la propiedad `_visible` del objeto `MC` al valor `false` lo cual hace que el objeto se vuelva invisible.

El panel Acciones puede utilizarse como referencia para la configuración de scripts sencillos. Para utilizar toda la potencia de ActionScript, es importante comprender el funcionamiento del lenguaje: los conceptos, elementos y reglas del lenguaje para organizar la información y crear películas interactivas.

Esta sección explica el flujo de trabajo de ActionScript, los conceptos fundamentales para la creación de scripts orientados a objetos, los objetos de Flash y el flujo de ejecución de scripts. También presenta información respecto a la ubicación de los scripts en las películas de Flash.

Planificación y depuración de scripts

Lo más común cuando se programan scripts para una película completa es que se trabaje con una gran cantidad y diversidad de scripts. Es importante planear cuidadosamente qué acciones utilizar en un script, cómo estructurar las instrucciones de forma efectiva y en qué elementos ubicarlos para asegurar el mejor desempeño de los mismos, especialmente cuando la complejidad de la película empieza a ser considerable.

Antes de empezar a programar scripts, es importante establecer un objetivo e intentar clarificar con exactitud lo que se propone conseguir. Esto es tan importante (y normalmente consume tanto tiempo) como el desarrollo de los scripts para el proyecto que se desea realizar. Es recomendable empezar por escribir las acciones que se desea sucedan en la película, como en el siguiente ejemplo:

- Deseo crear mi propio sitio Web utilizando Flash.
- A las personas que visiten el sitio se les pedirá su nombre, y éste se utilizará en los mensajes que se generen en todo el sitio Web.
- El sitio contendrá una barra de navegación que puede arrastrarse, y sus botones se vincularán con cada una de las secciones.
- Cuando se presione un botón, la nueva sección irá apareciendo progresivamente en el centro del Escenario.
- Una escena dispondrá de un formulario de contacto, en el que se mostrará el nombre del usuario.

Cuando tenga una idea clara de lo que desea, podrá construir los objetos necesarios y escribir los scripts que controlen dichos objetos.

Para lograr que los scripts se comporten del modo deseado se tiene que invertir una cantidad considerable de tiempo, en ocasiones es necesario pasar por más de un ciclo de escritura, pruebas y depuración. La mejor forma de abordar un problema suele ser empezar poco a poco y comprobar el trabajo a medida que se avanza. Cuando haya conseguido que funcione una parte del script, seleccione Guardar como, con el fin de guardar una versión del archivo (por ejemplo, miPelic01.fla) y empiece a escribir la siguiente parte. Este modo de trabajar ayuda a identificar errores de un modo eficiente y asegura que el código de ActionScript es fiable antes de proceder a escribir scripts más complejos.

Creación de scripts orientados a objetos

En los scripts orientados a objetos la información se organiza en grupos denominados *clases*. Pueden crearse varias instancias de una clase; a éstas se les denomina *objetos,* y los objetos pueden emplearse en los scripts. El desarrollador puede utilizar en sus scripts clases predefinidas de ActionScript y crear las suyas propias.

Cuando se crea una clase, es preciso definir todas las *propiedades* (características) y *métodos* (comportamientos) de cada objeto que ésta crea, de modo análogo a como se definen los objetos en el mundo real. Una persona, por ejemplo, tiene ciertas propiedades como el género, la altura y el color de pelo, así como ciertos métodos como son hablar, caminar y lanzar. En este ejemplo "persona" es, genéricamente, una clase y cada persona individual es un objeto, o una *instancia* de dicha clase.

Los objetos de ActionScript pueden contener datos o pueden representarse gráficamente en el Escenario como clips de película. Todos los clips de película son instancias de la clase predefinida MovieClip. Cada instancia de clip de película contiene todas las propiedades (por ejemplo, `_height`, `_rotation`, `_totalframes`) y todos los métodos (por ejemplo, `gotoAndPlay`, `loadMovie`, `startDrag`) de la clase MovieClip.

Para definir una clase, debe crearse una función especial, denominada *función constructora*; las clases predefinidas tienen funciones constructoras previamente definidas. Por ejemplo, si se desea obtener información acerca de un ciclista que aparece en la película, puede crearse una función constructora, `Biker`, con las propiedades `time` y `distance` y el método `rate`, que indique la velocidad a la que se desplaza el ciclista:

```
function Biker(t, d) {
   this.time = t;
   this.distance = d;
}
function Speed() {
   return this.time / this.distance;
}
Biker.prototype.rate = Speed;
```

A continuación pueden crearse copias, es decir, instancias, de la clase. El siguiente código crea instancias del objeto `Biker` denominadas `emma` y `hamish`.

```
emma = new Biker(30, 5);
hamish = new Biker(40, 5);
```

Las instancias también pueden comunicarse entre sí. Para el objeto `Biker`, podría crearse un método denominado `shove` que permita a un ciclista empujar a otro (la instancia `emma` podría llamar a su método `shove` en caso de que `hamish` se acercase demasiado). Para pasar información a un método se utilizan parámetros (también denominados argumentos): por ejemplo, el método `shove` podría aceptar los parámetros *`who`* y *`howFar`*. En este ejemplo `emma` empuja a `hamish` 10 píxeles:

```
emma.shove(hamish, 10);
```

En la creación de scripts orientados a objetos, las clases pueden recibir propiedades y métodos unas de otras, de acuerdo a un orden determinado; esta funcionalidad se denomina *herencia*. La herencia puede utilizarse para ampliar o redefinir las propiedades y métodos de una clase. Una clase que hereda propiedades y métodos de otra clase recibe el nombre de *subclase*. Una clase que pasa propiedades y métodos a otra clase recibe el nombre de *superclase*. Una clase puede ser a la vez subclase y superclase.

Objeto MovieClip

Las clases predefinidas de ActionScript reciben el nombre de *Objetos*. Cada objeto permite acceder a un determinado tipo de información. Por ejemplo, el objeto Date tiene dos métodos (por ejemplo, `getFullYear`, `getMonth`), que permiten leer la información del reloj del sistema. El objeto Sound tiene métodos (por ejemplo, `setVolume`, `setPan`) que permiten controlar el sonido en una película. El objeto MovieClip tiene métodos que permiten controlar las instancias de los clips de película (por ejemplo, `play`, `stop` y `getURL`) a la vez que obtienen y definen información acerca de sus propiedades (por ejemplo, `_alpha`, `_framesloaded`, `_visible`).

Los clips de película son los objetos más importantes de las películas de Flash porque contienen líneas de tiempo que se ejecutan independientemente de las demás. Por ejemplo, si la línea de tiempo principal sólo tiene un fotograma y en una de las capas se incluye un clip de película que contiene a su vez10 fotogramas, el clip de película presentará todo su contenido independientemente de que la línea principal sólo se extienda un fotograma. Esto permite que cada instancia de un clip de película se comporte como objeto autónomo con capacidad de comunicación con los demás.

Cada instancia de un clip de película tiene un nombre único, de modo que es posible controlarlas a través de acciones. Por ejemplo, si dispusiésemos de varias instancias de un clip de película en el Escenario (por ejemplo, `leftClip` y `rightClip`) podríamos asignar una acción para lograr que sólo una se reprodujese en un momento dado. Para esto, basta asignar una acción que active a la instancia a través del nombre de la misma. En el siguiente ejemplo, el nombre del clip de película es `leftClip`:

```
leftClip.play();
```

Al momento de reproducción de una película también se pueden duplicar, eliminar y arrastrar instancias de uno o varios clips de película utilizando su nombre de instancia. El siguiente ejemplo duplica la instancia `cartItem` para llenar un carrito de compra con el número de artículos comprados:

```
onClipEvent(load) {
    do {
        duplicateMovieClip("cartItem", "cartItem" + i, i);
        i  = i + 1;
    } while (i <= numberItemsPur);
}
```

Los clips de película poseen propiedades cuyos valores pueden establecerse y recuperarse en forma dinámica mediante ActionScript. La modificación y lectura de estas propiedades puede cambiar el aspecto e identidad de un clip de película y es la clave para darle interactividad. Por ejemplo, el siguiente script utiliza la acción `setProperty` para establecer la transparencia (parámetro alfa) de la instancia `navigationBar` al valor 10:

```
setProperty("navigationBar", _alpha, 10);
```

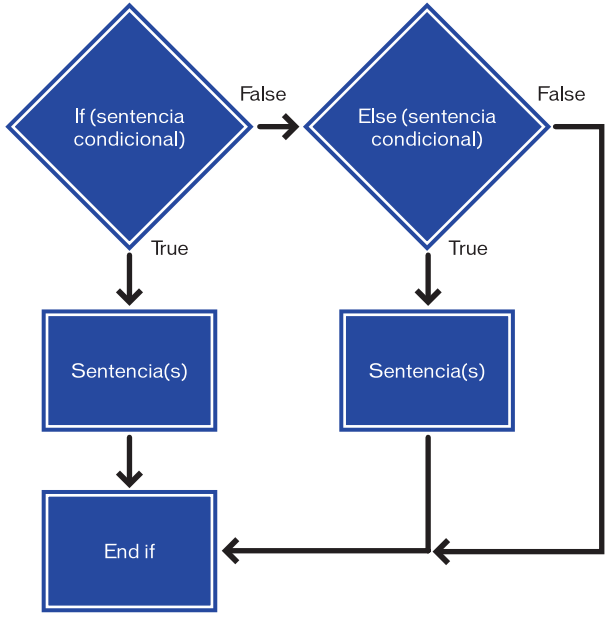
Para obtener más información referente a otro tipos de objetos, consulte la sección “Utilización de objetos predefinidos” a pagina 81.

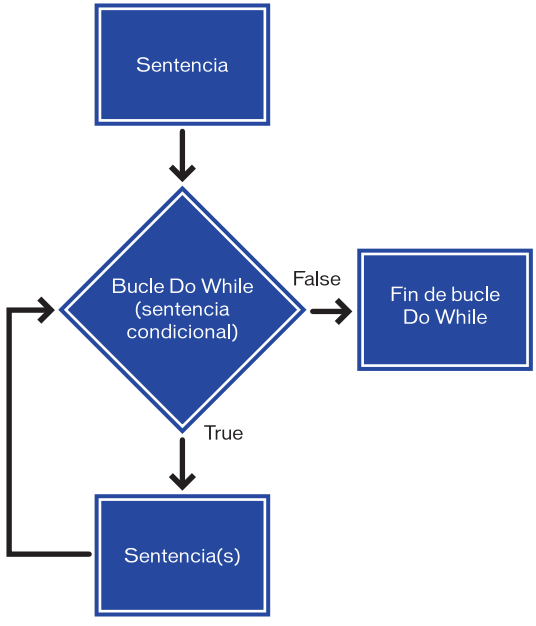
Flujo de los scripts

ActionScript sigue un flujo lógico. Flash ejecuta las sentencias de ActionScript empezando por la primera y continuando secuencialmente hasta llegar a la última o a una sentencia que le indique una bifurcación a otro script u otra parte en el mismo script.

A continuación se presentan algunas acciones que cambian el flujo de ActionScript hacia una sentencia diferente a la que aparece después de ella:

La sentencia `if`, los bucles `do...while`, y la acción `return`.





Una sentencia `if` recibe el nombre de sentencia condicional o "bifurcación lógica", ya que controla el flujo de un script basándose en la evaluación de una determinada condición. Por ejemplo, el siguiente código comprueba si el valor de la variable `number` es menor o igual que 10. Si la comprobación devuelve `true` (por ejemplo, el valor de `number` es 5), se establece la variable `alert` y muestra su valor en un campo de entrada de texto, como el siguiente:

```
if (number <= 10) {
    alert = "The number is less than or equal to 10";
}
```

También se pueden incluir sentencias `else` para crear sentencias condicionales más complejas. En el ejemplo siguiente, si la condición devuelve `true` (por ejemplo, el valor de `number` es 3), se ejecuta la sentencia encerrada entre el primer conjunto de llaves y la variable `alert` se establece en la segunda línea. Si la condición devuelve `false` (por ejemplo, el valor de `number` es 30), se ignora el primer bloque de código y se ejecuta la sentencia encerrada entre llaves detrás de la sentencia `else`, como a continuación:

```
if (number <= 10) {
    alert = "The number is less than or equal to 10";
} else {
    alert = "The number is greater than 10";
}
```

Si desea obtener más información, consulte "Utilización de las sentencias if" a pagina 73.

Los bucles repiten una acción un determinado número de veces, o hasta que se cumpla cierta condición. En el siguiente ejemplo se duplica cinco veces un clip de película:

```
i = 0;
do {
    duplicateMovieClip ("myMovieClip", "newMovieClip" + i, i);
    newName = eval("newMovieClip" + i);
   setProperty(newName, _x, getProperty("myMovieClip", _x) + (i *
5));
    i = i + 1;
} while (i <= 5);
```

Para obtener información detallada, consulte "Repetición de una acción" a pagina 74.

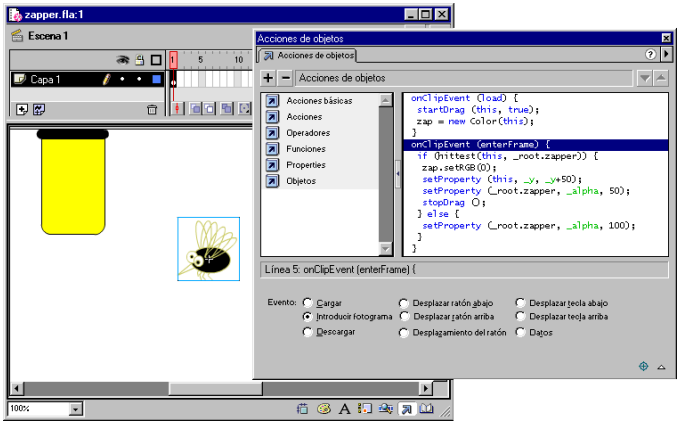
Control de la ejecución de ActionScript

Para escribir scripts se utiliza el panel Acciones. Dicho panel permite asignar el script a un fotograma de la línea de tiempo principal o a la línea de tiempo de cualquier clip de película, o bien a un botón o clip de película en el Escenario.

Flash ejecuta acciones en momentos diferentes, dependiendo del elemento al que estén asociadas:

- Las acciones asociadas con fotogramas se ejecutan cuando la cabeza lectora accede a dicho fotograma.
- Las acciones asociadas con botones se encuentran encerradas en acciones controladoras`on`.
- Las acciones asociadas con clips de película se encuentran encerradas en acciones controladoras `onClipEvent`.

Las acciones `onClipEvent` y `on` reciben el nombre de controladoras porque "controlan" o administran eventos. (Un evento es una incidencia, como un movimiento del ratón, una pulsación de tecla o un clip de película que se carga). Las acciones de botón y de clip de película se ejecutan cuando se produce el evento especificado por el controlador. Es posible asociar más de un controlador a un objeto para que éstos ejecuten acciones como respuesta a diferentes eventos. Para obtener más información, consulte el Capítulo 3, "Interacción con ActionScript".



Varios controladores `onClipEvent` asociados a un clip de película en el Escenario.

Terminología de ActionScript

Al igual que cualquier lenguaje de creación de scripts, ActionScript utiliza terminología específica y está sujeta a determinadas reglas sintácticas. La siguiente lista introduce términos importantes de ActionScript, en orden alfabético. En el Capítulo 2, "Programación de scripts con ActionScript", se estudian con mayor detalle dichos términos y la sintaxis que los gobierna.

Acciones: son sentencias que ordenan a una película efectuar alguna acción durante su reproducción. Por ejemplo, `gotoAndStop` envía la cabeza lectora a un fotograma o etiqueta determinados. En este manual, los términos *acción* y *sentencia* pueden emplearse indistintamente.

Argumentos: también llamados parámetros, actúan como contenedores que permiten pasar valores a las funciones. Por ejemplo, la siguiente función, denominada `welcome`, utiliza dos valores que recibe como argumentos `firstName` y `hobby`:

```
function welcome(firstName, hobby) {
   welcomeText = "Hello, " + firstName + "I see you enjoy " +
hobby;
}
```

Clases: son un tipo de datos que pueden emplearse para definir un nuevo tipo de objeto. Para definir una clase de objeto, debe utilizarse una función constructora.

Constantes: son elementos cuyo valor no cambia. Por ejemplo, la constante `TAB` tiene siempre el mismo significado. Las constantes son útiles para comparar valores.

Constructores: son funciones que se utilizan para definir las propiedades y métodos de una clase. Por ejemplo, el siguiente código crea un nueva clase Circle mediante una función constructora denominada `Circle`:

```
function Circle(x, y, radius){
   this.x = x;
   this.y = y;
   this.radius = radius;
}
```

Tipos de datos: son conjuntos de valores y las operaciones que pueden realizarse sobre ellos, e incluyen cadenas, números, los valores booleanos `true` y `false`, objetos y clips de película. Para conocer más detalles acerca de estos elementos del lenguaje ActionScript, consulte la sección "Tipos de datos" a pagina 56.

Eventos: son acciones que ocurren durante la reproducción de una película. Por ejemplo, cuando se carga un clip de película se generan diferentes eventos: la cabeza lectora accede a un fotograma, el usuario hace clic en un botón o clip de película, o el usuario introduce información mediante el teclado.

Expresiones: son las partes de una sentencia que generan valores. Por ejemplo, `2 + 2` es una expresión.

Funciones: son bloques de código reutilizables, que aceptan argumentos (parámetros) y pueden devolver valores. Por ejemplo, la función `getProperty` acepta como parámetro el nombre de una propiedad y el nombre de instancia de un clip de película y como resultado devuelve el valor de la propiedad. La función `getVersion` devuelve la versión de Flash Player que está reproduciendo la película en un momento dado.

Controladores: son acciones especiales que "controlan" o administran un evento, como `mouseDown` o `load`. Por ejemplo, `on (onMouseEvent)` y `onClipEvent` son controladores de ActionScript.

Identificadores: son nombres que se utilizan para identificar una variable, propiedad, objeto, función o método. El primer carácter debe ser una letra, símbolo de subrayado (_) o símbolo de dólar ($). Los siguientes caracteres pueden ser letras, números, símbolos de subrayado (_) o símbolos de dólar ($). Por ejemplo, `firstName` es el nombre de una variable.

Instancias: son objetos que pertenecen a una determinada clase. Cada instancia de una clase contiene todas las propiedades y métodos de dicha clase. Todos los clips de película son instancias con propiedades (por ejemplo, `_alpha` y `_visible`) y métodos (por ejemplo, `gotoAndPlay` y `getURL`) de la clase MovieClip.

Nombres de instancias: son nombres únicos que permiten controlar instancias de clips de película a través de scripts. Por ejemplo, un símbolo maestro de la Biblioteca podría llamarse `counter` y las dos instancias de dicho símbolo que se utilizan en la película podrían nombrarse instancia `scorePlayer1` y `scorePlayer2`. En el siguiente código se utilizan los nombres de instancia para establecer una variable llamada `score` en cada instancia de un clip de película:

```
_root.scorePlayer1.score += 1
_root.scorePlayer2.score -= 1
```

Palabras clave: son palabras reservadas que tienen un significado especial. Por ejemplo, `var` es una palabra clave que se utiliza para declarar variables locales.

Métodos: son funciones que han sido asignadas a un objeto. Una vez que se ha asignado una función se puede realizar un llamado a la misma como método de ese objeto. Por ejemplo, en el siguiente código, `clear` se convierte en un método del objeto `controller`:

```
function Reset(){
   x_pos = 0;
   x_pos = 0;
}
controller.clear = Reset;
controller.clear();
```

Objetos: son conjuntos de propiedades; cada objeto posee su propio nombre y valor. Los objetos permiten acceder a un determinado tipo de información. Por ejemplo, el objeto predefinido Date ofrece información procedente del reloj del sistema.

Operadores: son términos que calculan un nuevo valor a partir de uno o más valores. Por ejemplo, el operador suma (+) suma dos o más valores y arroja como resultado un nuevo valor.

Rutas de destino: son direcciones jerárquicas de nombres de instancias de clips de película, o bien variables y objetos en una película. Para asignar nombre a una instancia de clip de película se utiliza el panel Instancia. El nombre de la Línea de tiempo principal es siempre `_root`. Se puede usar una ruta de destino para dirigir una acción a un clip de película, o para obtener o establecer el valor de una variable. Por ejemplo, la siguiente sentencia establece una ruta de destino a la variable `volume` dentro del clip de película `stereoControl`:

```
_root.stereoControl.volume
```

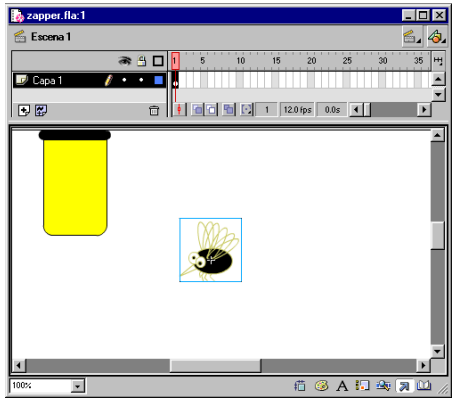
Propiedades: son atributos que definen a un objeto. Por ejemplo, `_visible` es una propiedad de todos los clips de película que define si el clip está visible o si se encuentra oculto.

Variables: son identificadores que almacenan valores de cualquier tipo de datos. Las variables pueden crearse, modificarse y actualizarse. Los valores almacenados en una variable pueden recuperarse para ser utilizados en scripts. En el siguiente ejemplo, los identificadores situados a la izquierda de los signos igual son variables:

```
x = 5;
name = "Lolo";
customer.address = "66 7th Street";
c = new Color(mcinstanceName);
```

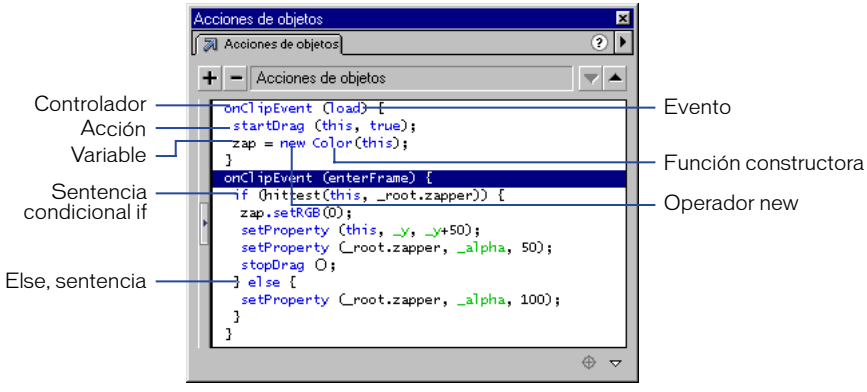
Análisis de un script de ejemplo

En la siguiente película de ejemplo, cuando un usuario arrastra el insecto hacia el matamoscas, el insecto cambia de color al color negro y cae, y el matamoscas parece parpadear. La película tiene una longitud de un fotograma y contiene dos objetos, la instancia del clip de película del insecto y la instancia del clip de película del matamoscas. Cada clip de película contiene asimismo un fotograma.



Las instancias de clip de película del insecto y del matamoscas en el Escenario en el fotograma 1.

Sólo hay un script en la película y está asociado a la instancia del `insecto`, como puede apreciarse en el panel Acciones de objetos que se muestra a continuación:



Panel Acciones de objetos, que muestra el script asignado a la instancia del `insecto`.

Ambos objetos deben ser clips de película, de forma que sea posible asignarles nombres de instancia en el panel Instancia y manipularlos mediante ActionScript. El nombre de instancia del insecto es `bug` y el del matamoscas es `zapper`. En el script, se hace referencia al insecto como `this` porque el script está anexado al insecto y la palabra reservada `this` hace referencia al objeto que lo llama.

Hay dos controladores `onClipEvent` con dos eventos diferentes: `load` y `enterFrame`. Las acciones de la sentencia `onClipEvent(load)` se ejecutan sólo una vez, al cargarse la película. Las acciones de la sentencia `onClipEvent(enterFrame)` se ejecutan cada vez que la cabeza lectora accede a un fotograma. Incluso aunque se trate de películas de un solo fotograma, la cabeza lectora accederá al fotograma reiterativamente y el script se ejecutará de la misma forma. Dentro de cada controlador `onClipEvent` se llevan a cabo las siguientes acciones:

onClipEvent(load) La acción `startDrag` hace posible que se pueda arrastrar el clip de película del insecto. El operador `new` y la función constructora de color, `Color`, crean una instancia del objeto Color y la asignan a la variable `zap`:

```
onClipEvent (load) {
   startDrag (this, true);
   zap = new Color(this);
}
```

onClipEvent(enterFrame) Una sentencia condicional `if` evalúa una acción `hitTest` para comprobar si la instancia del insecto (`this`) está tocando a la instancia del matamoscas (`_root.zapper`). La evaluación puede generar dos posibles resultados, `true` o `false`:

```
onClipEvent (enterFrame) {
   if (hitTest(_target, _root.zapper)) {
      zap.setRGB(0);
      setProperty (_target, _y, _y+50);
      setProperty (_root.zapper, _alpha, 50);
      stopDrag ();
   } else {
      setProperty (_root.zapper, _alpha, 100);
   }
}
```

Si la acción `hitTest` devuelve `true`, el objeto `zap` creado por el evento `load` se utilizará para cambiar el color del insecto a negro. La propiedad *y* (`_y`) del insecto se establece a su valor más `50`, de modo que el insecto caiga. La transparencia del matamoscas (`_alpha`) se establece al valor `50`, de modo que se atenúe. La acción `stopDrag` impide que se arrastre el objeto.

Si la acción `hitTest` devuelve el valor `false`, se ejecutará la acción que sigue a la sentencia `else` y el valor `_alpha` del matamoscas se establecerá al valor `100`. Esto provocará que el matamoscas aparezca intermitentemente, ya que su valor `_alpha` pasará de un estado inicial (100) a un estado activo (50), volviendo posteriormente al estado inicial. La acción `hitTest` arroja como resultado el valor `false` y las sentencias `else` se ejecutan una vez que se ha alcanzado y abatido al insecto.

Para ver cómo se reproduce la película, consulte la *Ayuda de Flash*.

Utilización del panel Acciones

El panel Acciones permite crear y editar acciones de un objeto o fotograma mediante el uso de dos modos de edición. Es posible seleccionar acciones previamente escritas en la lista de la Caja de herramientas, arrastrar y colocar acciones, así como utilizar botones para eliminar o reordenar acciones. El modo Normal permite escribir acciones utilizando campos de parámetros (argumentos) que solicitan los argumentos apropiados. El modo Experto permite escribir y editar acciones directamente en un cuadro de texto, como si se escribiesen en un procesador de texto.

Para mostrar el panel Acciones:

Seleccione Ventana > Acciones.

El panel Acciones se activa cuando se selecciona una instancia de un botón o clip de película. El título del panel Acciones cambia a Acciones de objetos si algún botón o clip de película se encuentra seleccionado, y a Acciones de fotograma si se encuentra seleccionado hay un fotograma.

Para seleccionar un modo de edición:

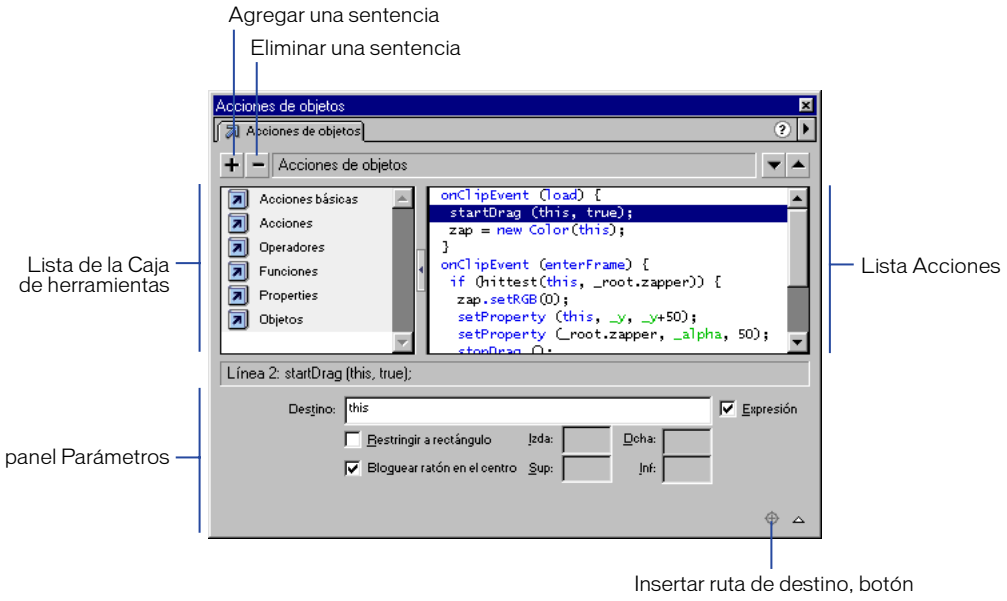
1. Con el panel Acciones en pantalla, haga clic en la flecha de la esquina superior derecha para que se muestre el menú emergente.
2. Seleccione Modo Normal o Modo Experto en el menú emergente.

   Cada script conserva su propio modo. Por ejemplo, si se crea un script para una instancia de un botón en modo Normal y otro en modo Experto, posteriormente se cambiará el modo del panel al seleccionar cada uno de los botones.

Modo Normal

El modo Normal permite crear acciones seleccionándolas de una lista situada en la parte izquierda del panel, denominada lista de la Caja de herramientas. La lista de la Caja de herramientas contiene las siguientes categorías: Acciones básicas, Acciones, Operadores, Funciones, Propiedades y Objetos. La categoría Acciones básicas contiene las acciones de Flash más simples y sólo está disponible en el modo Normal. Las acciones seleccionadas se muestran en la parte derecha del panel, en la lista Acciones. Es posible agregar, eliminar o modificar el orden de las sentencias de acciones; también es posible especificar parámetros (argumentos) para las acciones en los campos que se encuentran en la parte inferior del panel.

En modo Normal se pueden utilizar los controles del panel Acciones para eliminar o cambiar el orden de las sentencias de la lista Acciones. Estos controles resultan especialmente útiles para controlar acciones de botón o fotograma que contienen varias sentencias.



Panel Acciones en modo Normal.

Para seleccionar una acción:

1. Haga clic en una categoría de acciones de la Caja de herramientas para acceder a las acciones de dicha categoría.
2. Haga doble clic en una acción o arrástrela a la ventana de scripts.

Para utilizar los campos de parámetros:

1. Haga clic en el botón Parámetros, situado en la esquina inferior derecha del panel Acciones, para mostrar los campos.
2. Seleccione la acción y especifique nuevos valores en los campos de parámetros, si desea modificar los parámetros de las acciones existentes.

Para insertar una ruta de destino de clip de película:

1. Haga clic en el botón Ruta de destino, situado en la esquina inferior derecha del panel Acciones; accederá al cuadro de diálogo Insertar ruta de destino.
2. Seleccione un clip de película de la lista de visualización.

Para desplazar una sentencia de la lista hacia arriba o hacia abajo:

1. Seleccione una sentencia en la lista Acciones.
2. Haga clic en los botones de flecha arriba o abajo.

Para eliminar una acción:

1. Seleccione una sentencia en la lista Acciones.
2. Haga clic en el botón Eliminar (-).

Para cambiar los parámetros de las acciones existentes:

1. Seleccione una sentencia en la lista Acciones.
2. Introduzca los nuevos valores en los campos de parámetros.

Para cambiar el tamaño de la Caja de herramientas o de la lista Acciones, utilice uno de los siguientes procedimientos:

- Arrastre la barra separadora vertical que aparece entre la Caja de herramientas y la lista Acciones.
- Haga doble clic en la barra separadora para contraer la lista de la Caja de herramientas; vuelva a hacer doble clic en la barra para mostrar la lista de nuevo.
- Haga clic en el botón de flecha izquierda o derecha de la barra separadora para expandir o contraer la lista.

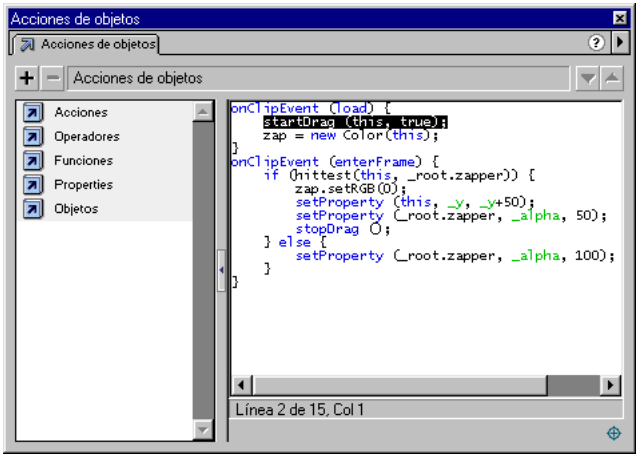
Aunque la Caja de herramientas esté oculta, es posible acceder a sus elementos mediante el botón Agregar (+), que se encuentra en la parte superior izquierda del panel Acciones.

Modo Experto

El modo Experto permite crear acciones introduciendo comandos de ActionScript en el cuadro de texto situado en la parte derecha del panel, o bien seleccionado acciones en la lista de la Caja de herramientas de la izquierda. Para editar acciones, especificar parámetros de acciones o eliminar acciones se utiliza directamente el cuadro de texto, como se haría si se estuviesen creando los scripts mediante un procesador de texto.

El modo Experto permite a los usuarios experimentados de ActionScript modificar sus propios scripts con un procesador de texto, como si se tratase de JavaScript o VBScript. Las principales diferencias respecto al modo Normal son las siguientes:

- Al seleccionar un elemento en el menú emergente Agregar o en la lista de la Caja de herramientas, se introduce dicho elemento en el área de edición de texto.
- No aparecen campos de parámetros.
- En el panel de botones, únicamente se encuentra activo el botón Agregar (+).
- Los botones de flecha arriba y abajo permanecen inactivos.



El panel Acciones en modo Experto.

Cambio del modo de edición

Si se cambia de modo de edición mientras se está creando un script, el formato de éste podría alterarse. Por ese motivo, se recomienda utilizar un solo modo de edición por script.

Al cambiar de modo Normal a Experto se conserva la sangría y el formato. Aunque se pueden convertir scripts creados en modo Normal que contienen errores a modo Experto, no podrán exportarse hasta que éstos hayan sido corregidos.

El cambio de modo Experto a modo Normal es ligeramente más complejo:

- Cuando se cambia de modo Experto a modo Normal, Flash vuelve a asignar formato al script y elimina los espacios en blanco y la sangría que hubiera podido agregarse.
- Si se cambia a modo Normal y, seguidamente, a modo Experto, Flash vuelve a asignar formato al script de acuerdo con el aspecto que tenía en modo Normal.
- Los scripts en modo Experto que contienen errores no pueden exportarse ni convertirse al modo Normal; si se intenta, presentará un mensaje de error.

Para cambiar el modo de edición:

Seleccione Modo Normal o Modo Experto en el menú emergente situado en la parte superior derecha del panel Acciones. Se indicará el modo seleccionado a través de una marca de verificación.

Para establecer las preferencias de un modo de edición:

1. Seleccione Edición > Preferencias.
2. Seleccione la ficha General.
3. En la sección Panel Acciones, elija Modo Normal o Modo Experto en el menú emergente.

Utilización de un editor externo

Aunque el modo Experto del panel Acciones proporciona un mayor control de edición con ActionScript, también puede optar por editar scripts externamente a Flash. Posteriormente puede hacer uso de la acción `include` para agregar los scripts creados mediante el editor externo a un script de Flash.

Por ejemplo, la siguiente sentencia importa un archivo de script:

```
#include "externalfile.as"
```

El texto del archivo de script sustituye a la acción `include`. El archivo de texto deberá estar presente cuando se exporte la película.

Para agregar los scripts creados con un editor externo a un script de Flash:

1. Arrastre la acción `include` desde la lista de la Caja de herramientas a la ventana de scripts.
2. Especifique la ruta del archivo externo en el cuadro Ruta.

   La ruta deberá ser relativa respecto al archivo FLA. Por ejemplo, si los archivos mi_Pelicula.fla y archivo_externo.as estuviesen situados en la misma carpeta, la ruta sería archivo_externo.as. Si archivo_externo.as se encontrase en una subcarpeta denominada scripts, la ruta sería scripts/archivo_externo.as.

Selección de opciones en el panel Acciones

El panel Acciones permite manipular scripts de varias maneras. El tamaño de la fuente puede modificarse en la ventana de scripts. Pueden importarse archivos de texto que contengan comandos de ActionScript en el panel Acciones, así como exportar acciones como archivos de texto, buscar y reemplazar texto en un script y utilizar elementos para realzar la sintaxis para facilitar la identificación de los scripts y la detección de errores. El panel Acciones muestra elementos de realzado de tipo advertencia cuando existen errores sintácticos e incompatibilidades de versiones con Flash Player. También realza elementos de ActionScript *no aprobados* o no deseables.

Estas opciones del panel Acciones se encuentran disponibles tanto en modo Normal como en modo Experto, a menos que se especifique lo contrario.

Para cambiar el tamaño de fuente en la ventana de scripts:

1. En el menú emergente situado a la derecha del panel Acciones, seleccione Tamaño de fuente.
2. Seleccione Pequeño, Normal o Grande.

Para importar un archivo de texto que contenga comandos de ActionScript:

1. En el menú emergente situado en la parte superior derecha del panel Acciones, seleccione Importar desde archivo.
2. Seleccione un archivo de texto que contenga comandos de ActionScript y haga clic en Abrir.

Nota: Los scripts que contienen errores sintácticos sólo pueden importarse en modo Experto. En modo Normal se generará un mensaje de error.

Para exportar acciones como archivos de texto:

1. En el menú emergente situado en la parte superior derecha del panel Acciones seleccione Exportar como archivo.
2. Elija una ubicación para el archivo y haga clic en Guardar.

Para imprimir acciones:

1. En el menú emergente situado en la parte superior derecha del panel Acciones, seleccione Imprimir.

   Se mostrará el cuadro de diálogo Imprimir.
2. Seleccione Opciones y haga clic en Imprimir.

Nota: El archivo impreso no incluirá información respecto al archivo de Flash del que procede. Es recomendable incluir esta información en una acción `comment` en el script.

Para buscar texto en un script, seleccione una opción en el menú emergente del panel Acciones:

- Seleccione Ir a línea para desplazarse a una línea determinada de un script.
- Seleccione Buscar para buscar texto.
- Seleccione Buscar otra vez para buscar una cadena de texto de nuevo.
- Seleccione Reemplazar para buscar y reemplazar texto.

  En modo Experto, el comando Reemplazar busca en todo el cuerpo de texto de un script. En modo Normal, el comando Reemplazar busca y reemplaza texto únicamente en el campo de parámetros de cada acción. Por ejemplo, en modo Normal no sería posible reemplazar todas las acciones `gotoAndPlay` por `gotoAndStop`.

Nota: Utilice los comandos Buscar o Reemplazar para buscar en la lista Acciones actual. Para buscar texto en todos los script de una película, utilice el Explorador de películas. Para obtener más información, consulte la sección *Utilización de Flash*.

Realzado y comprobación de la sintaxis

Los elementos de realzado de la sintaxis permiten identificar determinados elementos de ActionScript y asignarles diferentes colores. De esto modo se evitan errores sintácticos, como la incorrecta colocación de mayúsculas y minúsculas en palabras clave. Por ejemplo, si la palabra clave `typeof` se escribiese como `typeOf`, ésta no aparecería en color azul y esto ayudaría a que se detectase el error. Cuando se encuentra activada la opción de realzado de sintaxis, el texto se destacará del siguiente modo:

- Las palabras clave y los identificadores predefinidos (por ejemplo, `gotoAndStop`, `play` y `stop`) aparecen en color azul.
- Las propiedades aparecen en verde.
- Los comentarios aparecen en color morado.
- Las cadenas encerradas entre comillas aparecen en color gris.

Para activar o desactivar el realzado de sintaxis:

Seleccione Sintaxis en color en el menú emergente situado en la parte superior derecha del panel Acciones. Una marca de verificación indica que la opción está activada. El realzado se aplicará a todos los scripts de la película.

Es recomendable comprobar que la sintaxis de los scripts es correcta antes de exportar películas. Los errores se muestran en la Ventana de salida. Podrán exportarse películas que contengan scripts con errores. No obstante, el usuario recibirá una advertencia indicándole que los scripts con errores no se exportarán.

Para comprobar los errores sintácticos del script actual:

Seleccione Revisar sintaxis en el menú emergente situado en la parte superior derecha del panel Acciones.

Realzado de errores

En modo Normal, todos los errores sintácticos se realzan utilizando un fondo de color rojo sólido en la ventana de scripts. Esto ayuda a detectar problemas. Si se desplaza el puntero de ratón sobre una acción sintácticamente incorrecta, se mostrará el mensaje de error asociado en un cuadro de información. Cuando se selecciona la acción, el mensaje de error se muestra también en el título del panel del área de parámetros.

En modo Normal todas las posibles incompatibilidades de exportación de ActionScript se realzan con un fondo de color amarillo sólido en la ventana de scripts. Por ejemplo, si la versión de exportación de Flash Player se establece como Flash 4, los comandos de ActionScript que sólo sean compatibles con Flash Player 5 se mostrarán realzados en color amarillo. La versión de exportación se determina en el cuadro de diálogo Configuración de publicación.

Todas las acciones no aprobadas se realzan con fondo de color verde en la Caja de herramientas. Las acciones no aprobadas sólo se realzan cuando la versión de exportación de Flash se establece como Flash 5.

Para establecer la versión de exportación de Flash Player:

1. Seleccione Archivo > Configuración de publicación.
2. Haga clic en la ficha Flash.
3. Elija una versión de exportación en el menú emergente Versión.

Nota: El realzado de errores sintácticos no puede desactivarse.

Para realzar la sintaxis no aprobada:

Seleccione Mostrar sintaxis no aprobada en el menú emergente del panel Acciones.

Para desea obtener la relación completa de mensajes de error, consulte el Apéndice C, “Mensajes de error”.

Asignación de acciones a los objetos

Se pueden asignar acciones a botones o a clips de película, de modo que la acción se ejecute cuando el usuario haga clic en un botón o sitúe el puntero sobre él, o bien cuando se cargue el clip de película o se alcance un determinado fotograma. La acción se asigna a una instancia del botón o del clip de película; el resto de instancias del símbolo no se verán afectadas (si desea asignar una acción a un fotograma, consulte la sección "Asignación de acciones a fotogramas" a pagina 48).

Cuando asigna una acción a un botón, debe anidarla dentro de un controlador `on(mouse event)` y especificar los eventos de ratón o teclado que activan la acción. Cuando asigna una acción a un botón en el Modo Normal, el controlador `on(mouse event)` se inserta automáticamente.

Cuando asigna una acción a un clip de película, debe anidarla dentro de un controlador `onClipEvent` y especificar el evento de clip que activa la acción. Cuando asigna una acción a un clip de película en el Modo Normal, el controlador `on(mouse event)` se inserta automáticamente.

Las siguientes instrucciones describen el modo de asignar acciones a objetos mediante el panel Acciones utilizado en modo Normal.

Una vez asignada una acción, utilice el comando Control > Probar película para verificar si funciona. La mayoría de las acciones no funcionan en el modo de edición.

Para asignar una acción a un botón o clip de película:

1. Seleccione un botón o una instancia de clip de película y elija Ventana > Acciones.

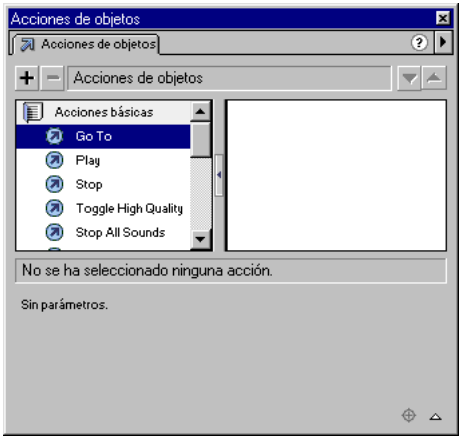
   Si la selección no es un botón, una instancia de clip de película o un fotograma, o si la selección incluye varios objetos, el panel Acciones aparecerá atenuado.

2. Seleccione Modo Normal en el menú emergente situado en la parte superior derecha del panel Acciones de objetos.

3. Para asignar una acción utilice uno de los siguientes procedimientos:

- Haga clic en la carpeta Acciones, que se encuentra en la lista de la Caja de herramientas, a la izquierda del panel Acciones. Haga doble clic en una acción para agregarla a la lista Acciones situada a la derecha del panel.
- Arrastre una acción desde la lista de la Caja de herramientas hasta la lista Acciones.
- Haga clic en el botón Agregar (+) y elija una acción en el menú emergente.

- Utilice el método abreviado de teclado situado junto a cada acción en el menú emergente.



Selección de objetos de la Caja de herramientas en modo Normal

4 Seleccione los parámetros que se requieran para la acción en los campos de parámetros, situados en la parte inferior del panel.

Los parámetros variarán dependiendo de la acción que se seleccione. Para obtener información detallada acerca de los parámetros necesarios para cada acción, consulte el Capítulo 7, "Diccionario de ActionScript". Para insertar una Ruta de destino para un clip de película en un campo de parámetros, haga clic en el botón Ruta de destino, que se encuentra en la esquina inferior derecha del panel Acciones. Para obtener más información, consulte el Capítulo 4, "Utilización de clips de película".

5 Repita los pasos 3 y 4 para asignar más acciones, si fuera necesario.

Para probar la acción de un objeto:

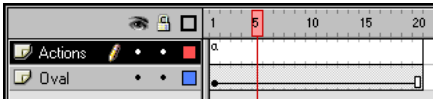
Seleccione Control > Probar película.

Asignación de acciones a fotogramas

Si se desea que una película realice un acción específica cuando alcance un fotograma clave, basta con asignar una acción de fotograma al fotograma clave. Por ejemplo, para crear un bucle entre los fotogramas 20 y 10 de la Línea de tiempo, agregue la siguiente acción de fotograma al fotograma 20:

```
gotoAndPlay (10);
```

Es conveniente situar las acciones de fotograma en capas independientes. Los fotogramas con acciones se indican con una pequeña *a* en la Línea de tiempo.



Una "a" en un fotograma clave indica una acción de fotograma.

Una vez asignada la acción, seleccione Control > Probar película para verificar si funciona. La mayoría de las acciones no funcionan en el modo de edición.

Las siguientes instrucciones describen el modo de asignar acciones de fotograma mediante el panel Acciones en modo Normal (para obtener información acerca de la asignación de acciones a botones o clips de película, consulte la sección "Asignación de una acción o método" a pagina 127).

Para asignar una acción a un fotograma clave:

1. Seleccione un fotograma clave en la Línea de tiempo y elija Ventana > Acciones.

   Si el fotograma seleccionado no es un fotograma clave, la acción se asignará al fotograma clave anterior. Si no se ha seleccionado ningún fotograma, o si la selección incluye varios fotogramas clave, el panel Acciones aparecerá atenuado.

2. Seleccione Modo Normal en el menú emergente situado en la parte superior derecha del panel Acciones de fotograma.

3 Para asignar una acción utilice uno de los siguientes procedimientos:

- Haga clic en la carpeta Acciones, que se encuentra en la lista de la Caja de herramientas, a la izquierda del panel Acciones. Haga doble clic en una acción para agregarla a la lista Acciones situada a la derecha del panel.
- Arrastre una acción desde la lista de la Caja de herramientas hasta la lista Acciones.
- Haga clic en el botón Agregar (+) y elija una acción en el menú emergente.
- Utilice el método abreviado de teclado situado junto a cada acción en el menú emergente.
- Seleccione los parámetros que se requieran para la acción en los campos de parámetros, situados en la parte inferior del panel.

4 Para asignar acciones adicionales, seleccione otro fotograma clave y repita el paso 3.

Para probar una acción de fotograma:

Seleccione Control > Probar película.

CAPÍTULO 2
Programación de Scripts en ActionScript

Es posible crear scripts en ActionScript, optando por el nivel de detalle que se desee utilizar. Para utilizar las acciones sencillas, puede acceder al panel Acciones en Modo Normal y crear scripts eligiendo las opciones que desea utilizar de los menús y de las listas. Sin embargo, si desea utilizar ActionScript para escribir scripts más poderosos, es recomendable entender las bases del lenguaje ActionScript.

Como otros lenguajes de scripts, ActionScript consta de componentes como objetos predefinidos y funciones, y le permite crear sus propios objetos y funciones. ActionScript tiene sus propias reglas sintácticas, se reserva palabras clave, proporciona operadores y le permite utilizar variables para almacenar y recuperar información.

La sintaxis y el estilo de ActionScript se parece mucho al de JavaScript. Flash 5 convierte los scripts de ActionScript escritos en cualquier versión anterior de Flash.

Utilización de la sintaxis de ActionScript

ActionScript sigue reglas gramaticales y de puntuación que determinan qué caracteres y palabras se utilizan para dar significado a una sentencia y el orden en que se deben escribir. Por ejemplo, en español, una oración termina con un punto. En ActionScript, se utiliza un punto y coma para finalizar una sentencia.

A continuación se detallan las reglas generales que se aplican a todo el lenguaje ActionScript. La mayoría de los términos de ActionScript tienen sus propios requisitos; para conocer las reglas de un término específico, consúltelo en el Capítulo 7, "Diccionario de ActionScript".

Sintaxis de punto

En ActionScript, se utiliza un punto (.) para indicar las propiedades o métodos relacionados con un objeto o un clip de película. También se utiliza para identificar la ruta de destino a un clip de película o a una variable. Una expresión de sintaxis de punto comienza con el nombre del objeto o del clip de película seguido por un punto y finaliza con la propiedad, el método o la variable que desee especificar.

Por ejemplo, la propiedad de clip de película `_x` indica la posición del clip de película en el eje *x* en el Escenario. La expresión `ballMC._x` se refiere a la propiedad `_x` de la instancia del clip de película `ballMC`.

Otro ejemplo, `submit` es una variable establecida en el clip de película `form` que se encuentra anidado dentro del clip de película `shoppingCart`. La expresión `shoppingCart.form.submit = true` establece la variable `submit` de la instancia `form` en `true`.

La expresión de un método de un objeto o de un clip de película sigue el mismo esquema. Por ejemplo, el método `play` de la instancia `ballMC` mueve la cabeza lectora en la Línea de tiempo de `ballMC`, como en la siguiente sentencia:

```
ballMC.play();
```

La sintaxis de punto también utiliza dos alias especiales, `_root` y `_parent`. El alias `_root` se refiere a la Línea de tiempo principal. Puede utilizar el alias `_root` para crear una ruta de destino absoluta. Por ejemplo, la siguiente sentencia llama a la función `buildGameBoard` en el clip de película `functions` en la Línea de tiempo principal:

```
_root.functions.buildGameBoard();
```

Puede utilizar el alias `_parent` para referirse a un clip de película en el que está anidado el clip de película actual. Puede utilizar `_parent` para crear una ruta de destino relativa. Por ejemplo, si el clip de película `dog` se encuentra anidado dentro del clip de película `animal`, la siguiente sentencia en la instancia `dog` dice a `animal` que se detenga:

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

Consulte el Capítulo 4, "Trabajo con clips de películas".

Sintaxis de barras

La sintaxis de barra se utilizó en Flash 3 y 4 para indicar la ruta de destino de un clip de película o de una variable. Esta sintaxis aún se admite en Flash Player 5, pero no se recomienda su utilización. En la sintaxis de barras inversas, éstas se utilizan en lugar de los puntos para indicar la ruta a un clip de película o a una variable. Para indicar una variable, debe anteponerle dos puntos como se muestra a continuación:

```
myMovieClip/childMovieClip:myVariable
```

Puede escribir la misma ruta de destino en sintaxis de punto, como se muestra a continuación:

```
myMovieClip.childMovieClip.myVariable
```

La sintaxis de barra inversa se utilizaba comúnmente con la acción `tellTarget`, pero su utilización ya no se recomienda.

Nota: Ahora es recomendable emplear la acción `with` en lugar de `tellTarget` debido a que es más compatible con la sintaxis de punto. Para obtener más información, consulte el Capítulo 7, "Diccionario de ActionScript".

Llaves

Las sentencias de ActionScript se agrupan en bloques con llaves `({ })`, como se muestra en el script siguiente:

```
on(release) {
    myDate = new Date();
    currentMonth = myDate.getMonth();
}
```

Consulte "Uso de acciones" a pagina 71.

Puntos y coma

Una sentencia de ActionScript se termina con punto y coma, pero si omite el punto y coma final, Flash compilará con éxito script. Por ejemplo, las siguientes sentencias finalizan con punto y coma:

```
column = passedDate.getDay();
row    = 0;
```

Las mismas sentencias podrían escribirse sin terminar en punto y coma:

```
column = passedDate.getDay()
row    = 0
```

Paréntesis

Cuando defina una función, coloque los argumentos entre paréntesis:

```
function myFunction (name, age, reader){
   ...
}
```

Cuando llame a una función, incluya cualquiera de los argumentos que se desean pasar a la misma entre paréntesis, como se muestra a continuación:

```
myFunction ("Steve", 10, true);
```

También puede utilizar paréntesis para cancelar el orden de precedencia de ActionScript o para hacer más legibles las sentencias de ActionScript. Consulte "Precedencia de operadores" a pagina 65.

También puede utilizar los paréntesis para evaluar una expresión a la izquierda de un punto en la sintaxis de punto. Por ejemplo, en el siguiente enunciado, los paréntesis hacen que `new color(this)` evalúe y cree un nuevo objeto Color:

```
onClipEvent(enterFrame) {
   (new Color(this)).setRGB(0xffffff);
}
```

Si no ha utilizado paréntesis, tendrá que agregar una sentencia al código para evaluarlo:

```
onClipEvent(enterFrame) {
   myColor = new Color(this);
   myColor.setRGB(0xffffff);
}
```

Letras en mayúsculas y minúsculas

ActionScript solamente distingue el uso de mayúsculas y minúsculas en sus palabras clave; con el resto de ActionScript puede utilizar mayúsculas y minúsculas según desee. Por ejemplo, las siguientes sentencias son equivalentes:

```
cat.hilite = true;
CAT.hilite = true;
```

Sin embargo, es conveniente seguir convenciones coherentes en el uso de mayúsculas y minúsculas, como las que se utilizan en este manual, para que sea más fácil identificar los nombres de las funciones y de las variables cuando se lea el código de ActionScript.

Si no utiliza el formato correcto de mayúsculas y minúsculas con las palabras clave, puede que su script tenga errores. Cuando está activada la Sintaxis en color en el panel de Acciones, las palabras clave escritas con el formato correcto de mayúsculas y minúsculas aparecen en azul. Si desea obtener más información, consulte las secciones "Palabras clave" a pagina 55 y "Realzado y comprobación de la sintaxis" a pagina 44.

Comentarios

En el panel Acciones utilice la sentencia `comment` para agregar notas a una acción de botón o un fotograma que le faciliten el seguimiento de lo que desea que haga una acción. Los comentarios también son útiles para pasar información a otros desarrolladores cuando se trabaje en un entorno de colaboración o se proporcionen ejemplos.

Cuando elija la acción `comment`, los caracteres `//` se insertan en el script. Incluso un script sencillo es más fácil de entender si documenta sus notas a medida que lo crea:

```
on(release) {
    // create new Date object
    myDate = new Date();
    currentMonth = myDate.getMonth();
    // convert month number to month name
    monthName = calcMonth(currentMonth);
    year = myDate.getFullYear();
    currentDate = myDate.getDat ();
}
```

Los comentarios aparecen en rosa color en la ventana Script. Pueden ser de cualquier longitud sin afectar al tamaño del archivo exportado, y no necesitan seguir las reglas de la sintaxis o palabras clave de ActionScript.

Palabras clave

ActionScript se reserva palabras para su uso específico en el lenguaje, de modo que no se pueden utilizar como nombres de variables, de funciones ni de etiquetas. En la siguiente tabla se muestra una lista de las palabras clave de ActionScript:

`break`	`for`	`new`	`var`
`continue`	`function`	`return`	`void`
`delete`	`if`	`this`	`while`
`else`	`in`	`typeof`	`with`

Si desea obtener más información acerca de una palabra clave concreta, consulte el Capítulo 7, “Diccionario de ActionScript”.

Constantes

Una constante es una propiedad cuyo valor nunca cambia. En la caja de herramientas Acciones y en el Capítulo 7, “Diccionario de ActionScript”, se muestra una lista de las constantes, todas en mayúsculas.

Por ejemplo, las constantes `BACKSPACE`, `ENTER`, `QUOTE`, `RETURN`, `SPACE`, y `TAB` son propiedades del objeto Key y se refieren a las teclas del teclado. Para comprobar si el usuario está presionando la tecla Intro, utilice la siguiente sentencia:

```
if(keycode() == Key.ENTER) {
   alert = "Are you ready to play?"
   controlMC.gotoAndStop(5);
}
```

Tipos de datos

Un tipo de datos describe la clase de información que puede contener una variable o el elemento de ActionScript. Existen dos clases de tipo de datos: primitivos y de referencia. Los datos de tipo primitivo (cadena, número y Booleano) tienen un valor constante y por lo tanto pueden contener el valor real del elemento que representan. El tipo de datos de referencia (clip de película y objeto) tienen valores que pueden cambiar y por lo tanto contienen referencias al valor real del elemento. Las variables que contienen datos de tipo primitivo se comportan de modo diferente en ciertas situaciones que las que contienen datos de tipo referencia. Consulte “Utilización de las variables en un script” a pagina 62.

Cada tipo de datos tiene sus propias reglas y aparece es esta lista. Se incluyen referencias del tipo de datos que se explican con más detalle.

Cadena

Una cadena es una secuencia de caracteres tales como letras, números y signos de puntuación. Las cadenas se introducen en una sentencia de ActionScript incluyéndolas entre comillas simples o dobles. A las cadenas se les trata como caracteres en lugar de como variables. Por ejemplo, en la siguiente sentencia, **"L7"** es una cadena:

```
favoriteBand = "L7";
```

Puede utilizar el operador de suma (`+`) para *concatenar*, o unir, dos cadenas. ActionScript trata los espacios del comienzo o del final de una cadena como parte literal de la cadena. La siguiente expresión incluye un espacio después de la coma:

```
greeting = "Welcome, " + firstName;
```

Aunque ActionScript no distingue entre mayúsculas y minúsculas en las referencias a variables, los nombres de instancia y las etiquetas de fotogramas, en las cadenas literales sí las distingue. Por ejemplo, las dos siguientes sentencias colocan texto diferente en las variables de campo de texto especificadas, ya que **"Hello"** y **"HELLO"** son cadenas literales.

```
invoice.display = "Hello";
invoice.display = "HELLO";
```

Para incluir un signo de interrogación en una cadena, ponga delante el carácter de barra inversa (\). A esto se le llama "escape" de un carácter. Hay otros caracteres que no pueden representarse en ActionScript excepto por secuencias de escape especiales. La siguiente tabla muestra todos los caracteres de escape de ActionScript:

Secuencia de escape	Carácter
`\b`	Carácter de retroceso (ASCII 8)
`\f`	Carácter de salto de página (ASCII 12)
`\n`	Carácter de avance de línea (ASCII 10)
`\r`	Carácter de retorno de carro (ASCII 13)
`\t`	Carácter de tabulación (ASCII 9)
`\"`	Comillas dobles
`\'`	Comillas simples
`\\`	Barra inversa
`\000 - \377`	Un byte especificado en octal
`\x00 - \xFF`	Un byte especificado en hexadecimal
`\u0000 - \uFFFF`	Un carácter Unicode de 16 bits especificado en hexadecimal

Numérico

El tipo de datos numérico es un número en coma flotante de doble precisión. Puede manipular los números utilizando los operadores aritméticos de suma (+), resta (-), multiplicación, (*), división (/), módulo (%), incremento (++) y decremento (--). También puede utilizar los métodos del objeto predefinido Math para manipular los números. El siguiente ejemplo utiliza el método `sqrt` (raíz cuadrada) para devolver la raíz cuadrada del número 100:

```
Math.sqrt(100);
```

Consulte "Operadores numéricos" a pagina 66.

Booleano

Un valor Booleano es uno que es `true` o `false`. ActionScript también convierte los valores `true` y `false` en 1 y 0 cuando sea adecuado. Los valores booleanos se usan con mayor frecuencia con los operadores lógicos en sentencias de ActionScript que realizan comparaciones para controlar el flujo de un script. Por ejemplo, en el siguiente script, la película se reproduce si la variable `password` es `true`:

```
onClipEvent(enterFrame) {
    if ((userName == true) && (password == true)){
        play();
    }
}
```

Consulte “Utilización de las sentencias if” a pagina 73 y “Operadores lógicos” a pagina 67.

Objeto

Un objeto es un conjunto de propiedades. Cada propiedad tiene un nombre y un valor. El valor de la propiedad puede ser cualquier tipo de datos de Flash, incluso el tipo de datos de objeto. Esto le permite organizar los objetos unos dentro de otros, o “anidarlos”. Para especificar objetos y sus propiedades, debe utilizar el operador punto (.). Por ejemplo, en el siguiente código, `hoursWorked` es una propiedad de `weeklyStats`, que a su vez es una propiedad de `employee`:

```
employee.weeklyStats.hoursWorked
```

Puede utilizar los objetos predefinidos de ActionScript para acceder y manipular tipos de información específicos. Por ejemplo, el objeto Math tiene métodos que realizan operaciones matemáticas con los números que le pasan. Este ejemplo utiliza el método `sqrt`:

```
squareRoot = Math.sqrt(100);
```

El objeto MovieClip de ActionScript tiene métodos que le permiten controlar las instancias del símbolo de clip de película en el Escenario. Este ejemplo, utiliza los métodos `play` y `nextFrame`:

```
mcInstanceName.play();
mc2InstanceName.nextFrame();
```

También puede crear sus propios objetos de modo que pueda organizar la información de su película. Para agregar interactividad a una película con ActionScript, necesitará varios elementos de información diferentes: por ejemplo, puede que necesite un nombre de usuario, la velocidad de una pelota, los nombres de los elementos en un carrito de la compra, el número de fotogramas cargados, el código postal del usuario y qué tecla se presionó en último lugar. La creación de objetos personalizados le permite organizar esta información en grupos, simplificando la creación de scripts y su reutilización. Si desea obtener más información, consulte “Utilización de objetos personalizados” a pagina 85.

Clip de película

Los clips de película son símbolos que pueden contener y ejecutar una animación en una película de Flash. Son el único tipo de datos que hacen referencia a elementos gráficos. El tipo de datos de clip de película le permite controlar los símbolos del clip de películas utilizando los métodos del objeto MovieClip. Puede llamar a los métodos utilizando el operador punto (.) como se muestra a continuación:

```
myClip.startDrag(true);
parentClip.childClip.getURL( "http://www.macromedia.com/support/"
+ product);
```

Acerca de las variables

Una variable es un contenedor que almacena información. El contenedor en sí es siempre el mismo, pero el contenido puede cambiar. La modificación del valor de una variable a medida que se reproduce la película permite registrar y guardar información sobre las acciones del usuario, registrar valores que se modifican conforme se reproduce la película o evaluar si una determinada condición es verdadera o falsa.

Es bastante útil asignar a una variable un valor conocido la primera vez la defina. Esto se llama inicializar una variable y a menudo se hace en el primer fotograma de una película. Inicializar las variables hace que sea más sencillo realizar el seguimiento y comparar el valor de la variable a medida que se reproduce la película.

Las variables pueden contener cualquier tipo de datos: número, cadena, Booleano, objeto o clip de película. El tipo de datos que contiene una variable afecta al modo en el que cambia el valor de la variable cuando se le asigna en un script.

El tipo de información normal que suele guardarse en una variable incluye una URL, un nombre de usuario, el resultado de una operación matemática, el número de veces que ocurre un evento o si se ha hecho clic en un botón. Cada película e instancia de clip de película tiene su propio conjunto de variables, cada una de ellas con su propio valor independiente de otras variables definidas en otras películas o clips de película.

Asignación de nombre para una variable

El nombre de una variable debe seguir estas reglas:

- Debe ser un identificador.
- No puede ser una palabra clave o un literal Booleano (`true` o `false`).
- Debe ser exclusivo dentro de su ámbito. (Consulte “Ámbito de una variable” a pagina 61).

Determinación del tipo de variable

En Flash, no tiene que definir explícitamente una variable de forma que contenga un número, una cadena u otro tipo de datos. Flash determina el tipo de datos de una variable cuando se asigna un valor a la misma:

```
x = 3;
```

En la expresión `x = 3` Flash evalúa el elemento de la parte derecha del operador y determina que es del tipo número. Una asignación posterior puede cambiar el tipo de `x`; por ejemplo, `x = "hello"` cambia el tipo de `x` a una cadena. Una variable a la que no se ha asignado un valor tiene el tipo de `undefined`.

ActionScript convierte los tipos de datos automáticamente cuando lo requiere una expresión. Por ejemplo, cuando pasa un valor a la acción `trace`, `trace` automáticamente convierte el valor en una cadena y la envía a la ventana Salida. En las expresiones con operadores, ActionScript convierte los tipos de datos según sea necesario; por ejemplo, cuando se utiliza con una cadena, el operador + espera que el otro operando sea una cadena:

```
"Next in line, number " + 7
```

ActionScript convierte el número 7 en la cadena `"7"` y la agrega al final de la primera cadena, lo que da como resultado la siguiente cadena:

```
"Next in line, number 7"
```

Cuando se depuran scripts, con frecuencia es muy útil determinar el tipo de datos de una expresión o variable para entender por qué se comporta de cierta manera. Puede realizar esto por medio del operador `typeof` como en el siguiente ejemplo:

```
trace(typeof(variableName));
```

Para convertir una cadena en un valor numérico, utilice la función `Number`. Para convertir un valor numérico en una cadena, utilice la función `String`. Consulte el Capítulo 7, “Diccionario de ActionScript” a pagina 175.

Ámbito de una variable

El "ámbito" de una variable se refiere al área en la que se conoce y se puede hacer referencia a ella. Las variables en ActionScript pueden ser globales o locales. Una variable global se comparte por todas las Líneas de tiempo, mientras que una variable local sólo está disponible dentro de su propio bloque de código (entre corchetes).

Puede utilizar la sentencia `var` para declarar una variable local dentro de un script. Por ejemplo, las variables `i` y `j` con frecuencia se utilizan como contadores de bucles. En el siguiente ejemplo, `i` se utiliza como la variable local, solamente existe dentro de la función `makeDays`:

```
function makeDays(){
   var i
   for( i = 0; i < monthArray[month]; i++ ) {

      _root.Days.attachMovie( "DayDisplay", i, i + 2000 );

      _root.Days[i].num = i + 1;
      _root.Days[i]._x = column * _root.Days[i]._width;
      _root.Days[i]._y = row * _root.Days[i]._height;

      column = column + 1;

      if (column == 7 ) {

         column = 0;
         row = row + 1;
      }
   }
}
```

Las variables locales también pueden evitar los conflictos ocasionados por la utilización de los mismos nombres en diferentes variables, que pueden originar errores en su película. Por ejemplo, si utiliza `name` como una variable local, podría utilizarla para almacenar un nombre de usuario en un contexto y el nombre de una instancia de un clip de película en otro; como estas variables se ejecutarán en ámbitos independientes, no se presentaría conflicto alguno.

Es usual y recomendable utilizar variables locales en el contexto de una función de modo que ésta pueda actuar como un segmento de código independiente. Una variable local solamente puede cambiar dentro de su propio bloque de código. Si la expresión contenida en una función utiliza una variable global, algo externo a la función podría modificar su valor, lo cual cambiaría la función.

Declaración de variables

Para declarar variables globales, utilice la acción `setVariables` o el operador de asignación (=). Ambos métodos consiguen el mismo resultado.

Para declarar una variable local, utilice la sentencia `var` dentro de una función. El ámbito de las variables locales es el contexto del bloque de código y expiran al final del mismo. Las variables locales que no han sido declaradas dentro de un bloque de código expiran cuando finaliza el script en que se utilizan.

Nota: La acción `call` también crea el ámbito de una nueva variable local y que está constituido por el script al que llama. Cuando se sale del script que llamó, desaparece el ámbito de esta variable local. Sin embargo, esto no es recomendable ya que la acción `call` ha sido sustituida por la acción `with` que es más compatible con la sintaxis de punto.

Para comprobar el valor de una variable, utilice la acción `trace` para enviar el valor a la ventana Salida. Por ejemplo, `trace(hoursWorked)` envía el valor de la variable `hoursWorked` a la ventana Salida en el modo probar película. También puede comprobar y establecer los valores de las variables en el Depurador en el modo probar película. Para obtener más información, consulte el Capítulo 6, "Resolución de problemas de ActionScript."

Utilización de las variables en un script

Debe declarar una variable en un script antes de poder utilizarla en una expresión. Si utiliza una variable que no ha sido declarada, como en el siguiente ejemplo, el valor de la variable estará `undefined` y su script generará un error:

```
getURL(myWebSite);
myWebSite = "http://www.shrimpmeat.net";
```

La sentencia que declara la variable `myWebSite` debe ir al principio de modo que la variable de la acción `getURL` pueda ser sustituida por un valor.

El valor de una variable puede cambiar muchas veces en un script. El tipo de datos que contiene la variable afecta a cómo y cuándo cambia la misma. Las variables que contienen datos de tipo primitivo, como son las cadenas y números, se pasan basados en su valor. Esto quiere decir se pasa a la variable el contenido real de la misma.

En el siguiente ejemplo, `x` está establecida en 15 y ese valor se copia en `y`. Cuando `x` se cambia a 30, el valor de `y` sigue siendo 15 ya que `y` no busca en `x` su valor, sino que contiene el valor de `x` que se le pasó.

```
var x = 15;
var y = x;
var x = 30;
```

Otro ejemplo, es el caso en que la variable `in` contiene un valor primitivo de 9, de modo que el valor real se pasa a la función `sqrt` y el valor que devuelve es 3:

```
function sqrt(x){
  return x * x;
}

var in = 9;
var out = sqr(in);
```

El valor de la variable `in` no cambia.

Los datos de tipo objeto pueden contener una cantidad de información tan compleja y grande que una variable cuyo contenido es este tipo de dato no contiene el valor real, contiene una referencia al valor. Esta referencia es como un alias que apunta al contenido de la variable. Cuando la variable necesita conocer su valor, la referencia solicita el contenido y devuelve la respuesta sin transferir el valor a la variable.

A continuación se muestra un ejemplo en el que se pasa un valor por referencia:

```
var myArray = ["tom", "dick"];
var newArray = myArray;
myArray[1] = "jack";
trace(newArray);
```

El código anterior crea un objeto Array llamado `myArray` que tiene dos elementos. Se crea la variable `newArray` y se pasa una referencia a `myArray`. Cuando el segundo elemento de `myArray` cambia, afectará a cualquier variable que haga referencia al mismo. La acción `trace` enviaría `["tom", "jack"]` a la ventana Salida.

En el siguiente ejemplo, `myArray` contiene un objeto Array, de modo que se pasa a la función `zeroArray` por referencia. La función `zeroArray` cambia el contenido de la matriz en `myArray`.

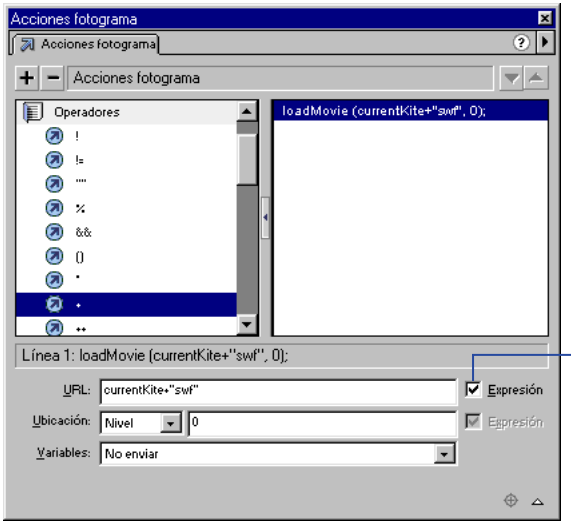
```
function zeroArray (array){
  var i;
  for (i=0; i < array.length; i++) {
    array[i] = 0;
  }
}

var myArray = new Array();
myArray[0] = 1;
myArray[1] = 2;
myArray[2] = 3;

var out = zeroArray(myArray)
```

La función `zeroArray` acepta como argumento un objeto Array y establece todos los elementos del mismo en 0. Puede modificar la matriz ya que sus valores se pasan a través de una referencia.

Las referencias a todos los objetos excepto a los clips de película reciben el nombre de *referencias fijas* debido a que no se puede borrar el objeto al que se hace referencia. Una referencia a un clip de película es un tipo especial de referencia llamada una *referencia variable*. Las referencias variables no obligan a que exista el objeto al que hacen referencia. Si se destruye un clip de película con una acción como `removeMovieClip`, cualquier referencia a él ya no funcionará.

Utilización de operadores para manipular los valores de las expresiones

Una expresión es cualquier sentencia que Flash puede evaluar y que devuelve un valor. Puede crear una expresión combinando operadores y valores, o bien llamando a una función. Cuando escribe una expresión en el panel Acciones en Modo Normal, asegúrese de que el cuadro Expresión se encuentra marcado en el panel Parámetros, en caso contrario el campo contendrá el valor literal de una cadena.



Los operadores son caracteres que especifican cómo combinar, comparar o modificar los valores de una expresión. Los elementos sobre los que el operador actúa se llaman *operandos*. Por ejemplo, en la siguiente sentencia, el operador `+` agrega el valor de un literal numérico al valor de la variable `foo`; `foo` y `3` son los operandos:

```
foo + 3
```

En esta sección se describen las reglas generales sobre los tipos de operadores más comunes. Para obtener información detallada sobre cada operador que se menciona aquí, y sobre los operadores especiales que no se cubren en esta sección, consulte el Capítulo 7, "Diccionario de ActionScript".

Precedencia de operadores

Cuando se utilizan dos o más operadores en la misma sentencia, algunos operadores toman precedencia sobre otros. ActionScript sigue una jerarquía muy estricta para determinar qué operadores deben ejecutarse en primer lugar. Por ejemplo, la multiplicación siempre se realiza antes que la suma; sin embargo, los elementos entre paréntesis tienen precedencia sobre la multiplicación. De modo que, sin paréntesis, ActionScript realiza primero la multiplicación en el ejemplo siguiente:

```
total = 2 + 4 * 3;
```

El resultado es 14.

Pero cuando la operación de suma está entre paréntesis, ActionScript realiza la suma en primer lugar:

```
total = (2 + 4) * 3;
```

El resultado es 18.

Si desea ver una tabla con todos los operadores y su precedencia, consulte el Apéndice B, "Precedencia de operadores y su asociatividad".

Asociatividad de operadores

Cuando dos o más operadores comparten la misma precedencia, su asociatividad determina el orden en el que se llevan a cabo. La asociatividad puede ser de izquierda a derecha o de derecha a izquierda. Por ejemplo, el operador de multiplicación tiene una asociatividad de izquierda a derecha, las dos sentencias siguientes son equivalentes:

```
total = 2 * 3 * 4;
total = (2 * 3) * 4;
```

Si desea ver una tabla con todos los operadores y su asociatividad, consulte el Apéndice B, "Precedencia de operadores y su asociatividad".

Operadores numéricos

Los operadores numéricos realizan sumas, restas, multiplicaciones, divisiones y otras operaciones aritméticas. Los paréntesis y el signo menos son operadores aritméticos. En la tabla siguiente se muestran los operadores numéricos de ActionScript:

Operador	Operación realizada
+	Suma
*	Multiplicación
/	División
%	Módulo
-	Resta
++	Incremento
--	Decremento

Operadores de comparación

Los operadores de comparación comparan los valores de las expresiones y devuelven un valor Booleano (`true` o `false`). Estos operadores se utilizan con mayor frecuencia en los bucles y en las sentencias condicionales En el ejemplo siguiente, si la variable `score` es `100`, se carga cierta película; en caso contrario se carga una película diferente:

```
if (score == 100){
   loadMovie("winner.swf", 5);
} else {
   loadMovie("loser.swf", 5);
   }
```

En la tabla siguiente se muestran los operadores de comparación de ActionScript:

Operador	Operación realizada
<	Menor que
>	Mayor que
<=	Menor que o igual
>=	Mayor que o igual

Operadores de cadena

El operador + tiene un efecto especial cuando se aplica sobre cadenas: concatena los operandos de las dos cadenas. Por ejemplo, las sentencia siguiente agrega:

```
"Congratulations, " a "Donna!":
"Congratulations, " + "Donna!"
```

El resultado es `"Congratulations, Donna!"` Si solamente uno de los operandos del operador + es una cadena, Flash convierte el otro operando en una cadena.

Los operadores de comparación, `>`, `>=`, `<`, y `<=` también tienen un efecto especial cuando actúan sobre cadenas. Estos operadores comparan dos cadenas para determinar cuál es la primera de acuerdo a su orden alfabético. Los operadores de comparación solamente comparan cadenas si ambos operandos son cadenas. Si solamente uno de los operandos es una cadena, ActionScript convierte los números y realiza una comparación numérica.

Nota: Los tipos de datos que maneja ActionScript de Flash 5 permite que se utilicen los mismos operadores en diferentes tipos de datos. Ya no es necesario utilizar los operadores de cadenas de Flash 4 (por ejemplo, `eq`, `ge`, y `lt`) a no ser que se planee exportar los scripts como una película de Flash 4.

Operadores lógicos

Los operadores lógicos comparan los valores Booleanos (`true` y `false`) y devuelven un tercer valor Booleano. Por ejemplo, si ambos operandos evalúan como `true`, el operador lógico AND (`&&`) devuelve `true`. Si uno o ambos operandos evalúan como `true`, el operador lógico OR (`||`) devuelve `false`. Los operadores lógicos se utilizan con frecuencia junto con los operadores de comparación para determinar la condición de una acción `if`. Por ejemplo, en el siguiente script, si ambas expresiones son verdaderas, la acción `if` se ejecutará:

```
if ((i > 10) && (_framesloaded > 50)){
   play()
}
```

En la siguiente tabla se muestra una lista de los operadores lógicos de ActionScript:

Operador	Operación realizada
&&	AND lógico
||	OR lógico
!	NOT lógico

Operadores como bit

Los operadores como bit manipulan internamente los números en coma flotante para convertirlos en números enteros de 32 bits, con los que es más sencillo trabajar. La operación precisa como bit que se realizará depende del operador, pero todas las operaciones como bit evalúan cada dígito de un número en coma flotante para calcular un nuevo valor.

En la siguiente tabla se muestran los operadores como bit de ActionScript:

Operador	Operación realizada
&	And como bit
|	Or como bit
^	Xor como bit
~	Not como bit
«	Desplazamiento a la izquierda
»	Desplazamiento a la derecha
»»	Desplazamiento a la derecha rellenando con ceros

Operadores de igualdad y de asignación

Puede utilizar el operador de igualdad (==) para determinar si los valores o las identidades de dos operandos son iguales. Esta comparación devuelve un valor Booleano (`true` o `false`). Si los operandos son cadenas, números o valores Booleanos, se comparan en base a su valor. Si los operandos son objetos o matrices, se comparan por referencia.

Puede utilizar el operador de asignación (=) para asignar un valor a una variable, como se muestra a continuación:

contraseña = “Sk8tEr”;

También puede utilizar el operador de asignación para asignar valores a múltiples variables en la misma expresión. En la siguiente sentencia, el valor de b se asigna a las variables c y d:

```
a = b = c = d;
```

También puede utilizar operadores de asignación compuestos para combinar operaciones. Los operadores compuestos actúan sobre los dos operandos y después asignan un nuevo valor al primer operando. Por ejemplo, las dos siguientes sentencias son equivalentes:

```
x += 15;
x = x + 15;
```

En la siguiente tabla se muestra una lista de los operadores de igualdad y de asignación de ActionScript:

Operador	Operación realizada
==	Igualdad
!=	No igualdad
=	Asignación
+=	Suma y asignación
-=	Resta y asignación
*=	Multiplicación y asignación
%=	Módulo y asignación
/=	División y asignación
<<=	Desplazamiento a la izquierda como bit y asignación
>>=	Desplazamiento a la derecha como bit y asignación
>>>=	Desplazamiento a la derecha rellenando con ceros y asignación
^=	Xor como bit y asignación
|=	Or como bit y asignación
&=	And como bit y asignación

Operadores punto y de acceso a una matriz

Puede utilizar el operador de punto (`.`) y el operador de acceso a una matriz (`[]`) para acceder a cualquiera de las propiedades del objeto predeterminadas o personalizadas de ActionScript, incluyendo las de un clip de película.

El operador de punto utiliza el nombre de un objeto a su lado izquierdo y el nombre de una propiedad o variable a su lado derecho. El nombre de la propiedad o la variable no puede ser una cadena o una variable que evalúe en una cadena, debe ser un identificador. A continuación se muestran ejemplos que utilizan el operador de punto:

```
year.month = "June";
year.month.day = 9;
```

El operador de punto y el operador de acceso a una matriz se comportan de la misma manera, pero el operador de punto toma un identificador como su propiedad y el operador de acceso a una matriz evalúa su contenido respecto a un nombre y después accede al valor de esa propiedad con nombre. Por ejemplo, las siguientes dos líneas de código acceden a la misma variable `velocity` en el clip de película `rocket`:

```
rocket.velocity;
rocket["velocity"];
```

Puede utilizar el operador de acceso a una matriz para establecer y recuperar dinámicamente nombres de instancias y variables. Por ejemplo, en el código que se muestra a continuación, la expresión dentro del operador `[]` se evalúa y el resultado de la evaluación se utiliza como nombre de la variable que se va a recuperar del clip de película `name`:

```
name["mc" + i ]
```

Si está familiarizado con la sintaxis de diagonal de ActionScript de Flash 4, puede que haya hecho lo mismo utilizando la función `eval`, como se muestra a continuación:

```
eval("mc" & i);
```

El operador de acceso a una matriz también puede utilizarse al lado izquierdo de una sentencia de asignación. Esto le permite establecer dinámicamente los nombres de sentencia, de variable y de objeto, como se muestra en el ejemplo siguiente:

```
name[index] = "Gary";
```

De nuevo, esto es equivalente a la sintaxis de diagonal de ActionScript de Flash 4 que se muestra a continuación:

```
Set Variable: "name:" & index = "Gary"
```

El operador de acceso a una matriz también puede anidarse en sí mismo para simular matrices multidimensionales.

```
chessboard[row][column]
```

Esto es equivalente a la sintaxis de barra que se muestra a continuación:

```
eval("chessboard/" & row & ":" & column)
```

Nota: Si desea escribir un ActionScript que sea compatible con Flash Player 4, puede utilizar la acción `eval` con el operador `add`.

Uso de acciones

Las acciones son sentencias o comandos de ActionScript. Si se asignan varias acciones al mismo fotograma u objeto se crea un script. Las acciones pueden actuar independientemente unas de otras, como se muestra en las siguientes sentencias:

```
swapDepths("mc1", "mc2");
gotoAndPlay(15);
```

También puede anidar acciones utilizando una acción dentro de otra, esto permite que las acciones afecten unas a las otras. En el siguiente ejemplo, la acción `if` dice a la acción `gotoAndPlay` cuándo debe ejecutarse:

```
if (i >= 25) {
   gotoAndPlay(10);
}
```

Las acciones pueden mover la cabeza lectora en la Línea de tiempo (`gotoAndPlay`), controlar el flujo de un script creando bucles (`do while`) o lógica condicional (`if`), o crear nuevas funciones y variables (`function`, `setVariable`). En la siguiente tabla se muestra una lista de las acciones de ActionScript:

Acciones				
break	evaluate	include	print	stopDrag
call	for	loadMovie	printAsBitmap	swapDepths
comment	for...in	loadVariables	removeMovieClip	tellTarget
continue	fsCommand	nextFrame nextScene	return	toggleHighQuality
delete	function	on	setVariable	stopDrag
do...while	getURL	onClipEvent	setProperty	trace
duplicateMovieClip	gotoAndPlay gotoAndStop	play	startDrag	unloadMovie
else	if	prevFrame	stop	var
else if	ifFrameLoaded	prevScene	stopAllSounds	while

Si desea ver ejemplos de la sintaxis y de la utilización de cada acción, consulte el Capítulo 7, “Diccionario de ActionScript”.

Nota: En este manual, el término *`action`* de ActionScript es sinónimo del término *`statement`* de JavaScript.

Programación de una ruta de destino

Para utilizar una acción a fin de controlar un clip de película o una película que ha sido cargada, debe especificar su nombre y su dirección, llamada *ruta de destino*. Las siguientes acciones toman una o más rutas de destino como argumentos:

- `loadMovie`
- `loadVariables`
- `unloadMovie`
- `setProperty`
- `startDrag`
- `duplicateMovieClip`
- `removeMovieClip`
- `print`
- `printAsBitmap`
- `tellTarget`

Por ejemplo, la acción `loadMovie` toma los argumentos *`URL`*, *`Location`*, y *`Variables`*. La dirección *`URL`* es la ubicación en la Web de la película que desea cargar. *`Location`* es la ruta de destino en la que se cargará la película.

```
loadMovie(URL, Location, Variables);
```

Nota: El argumento *`Variables`* no es necesario para este ejemplo.

La siguiente sentencia carga la dirección URL `http://www.mySite.com/myMovie.swf` en la instancia `bar` en la Línea de tiempo principal, `_root`; `_root.bar` es la ruta de destino;

```
loadMovie("http://www.mySite.com/myMovie.swf", _root.bar);
```

En ActionScript se identifica a un clip de película por su nombre de instancia. Por ejemplo, en la siguiente sentencia, la propiedad `_alpha` del clip de película llamado `star` está establecida a un 50% de visibilidad:

```
star._alpha = 50;
```

Dar un nombre de instancia a un clip de película:

1. Seleccione el clip de película en el Escenario.
2. Seleccione Ventana > Paneles > Instancia.
3. Introduzca un nombre de instancia en el campo Nombre.

Para identificar una película que ha sido cargada:

Utilice `_levelX` donde `X` es el número de nivel especificado en la acción `loadMovie` que cargó la película.

Por ejemplo, una película que ha sido cargada en el nivel 5 tiene el nombre de instancia `_level5`. En el siguiente ejemplo, se carga una película en el nivel 5 y su visibilidad está establecida como falsa:

```
onClipEvent(load) {
    loadMovie("myMovie.swf", 5);
}
onClipEvent(enterFrame) {
   _level5._visible = false;
}
```

Para introducir una ruta de destino de película:

Haga clic sobre el botón Insertar ruta de destino en el panel Acciones y seleccione un clip de película de la lista que se muestra.

Si desea obtener más información sobre cómo escribir rutas de destino, consulte el Capítulo 4, "Trabajo con clips de película".

Control de los scripts de flujo

ActionScript utiliza las acciones `if`, `for`, `while`, `do...while`, y `for...in` para realizar una acción dependiendo de si se da una condición.

Utilización de las sentencias if

Las sentencias que comprueban si una condición es verdadera o falsa comienzan con el término `if`. Si la condición se da, ActionScript ejecuta la sentencia que aparece a continuación de la misma. Si la condición no se da, ActionScript salta a la siguiente sentencia en el bloque de código.

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

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

```
if ((password == null) || (email == null)){
   gotoAndStop("reject");
}  else {
   gotoAndPlay("startMovie");
}
```

Repetición de una acción

ActionScript puede repetir una acción un número especificado de veces o mientras se dé una condición. Utilice las acciones `while`, `do...while`, `for`, y `for...in` para crear bucles.

Para repetir una acción mientras se dé una condición:

Utilice la sentencia `while`.

Un bucle `while` evalúa una expresión y ejecuta el código del bucle si la expresión es `true`. Después de que se ha ejecutado cada sentencia del bucle, la expresión se evalúa de nuevo. En el siguiente ejemplo, el bucle se ejecuta cuatro veces:

```
i = 4
while (i > 0) {
   myMC.duplicateMovieClip("newMC" + i, i );
   i --;
}
```

Puede utilizar la sentencia `do...while` para crear el mismo tipo de bucle como el de un bucle `while`. En un bucle `do...while` la expresión se evalúa al final del segmento de código de modo que el bucle siempre se ejecuta al menos una vez, como se muestra a continuación:

```
i = 4
do {
   myMC.duplicateMovieClip("newMC" +i, i );
   i --;
} while (i > 0);
```

Para repetir una acción utilizando un contador incorporado:

Utilice la sentencia `for`.

La mayoría de los bucles utilizan algún tipo de contador para controlar cuantas veces se ejecuta el bucle. Puede declarar una variable y escribir una sentencia que incremente o disminuya el valor de la variable cada vez que se ejecute el bucle. En la acción `for`, el contador y la sentencia que incrementa el contador son parte de la acción, como se muestra a continuación:

```
for (i = 4; i > 0; i--){
   myMC.duplicateMovieClip("newMC" + i, i + 10);
}
```

Para que se realice el bucle a través de los subniveles de un clip de película u objeto:

Utilice la sentencia `for...in`.

Los subniveles incluyen otros clips de películas, funciones, objetos y variables. El siguiente ejemplo utiliza `trace` para imprimir sus resultados en la ventana Salida:

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

Este ejemplo produce los siguientes resultados en la ventana Salida:

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

Puede que desee que su script se repita sobre un tipo de subnivel en particular: por ejemplo, solamente sobre el clip de película secundario. Puede hacer esto con `for...in` junto con el operador `typeof`.

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

Nota: La sentencia `for...in` se repite sobre las propiedades de los objetos que forman parte de la cadena prototipo del objeto sobre el que se está efectuando la repetición. Si el prototipo de un objeto secundario es `parent`, `for...in` también se repetirá en las propiedades de `parent`. Consulte "Definición de herencia" a pagina 86.

Si desea obtener más información sobre cada acción, consulte el Capítulo 7, "Diccionario de ActionScript".

Utilización de las funciones predefinidas

Una función es un bloque de código de ActionScript que puede volver a utilizarse en cualquier parte de una película. Si pasa valores específicos llamados argumentos a una función, la función actuará sobre esos valores. Una función también puede devolver valores. Flash dispone de funciones predeterminadas que le permiten acceder a cierta información y realizar ciertas tareas, como la detección de choque (`hitTest`), la obtención del valor de la última tecla presionada (`keycode`), y la obtención del número de versión de Flash Player que contiene la película (`getVersion`).

Llamada a una función

Puede llamar a una función de cualquier Línea de tiempo desde cualquier Línea de tiempo, incluida una película que ha sido cargada. Cada función tiene sus propias características y algunas necesitan que se le pasen ciertos valores. Si se le pasan más argumentos de los que necesita la función, los valores sobrantes se ignorarán. Si no se le pasa un argumento necesario, los argumentos vacíos se asignarán al tipo de datos `undefined`, que puede causar errores cuando exporte un script. Para llamar a una función, ésta debe encontrase en un fotograma al que haya llegado la cabeza lectora.

Las funciones predeterminadas de Flash se enumeran en la siguiente tabla:

Boolean	getTimer	isFinite	newline	scroll
escape	getVersion	isNaN	number	String
eval	globalToLocal	keycode	parseFloat	targetPath
false	hitTest	localToGlobal	parseInt	true
getProperty	int	maxscroll	random	unescape

Nota: Las funciones de cadena no están aprobadas y no aparecen en la lista de la tabla anterior.

Para llamar a una función en Modo Experto:

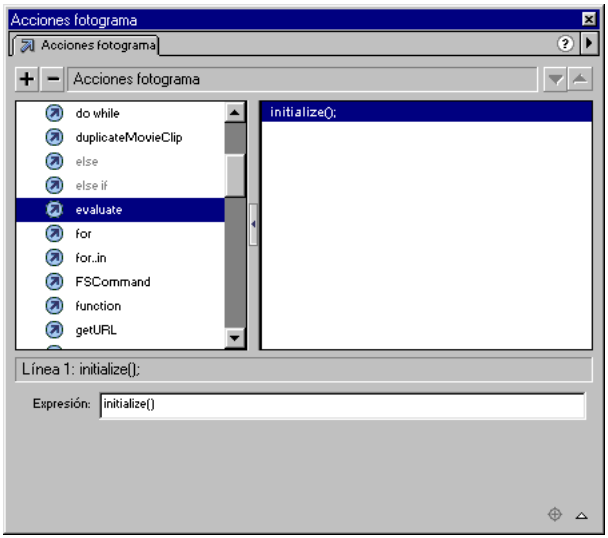
Utilice el nombre de la función. Pase cualquiera de los argumentos necesarios entre paréntesis.

El siguiente ejemplo llama a la función `initialize` que no requiere argumentos:

```
initialize();
```

Para llamar a una función en Modo Normal:

Utilice la acción `evaluate`. Introduzca el nombre de la función y cualquier argumento necesario en el campo Expresión.



Utilice la acción evaluate para llamar a una función en Modo Normal

Para llamar a una función en otra Línea de tiempo utilice una ruta de destino. Por ejemplo, para llamar a la función `calculateTax` que se declaró en la instancia `functionsMovieClip`, utilice la siguiente ruta:

```
_root.functionsMovieClip.calculateTax(total);
```

Nota: Pase cualquiera de los argumentos entre paréntesis.

Si desea obtener más información sobre cada función, incluidas las funciones de cadena no aprobadas, consulte el Capítulo 7, “Diccionario de ActionScript”.

Creación de funciones personalizadas

Puede definir funciones para que éstas ejecuten una serie de sentencias en función de los valores que se le han pasado. Sus funciones también pueden devolver valores. Una vez que se ha definido una función, podrá realizar una llama a la misma desde cualquier Línea de tiempo, incluida la Línea de tiempo de una película que ha sido cargada.

Se puede pensar en una función como si fuera una "caja negra": al realizar una llama a una función, se deberá proporcionar con entrada (argumentos). La función realizará alguna operación y después generará un resultado (un valor de retorno). Una función escrita correctamente debe incluir comentarios bien situados acerca de los valores de entrada, su resultado y su propósito. De este modo, el usuario de la función no tendrá que entender exactamente cómo actúa la función.

Definición de una función

Las funciones, al igual que las variables, están vinculadas al clip de película que las define. Cuando se vuelve a definir una función, la nueva definición sustituye a la definición anterior.

Para definir una función, utilice la acción `function` seguida del nombre de la función, los argumentos que va a pasar a la función y las sentencias de ActionScript que indican lo que hace la acción.

A continuación se muestra una función llamada `Circle` con el argumento `radius`:

```
function Circle(radius) {
   this.radius = radius;
   this.area = Math.PI * radius * radius;
}
```

Nota: La palabra clave `this`, utilizada en el cuerpo de una función, hace referencia al clip de película al que pertenece la función.

También puede definir la función creando una *expresión literal de función*. Una expresión literal de función es una función sin nombre que se declara en una expresión en lugar de en una sentencia. Puede utilizar la expresión literal de una función para definir una función, para devolver su valor y asignarlo a una variable en una expresión, como se muestra a continuación:

```
area = (function () {return Math.PI * radius *radius;})(5);
```

Pasar argumentos a una función

Los argumentos son los elementos sobre los que una función ejecuta su código. (En este manual, los términos *argumento* y *parámetro* pueden intercambiarse.) Por ejemplo, la función siguiente toma los argumentos `initials` y `finalScore`:

```
function fillOutScorecard(initials, finalScore) {
   scorecard.display = initials;
   scorecard.score = finalScore;
}
```

Cuando se llama a la función, los argumentos que ésta necesita deberán pasarse a la misma. La función sustituye los valores pasados por los argumentos en la definición de la función. En este ejemplo, `scorecard` es el nombre de instancia de un clip de película; `display` y `score` son campos de texto de entrada en la instancia. La siguiente llamada a una función asigna a la variable `display` el valor `"JEB"` y a la variable `score` el valor `45000`:

```
fillOutScorecard("JEB", 45000);
```

El argumento `initials` en la función `fillOutScorecard` es similar a una variable local, existe mientras que se llama a la función y deja de existir cuando se sale de la función. Si omite los argumentos durante la llamada a una función, los argumentos omitidos se pasan como `undefined`. Si introduce argumentos de más en una llamada a una función que no son necesarios para la declaración de la función, éstos se ignorarán.

Utilización de variables locales en una función

Las variables locales son herramientas muy valiosas para organizar el código y hacer que sea más sencillo de entender. Cuando una función utiliza variables locales, la función puede ocultar sus variables de todos los demás scripts de la película, las variables locales tienen su ámbito en el cuerpo de la función y se destruyen cuando se sale de la función. Cualquiera de los argumentos que se pasen a una función también se tratarán como variables locales.

Nota: Si modifica las variables locales de una función, utilice los comentarios de script para documentar estas modificaciones.

Devolución de valores de una función

Puede utilizar la función `return` para devolver valores de las funciones. La acción `return` detiene la función y la sustituye con el valor de la acción `return`. Si Flash no encuentra una acción `return` antes del final de una función, devolverá una cadena vacía. Por ejemplo, la siguiente función devuelve el cuadrado del argumento `x`:

```
function sqr(x) {
   return x * x;
}
```

Algunas funciones realizan una serie de tareas sin devolver un valor. Por ejemplo, la siguiente función inicializa una serie de variables globales:

```
function initialize() {
   boat_x = _root.boat._x;
   boat_y = _root.boat._y;
   car_x = _root.car._x;
   car_y = _root.car._y;
}
```

Llamada a una función

Para invocar una función utilizando el panel Acciones en Modo Normal, utilice la acción `evaluate`. Pase los argumentos necesarios entre paréntesis. Puede llamar a una función de cualquier Línea de tiempo desde cualquier Línea de tiempo, incluida una película que ha sido cargada. Por ejemplo, la siguiente sentencia invoca a la función `sqr` del clip de película `MathLib` de la Línea de tiempo principal, le pasa el argumento 3 y almacena el resultado en la variable `temp`:

```
var temp = _root.MathLib.sqr(3);
```

En Flash 4, para simular la llamada a una función podría escribir un script en un fotograma después del final de la película e invocarle pasando el nombre de la etiqueta del fotograma a la acción `call`. Por ejemplo, si el script que inicializó las variables se encontraba en un fotograma con la etiqueta `initialize`, la llamada se realizaría como se muestra a continuación:

```
call("initialize");
```

Este tipo de script no era una verdadera función ya que no podría aceptar argumentos y no podría devolver un valor. Aunque la acción `call` aún funciona en Flash 5, no se recomienda su utilización.

Utilización de objetos predefinidos

Puede utilizar los objetos predefinidos de Flash para acceder a ciertos tipos de información. La mayoría de los objetos predefinidos tienen *métodos* (funciones asignadas a un objeto) que puede llamar para devolver un valor o para realizar una acción. Por ejemplo, el objeto Date devuelve información del reloj del sistema y el objeto Sound le permite controlar los elementos de sonido de su película.

Algunos objetos predefinidos tienen propiedades cuyos valores pueden ser leídos por la misma. Por ejemplo, el objeto Key tiene valores constantes que representan teclas del teclado. Cada objeto tiene sus propias características y capacidades que pueden utilizarse en su película.

A continuación se muestran los objetos predefinidos de Flash:

- Array
- Boolean
- Color
- Date
- Key
- Math
- MovieClip
- Number
- Object
- Selection
- Sound
- String
- XML
- XMLSocket

Las instancias de clip de película se representan como objetos en ActionScript. Las llamadas a los métodos de clip de película predefinidos se pueden efectuar del mismo modo que como llamaría a los métodos de cualquier otro objeto de ActionScript.

Si desea obtener información detallada sobre cada objeto, consulte el Capítulo 7, “Diccionario de ActionScript”.

Creación de un objeto

Existen dos modos de crear un objeto: el operador `new` y el operador inicializador de objeto (`{}`). Puede utilizar el operador `new` para crear un objeto desde una clase de objetos predefinidos o desde una clase de objetos definidos personalizados. Puede utilizar el operador inicializador de objeto (`{}`) para crear un objeto del tipo genérico Object.

Para utilizar el operador `new` a fin de crear un objeto, deberá utilizarlo con una función constructor. (Una función constructor es sencillamente una función cuyo único propósito es crear un cierto tipo de objeto.) Los objetos predefinidos de ActionScript son en esencia funciones constructor escritas previamente. El objeto nuevo *instancia,* o crea, una copia del objeto y le asigna todas las propiedades y métodos de ese objeto. Esto es similar a arrastrar un clip de película desde la Biblioteca hasta el Escenario en una película. Por ejemplo, las siguientes sentencias instancian un objeto Date:

```
currentDate = new Date();
```

Puede acceder a los métodos de algunos objetos predefinidos sin instanciarlos. Por ejemplo, la siguiente sentencia llama al método del objeto Math `random`:

```
Math.random();
```

Cada objeto que necesite una función constructor tiene su elemento correspondiente en la caja de herramientas del panel Acciones, por ejemplo, `new Color`, `new Date`, `new String`, y así sucesivamente.

Para crear un objeto con el operador `new` en Modo Normal:

1. Seleccione `setVariable`
2. Introduzca un nombre de variable en el campo Nombre.
3. Introduzca `new Object`, `new Color`, y así sucesivamente en el campo Valor. Introduzca cualquiera de los argumentos necesarios para la función constructor entre paréntesis.
4. Marque el cuadro Expresión en el campo Valor.

   Si no activa el cuadro Expresión, todo el valor será un literal de cadena.

En el siguiente código, el objeto `c` se crea desde el constructor Color:

```
c = new Color(this);
```

Nota: Un nombre de objeto es una variable con el tipo de datos de objeto que se le ha asignado.

Acceso a un método en Modo Normal:

1. Seleccione la acción `evaluate`.
2. Introduzca el nombre del objeto en el campo Expresión.
3. Introduzca una propiedad del objeto en el campo Expresión.

Utilización del operador inicializador de objeto (`{}`) en Modo Normal:

1. Seleccione la acción `setVariable`.
2. Introduzca el nombre en el campo Variable, este es el nombre del objeto nuevo.
3. Introduzca los pares nombre de propiedad y valor separados por dos puntos dentro del operador inicializador de objeto (`{}`).

Por ejemplo, en esta sentencia los nombres de propiedad son `radius` y `area` y sus valores son `5` y el valor de una expresión:

```
myCircle = {radius: 5, area:(pi * radius * radius)};
```

Los paréntesis hacen que la expresión se evalúe. El valor que se devuelve es el valor de la variable `area`.

También puede anidar los inicializadores de matriz y de objeto, como en esta sentencia:

```
newObject = {name: "John Smith", projects: ["Flash",
"Dreamweaver"]};
```

Si desea obtener información detallada sobre cada objeto, consulte el Capítulo 7, “Diccionario de ActionScript”.

Acceso a las propiedades del objeto

Utilice el operador punto (`.`) para acceder al valor de las propiedades en un objeto. El nombre del objeto se haya a la izquierda del punto y el nombre de la propiedad a la derecha. Por ejemplo, en la siguiente sentencia, `myObject` es el objeto y `name` es la propiedad:

```
myObject.name
```

Para asignar un valor a una propiedad en Modo Normal, utilice la acción `setVariable`:

```
myObject.name = "Allen";
```

Para cambiar el valor de una propiedad, asigne un nuevo valor como se muestra a continuación:

```
myObject.name = "Homer";
```

También puede utilizar el operador de acceso a una matriz (`[]`) para acceder a las propiedades de un objeto. Consulte “Operadores punto y de acceso a una matriz” a pagina 69.

Llamada a métodos de objeto

Puede llamar al método de un objeto utilizando el operador punto seguido por el método. Así, el siguiente ejemplo llama al método `setVolume` del objeto Sound:

```
s = new Sound(this);
s.setVolume(50);
```

Para llamar al método de un objeto predefinido en Modo Normal, utilice la acción `evaluate`.

Utilización del objeto MovieClip

Puede utilizar los métodos del objeto predefinido MovieClip para controlar las instancias del símbolo del clip de película en el Escenario. El siguiente ejemplo dice a la instancia `dateCounter` que se ejecute:

```
dateCounter.play();
```

Si desea obtener información detallada sobre el objeto MovieClip, consulte el Capítulo 7, “Diccionario de ActionScript”.

Utilización del objeto Array

El objeto Array es un objeto de ActionScript predefinido utilizado con frecuencia que almacena sus datos en propiedades numeradas en lugar de en propiedades con nombre. El nombre de un elemento de matriz se llama *índice*. Esto es muy útil para almacenar y recuperar ciertos tipos de información como una lista de estudiantes o una secuencia de movimientos en un juego.

Puede asignar elementos del objeto Array del mismo modo en que lo haría con la propiedad de cualquier objeto:

```
move[1] = "a2a4";
move[2] = "h7h5";
move[3] = "b1c3";
...
move[100] = "e3e4";
```

Para acceder al segundo elemento de la matriz, utilice la expresión `move[2]`.

El objeto Array tiene una propiedad predefinida `length` que es el valor del número de elementos de la matriz. Cuando se asigna un elemento del objeto Array y el índice del elemento es un número entero positivo como `index >= length`, `length` se actualiza automáticamente a `index + 1`.

Utilización de objetos personalizados

Puede crear objetos personalizados para organizar la información de sus scripts con el fin de almacenarlos y acceder a ellos fácilmente; para ello es necesario definir las propiedades y métodos del objeto. Después de crear un objeto maestro o "clase", puede utilizar o "instanciar" copias (es decir, instancias) de ese objeto en una película. Esto le permite reutilizar el código y tener control sobre el tamaño del archivo.

Un objeto es un tipo de datos complejo que contiene cero o más propiedades. Cada propiedad tiene un nombre y un valor, como lo tiene una variable. Las propiedades están vinculadas al objeto y contienen los valores que pueden cambiarse y recuperarse. Estos valores pueden ser de cualquier tipo de datos: cadena, número, Booleano, clip de película o `undefined`. Las siguientes propiedades son varios tipos de datos:

```
customer.name = "Jane Doe"
customer.age = 30
customer.member = true
customer.account.currentRecord = 000609
customer.mcInstanceName._visible = true
```

La propiedad de un objeto también puede ser un objeto. En la línea 4 del ejemplo anterior, `account` es una propiedad del objeto `customer` y `currentRecord` es una propiedad del objeto `account`. El tipo de datos de la propiedad `currentRecord` es numérico.

Creación de un objeto

Puede utilizar el operador `new` para crear un objeto desde una función constructor. A una función constructor siempre se le da el mismo nombre que al tipo de objeto que está creando. Por ejemplo, una función constructor que creara un objeto de tipo cuenta se llamaría `Account`. La siguiente sentencia crea un nuevo objeto desde la función llamada `MyConstructorFunction`:

```
new MyConstructorFunction (argument1, argument2, ... argumentN);
```

Cuando se llama a `MyConstructorFunction`, Flash le pasa el argumento oculto `this`, que es una referencia al objeto que está creando `MyConstructorFunction`. Cuando define una función constructor, `this` le permite hacer referencia a los objetos que creará el constructor. Por ejemplo, a continuación se muestra una función constructor que crea un círculo:

```
function Circle(radius) {
   this.radius = radius;
   this.area = Math.PI * radius * radius;
}
```

Las funciones constructor se usan con frecuencia para rellenar los métodos de un objeto.

```
function Area() {
   this.circleArea = Math.PI * radius * radius;
}
```

Para utilizar un objeto en un script, deberá asignarlo a una variable. Para crear un nuevo objeto círculo con un radio de 5, utilice el operador`new` para crear el objeto y asignarle una variable local `myCircle`:

```
var myCircle = new Circle(5);
```

Nota: Los objetos tienen el mismo ámbito que la variable a la están asignados. Consulte "Ámbito de una variable" a pagina 61.

Definición de herencia

Todas las funciones tienen una propiedad `prototype` que se crea automáticamente cuando se define la función. Cuando utiliza una función constructor para crear un nuevo objeto, todas las propiedades y métodos de la propiedad `prototype` del constructor se convierten en propiedades y métodos de la propiedad `__proto__` del nuevo objeto. La propiedad `prototype` indica los valores de la propiedad predeterminados para los objetos creados con esa función. Al hecho de pasar valores utilizando las propiedades `__proto__` y `prototype` se llama herencia.

La herencia se comporta de acuerdo a una jerarquía determinada. Cuando llama a la propiedad o al método de un objeto, ActionScript busca en el objeto para ver si existe tal elemento. Si no existe, ActionScript busca en la propiedad `__proto__` del objeto la información (`object.__proto__`). Si la propiedad llamada no es una propiedad de objeto `__proto__`, Actionscript busca en `object.__proto__.__proto__`.

Es bastante común asociar métodos a un objeto asignándolos a la propiedad `prototype` del objeto. Los siguientes pasos describen cómo definir un método de ejemplo:

1 Defina la función constructor `Circle`, como se detalla a continuación:

```
function Circle(radius) {
    this.radius = radius
}
```

2 Defina el método `area` del objeto `Circle`. El método `area` calculará el área del círculo. Puede utilizar un literal de función para definir el método `area` y establecer la propiedad área del objeto prototipo del círculo, como se detalla a continuación:

```
Circle.prototype.area = function () {
       return Math.PI * this.radius * this.radius
    }
```

3 Cree una instancia del objeto `Circle`, como se detalla a continuación:

```
var myCircle = new Circle(4);
```

4 Llame al método `area` del nuevo objeto `myCircle`, como se detalla a continuación:

```
var myCircleArea = myCircle.area()
```

ActionScript busca el objeto `myCircle` para el método `area`. Como el objeto no tiene un método `area`, se busca su objeto prototipo `Circle.prototype` para el método`area`. ActionScript lo encuentra y lo llama.

También puede vincular un método a un objeto anexando el método a cada instancia individual del objeto, como se muestra en este ejemplo:

```
function Circle(radius) {
   this.radius = radius
   this.area = function() {
      return Math.PI * this.radius * this.radius
   }
}
```

No se recomienda esta técnica. Es más eficiente utilizar el objeto `prototype`, ya que solamente una definición de `area` es necesaria y esa definición se copia automáticamente en todas las instancias creadas por la función `Circle`.

La propiedad `prototype` está admitida en la versión 5 de Flash Player y posteriores. Si desea obtener más información, consulte el Capítulo 7, “Diccionario de ActionScript”.

Apertura de archivos de Flash 4

ActionScript ha cambiado substancialmente en la última versión de Flash 5. Ahora se trata de un lenguaje orientado a objetos con múltiples tipos de datos y sintaxis de punto. ActionScript de Flash4 solamente tenía un tipo de datos verdadero: cadena. Utilizaba diferentes tipos de operadores en expresiones para indicar si el valor debería ser tratado como una cadena o como un número. En Flash 5, puede utilizar un conjunto de operadores sobre todos los tipos de datos.

Cuando utiliza Flash 5 para abrir un archivo creado en Flash 4, Flash automáticamente convierte las expresiones de ActionScript para hacerlas compatibles con la nueva sintaxis de Flash 5. Verá el siguiente tipo de datos y las conversiones de operador en su código de ActionScript:

- El operador = en Flash 4 se utilizó para la igualdad numérica. En Flash 5, == es el operador de igualdad y = es el operador asignado. Cualquiera de los operadores = en los archivos de Flash 4 se convierte automáticamente en ==.
- Flash realiza automáticamente conversiones de tipo para asegurar que los operadores se comportan del modo esperado. Debido a la introducción de múltiples tipos de dato, los siguientes operadores tienen nuevos significados:

  `+, ==, !=, <>, <, >, >=, <=`
- En ActionScript de Flash 4, estos operadores siempre eran operadores numéricos. En Flash 5, se comportan de modo diferente dependiendo de los tipos de datos de los operandos. Para evitar cualquier diferencia semántica en los archivos importados, la función `Number` se inserta alrededor de todos los operandos de estos operadores. (Los números constantes evidentemente ya son números, de modo que no se incluyen entre `Number`).
- En Flash 4, la secuencia de escape`\n` generaba un carácter de retorno de carro (ASCII 13). En Flash 5, para cumplir el estándar ECMA-262, `\n` genera un carácter de avance de línea (ASCII 10). Una secuencia `\n` en los archivos FLA de Flash 4 se convierte automáticamente en `\r`.
- El operador `&` en Flash 4 se utilizaba para la suma de cadenas. En Flash 5, `&` es el operador AND como bit. El operador de suma de cadenas ahora se llama `add`. Cualquiera de los operadores `&` en los archivos de Flash 4 se convierten automáticamente en operadores `add`.
- Muchas de las funciones de Flash 4 no necesitan paréntesis de cierre, por ejemplo, `Get Timer`, `Set Variable`, `Stop` y `Play`. Para crear una sintaxis coherente, la función `getTimer` de Flash 5 y todas las acciones ahora requieren paréntesis de cierre. Estos paréntesis se agregan automáticamente durante la conversión.

- Cuando la función `getProperty` se ejecuta en un clip de película que no existe, devuelve el valor `undefined`, no devuelve 0, en Flash 5. Y `undefined == 0` es `false` en ActionScript de Flash 5. Flash soluciona este problema cuando convierte los archivos de Flash 4 introduciendo funciones `Number` en las comparaciones de igualdad. En el siguiente ejemplo, `Number` hace que `undefined` se convierta en 0 de modo que la comparación sea efectiva:

```
getProperty("clip", _width) == 0
Number(getProperty("clip", _width)) == Number(0)
```

Nota: Si ha utilizado cualquiera de las palabras clave de Flash 5 en el ActionScript de Flash 4, la sintaxis devolverá un error en Flash 5. Para solucionarlo, cambie el nombre de sus variables en todas las ubicaciones. Consulte "Palabras clave" a pagina 55.

Utilización de Flash 5 para crear contenidos de Flash 4

Si está utilizando Flash 5 para crear contenidos para Flash Player 4 (exportándolo como Flash 4), no podrá aprovechar las ventajas de las nuevas funciones presentes en ActionScript de Flash 5. Sin embargo, muchas de las nuevas características de ActionScript aún se encuentran disponibles. ActionScript de Flash 4 solamente tiene un tipo de datos primitivo básico que se utiliza tanto para la manipulación numérica como para manipulación de cadenas. Cuando es autor de una película para Flash Player 4, necesitará utilizar los operadores de cadena no aprobados situados en la categoría Operadores de cadena de la caja de herramientas.

Puede utilizar las siguientes características de Flash 5 cuando exporte al formato de archivo SWF de Flash 4.

- El operador de acceso a objeto y a matriz (`[]`).
- El operador de punto (`.`).
- Los operadores lógicos, los operadores de asignación y los operadores de incremento/decremento previo e incremento/decremento posterior.
- El operador módulo (`%`), todos los métodos y propiedades del objeto Math.

  Flash Player 4 no admite estos operadores y funciones originalmente. Flash 5 debe exportarlos como aproximaciones de series. Esto quiere decir que los resultados son solamente aproximados. Además, debido a la inclusión de aproximaciones de series en el archivo SWF, estas funciones ocupan más espacio en los archivos SWF de Flash 4 que en los archivos SWF de Flash 5.
- Las acciones `for`, `while`, `do while`, `break` y `continue`.
- Las acciones `print` y `printAsBitmap`.

Las siguientes funciones de Flash 5 no pueden utilizarse en películas exportadas al formato de archivo SWF de Flash 4:

- Funciones personalizadas
- Soporte XML
- Variables locales
- Objetos predefinidos (excepto objetos Math)
- Acciones de clip de película
- Tipos de datos múltiples
- `eval` con sintaxis de punto (por ejemplo, `eval("_root.movieclip.variable")`)
- `return`
- `new`
- `delete`
- `typeof`
- `for...in`
- `keycode`
- `targetPath`
- `escape`
- `globalToLocal` y `localToGlobal`
- `hitTest`
- `isFinite` y `inNaN`
- `parseFloat` y `parseInt`
- `tunescape`
- `_xmouse` y `_ymouse`
- `_quality`

CAPÍTULO 3
Interactividad con ActionScript

Una película interactiva toma en cuenta a la audiencia. Mediante el teclado, el ratón o ambos, la audiencia puede acceder a distintas partes de las películas, mover objetos, introducir información, hacer clic en botones y llevar a cabo muchas otras operaciones interactivas.

Como desarrollador, usted crea su película interactiva con scripts, cuyas especificaciones establecen que se deberán ejecutar cuando se presenten eventos específicos. Los eventos que ocasionan que se accione un script se dan cuando la cabeza lectora llega a un fotograma, cuando se carga o descarga un clip de película o cuando el usuario hace clic con el ratón o presiona una tecla del teclado. ActionScript se utiliza para crear scripts que indican a Flash la acción que debe llevar a cabo cuando ocurra el evento.

Las siguientes acciones básicas son modos comunes de control de la navegación e interacción del usuario en una película:

- Reproducción y detención de películas
- Ajuste de la calidad de visualización de una película
- Detención de todos los sonidos
- Salto a fotogramas o a escenas
- Salto a otras URL
- Comprobación de carga de fotogramas
- Carga y descarga de películas adicionales

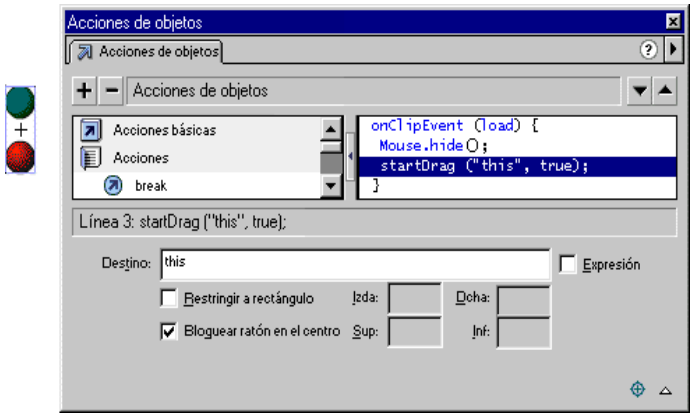
Si desea obtener información detallada sobre estas acciones, consulte *Utilización de Flash*.

Si desea crear una interactividad de mayor nivel de complejidad, es necesario que comprenda las siguientes técnicas:

- Creación de un cursor personalizado
- Obtención de la posición del ratón
- Captura de presión de una tecla
- Creación de un campo de texto desplazable
- Establecimiento de los valores del color
- Creación de controles de sonido
- Detección de colisiones

Creación de un cursor personalizado

Para ocultar el cursor estándar (es decir, la representación en pantalla del puntero del ratón), puede utilizar el método `hide` del objeto predefinido Mouse. Para utilizar un clip de película como el cursor personalizado, puede utilizar la acción `startDrag`.



Acciones asociadas a un clip de película para crear un cursor personalizado

Para crear un cursor personalizado:

1. Cree un clip de película para utilizarlo como cursor personalizado.
2. Seleccione el clip de película en el Escenario.
3. Seleccione Ventana > Acciones para abrir el panel Acciones de objeto.
4. En la lista de la Caja de herramientas, seleccione Objetos; después seleccione Mouse y arrastre `hide` a la ventana Script.

   El código deberá ser similar al siguiente:

```
onClipEvent(load){
    Mouse.hide();
}
```

5. En la lista de la Caja de herramientas, seleccione Acciones; después arrastre `startDrag` a la ventana Script.
6. Active el cuadro Bloquear ratón al centro.

   El código deberá ser similar al siguiente:

```
onClipEvent(load){
    Mouse.hide()
    startDrag(this, true);
}
```

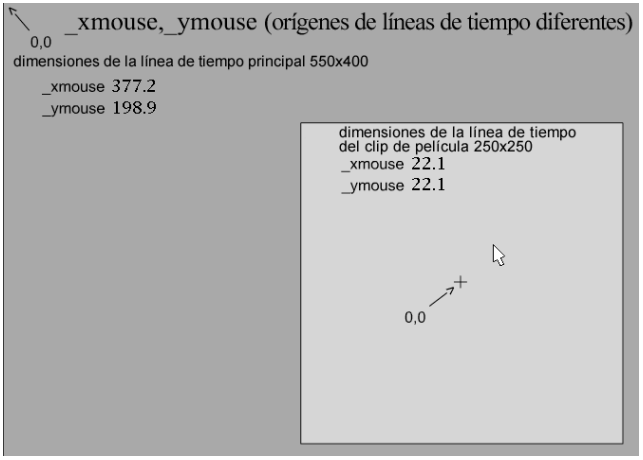
7. Seleccione Control > Probar película para utilizar el cursor personalizado.

Los botones seguirán funcionado cuando utilice un cursor personalizado. Es una buena idea colocar el cursor personalizado en la capa superior de la Línea de tiempo de modo que se mueva por encima de los botones y de otros objetos cuando desplace el ratón por la película.

Si desea obtener más información sobre los métodos del objeto Mouse, consulte el Capítulo 7, “Diccionario de ActionScript”.

Obtención de la posición del ratón

Puede utilizar las propiedades `_xmouse` e `_ymouse` para encontrar la ubicación del puntero del ratón (cursor) en una película. Cada Línea de tiempo tiene una propiedad `_xmouse` e `_ymouse` que devuelven la ubicación del ratón dentro de su sistema de coordenadas.



Las propiedades `_xmouse` e `_ymouse` en la Línea de tiempo principal y en la Línea de tiempo de un clip de película

La siguiente sentencia podría colocarse en cualquier Línea de tiempo de la película `_level0` para que devuelva la posición `_xmouse` en la Línea de tiempo principal:

```
x_pos = _root._xmouse;
```

Para determinar la posición del ratón dentro de un clip de película, puede utilizar el nombre de instancia del clip de película. Por ejemplo, la siguiente sentencia podría colocarse en cualquier Línea de tiempo de la película `_level0` para que devuelva la posición `_ymouse` en la instancia `myMovieClip`:

```
y_pos = _root.myMovieClip._ymouse
```

También puede determinar la posición del ratón en un clip de película utilizando las propiedades _xmouse e _ymouse en una acción de clip, como se muestra a continuación:

```
onClipEvent(enterFrame){
   xmousePosition = _xmouse;
   ymousePosition = _ymouse;
}
```

Las variables `x_pos` e `y_pos` se utilizan como contenedores para guardar los valores de las posiciones del ratón. Puede utilizar estas variables en cualquier script de su película. En el siguiente ejemplo, los valores de `x_pos` e `y_pos` se actualizan cada vez que el usuario mueve el ratón.

```
onClipEvent(mouseMove){
   x_pos = _root._xmouse;
   y_pos = _root._ymouse;
}
```

Si desea obtener más información sobre las propiedades `_xmouse` e `_ymouse`, consulte el Capítulo 7, “Diccionario de ActionScript”.

Captura de presión de una tecla

Puede utilizar los métodos del objeto predefinido Key para detectar la ultima tecla que ha sido presionada por el usuario. El objeto Key no requiere una función constructora, para utilizar sus métodos, simplemente llame al propio objeto, como se muestra en el siguiente ejemplo:

```
Key.getCode();
```

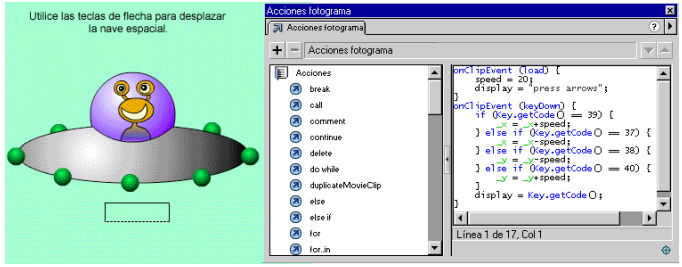
Puede obtener códigos de tecla virtuales o valores ASCII de la tecla que ha sido presionada:

- Para obtener el código de tecla virtual de la última tecla presionada, utilice el método `getCode`.
- Para obtener el valor ASCII de la última tecla presionada, utilice el método `getAscii`.

Se asigna un código de tecla virtual a cada tecla física del teclado. Por ejemplo, la tecla de flecha izquierda tiene el código virtual 37. Utilizando el código de tecla virtual, puede asegurarse de que los controles de su película son los mismos en cualquier teclado sin tener en cuenta el lenguaje o la plataforma.

Los valores ASCII (código americano estándar para intercambio de información) se asignan a los primeros 127 caracteres de todos los conjuntos de caracteres. Los valores ASCII proporcionan información sobre un carácter de la pantalla. Por ejemplo, la letra “A” y la letra “a” tiene diferentes valores ASCII.

Comunmente se utiliza `Key.getCode` en un controlador `onClipEvent`. Pasando `keyDown` como parámetro, el controlador ordena a ActionScript que compruebe el valor de la última tecla presionada sólo cuando realmente se presiona la tecla. Este ejemplo usa `Key.getCode` en una sentencia `if` para crear controles de navegación para la nave espacial.



Para crear controles de teclado para una película:

1. Decida qué teclas va a utilizar y determine sus códigos virtuales utilizando alguno de los siguientes métodos:

- Consulte la lista de códigos de tecla del Apéndice B, “Teclas del teclado y valores de códigos de tecla”.
- Utilice una constante de objeto Key. (En la lista de la Caja de herramientas, seleccione Objetos y a continuación Key. Las constantes de la lista aparecen en mayúsculas).
- Asigne la siguiente acción de clip, después seleccione Control > Probar película y presione la tecla que desee.

```
onClipEvent(keyDown) {
    trace(Key.getCode());
}
```

2. Seleccione un clip de película en el Escenario.
3. Seleccione Ventana > Acciones.
4. Haga doble clic sobre la acción `onClipEvent` en la categoría Acciones de la caja de herramientas.
5. Seleccione el evento `Key down` en el panel de parámetros.
6. Haga doble clic sobre la acción `if` en la categoría Acciones de la caja de herramientas.
7. Haga clic sobre el parámetro Condition, seleccione Objects y después Key y `getCode`.
8. Haga doble clic sobre el operador de igualdad (==) en la categoría Operadores de la caja de herramientas.

9 Introduzca el código de tecla virtual a la derecha del operador de igualdad.

El código deberá ser similar al siguiente:

```
onClipEvent(keyDown) {
    if (Key.getCode() == 32) {
    }
}
```

10 Seleccione una acción a llevar a cabo si se presiona la tecla correcta.

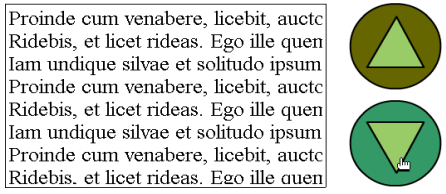
Por ejemplo, la siguiente acción hace que la Línea de tiempo principal vaya al siguiente fotograma cuando se presiona la Barra espaciadora (32).

```
onClipEvent(keyDown) {
    if (Key.getCode() == 32) {
       nextFrame();
    }
}
```

Si desea obtener más información sobre los métodos del objeto Key, consulte el Capítulo 7, “Diccionario de ActionScript”.

Creación de un campo de texto desplazable

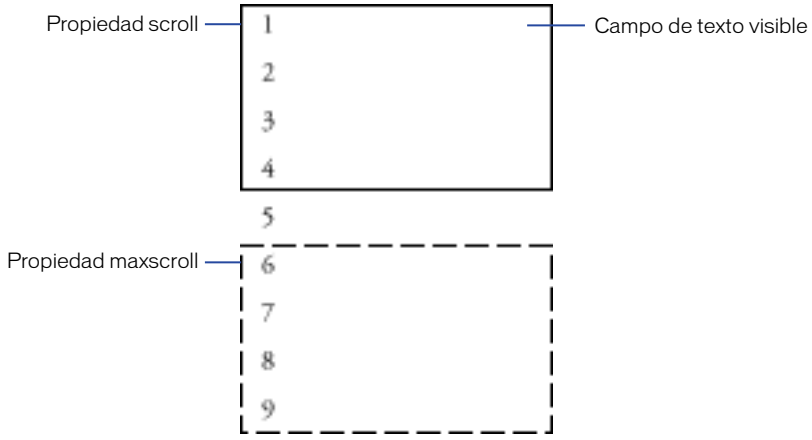
Puede utilizar las propiedades scroll y maxscroll para crear un campo de texto desplazable.



En el panel Opciones de texto, puede asignar una variable a cualquier campo de texto establecido como Introducción de texto o Texto dinámico. El campo de texto actúa como una ventana que indica el valor de esa variable.

Cada variable asociada con un campo de texto tiene una propiedad `scroll` y una propiedad `maxscroll`. Puede utilizar estas propiedades para desplazar texto en un campo de texto. La propiedad `scroll` devuelve el número de la línea superior visible en un campo de texto y puede establecerla o recuperarla. La propiedad `maxscroll` devuelve la línea superior visible de un campo de texto cuando la línea inferior está visible, puede leer, pero no establecer esta propiedad.

Por ejemplo, suponga que tiene un campo de texto que dispone de cuatro líneas. Si este campo contiene la variable `speech` que tiene nueve líneas de texto, solamente cuatro de las líneas se visualizarían en el campo de texto, como se muestra en el recuadro sólido:

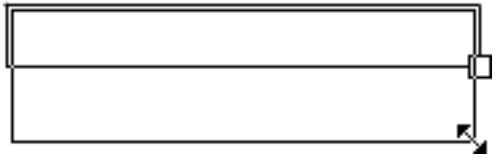


Puede acceder a estas propiedades utilizando la sintaxis de punto, como se muestra a continuación:

```
textFieldVariable.scroll
myMovieClip.textFieldVariable.scroll
textFieldVariable.maxscroll
myMovieClip.textFieldVariable.maxscroll
```

Para crear un campo de texto con desplazable:

1. Arrastre un campo de texto al Escenario.
2. Seleccione Ventana > Paneles > Opciones de texto.
3. Elija Introducción de texto en el menú emergente.
4. Introduzca el nombre de variable **text** en el campo Variable.
5. Arrastre la esquina inferior derecha del campo de texto para cambiar el tamaño del campo de texto.



6. Seleccione Ventana > Acciones.

7 Seleccione el fotograma 1 en la Línea de tiempo principal y asigne una acción `set variable` que establezca el valor de `text`.

No aparecerá texto en el campo hasta que se haya establecido la variable. Por lo tanto, aunque puede asignar esta acción a cualquier fotograma, botón o clip de película, es buena idea asignar la acción al fotograma 1 en la Línea de tiempo principal, como se muestra a continuación:

8 Seleccione Ventana > Bibliotecas comunes > Botones y arrastre un botón hasta el Escenario.

9 Pulse Alt (en Windows) o bien Opción (en Macintosh) y arrastre el botón para crear una copia.

10 Seleccione el botón superior y elija Ventana > Acciones.

11 Arrastre la acción `set variables` desde la caja de herramientas hasta la ventana Script en el panel Acciones.

12 Introduzca **text.scroll** en el cuadro Variable.

13 Introduzca **text.scroll -1** en el cuadro Valor y active la casilla de verificación Expresión.

14 Seleccione el botón de flecha hacia abajo y asigne la siguiente acción `set variables`:

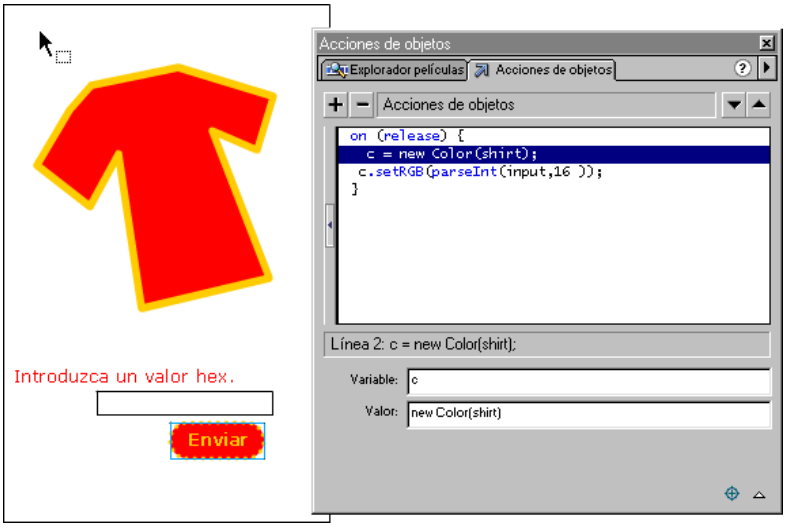
```
text.scroll = text.scroll+1;
```

15 Seleccione Control > Probar película para comprobar el campo de texto desplazable.

Si desea obtener más información sobre las propiedades `scroll` y `maxscroll`, consulte el Capítulo 7, “Diccionario de ActionScript”.

Establecimiento de los valores de color

Puede utilizar los métodos del objeto predefinido Color para ajustar el color de un clip de película. El método `setRGB` asigna valores hexadecimales RGB (rojo, verde, azul) al objeto y el método `setTransform` establece el porcentaje de valores de desplazamiento para los componentes rojo, verde, azul y transparencia (alfa) de un color. El siguiente ejemplo utiliza `setRGB` para cambiar el color de un objeto basándose en la información introducida por el usuario.



La acción del botón crea un objeto de color y cambia el color de la camiseta basándose en la información introducida por el usuario.

Para utilizar el objeto Color, necesita crear una instancia del objeto y aplicarla a un clip de película.

Para establecer el valor de color de un clip de película:

1. Seleccione un clip de película en el Escenario y elija Ventana > Paneles > Instancia.
2. Introduzca el nombre de instancia **colorTarget** en el cuadro Nombre.
3. Arrastre un campo de texto al Escenario.
4. Seleccione Ventana > Paneles > Opciones de texto y asígnele el nombre de variable **input**.
5. Arrastre un botón al Escenario y selecciónelo.
6. Seleccione Ventana > Acciones.

7 Arrastre la acción `set variable` desde la caja de herramientas hasta la ventana Script.

8 En el cuadro Variable, introduzca **c**.

9 En la caja de herramientas, seleccione Objetos, después Color, y arrastre `new Color` al cuadro Color.

10 Active la casilla de verificación Expresión.

11 Haga clic sobre el botón Ruta de destino y seleccione `colorTarget`. Haga clic en Aceptar.

El código de la ventana Script deberá parecerse al que se muestra a continuación:

```
on(release) {
    c = new Color(colorTarget);
}
```

12 Arrastre la acción `evaluate` desde la caja de herramientas hasta la ventana Script.

13 Introduzca **c** en el cuadro Expresión.

14 En la categoría Objetos de la lista de la Caja de herramientas, seleccione Color y arrastre `setRGB` al cuadro Expresión.

15 Seleccione Funciones y arrastre `parseInt` al cuadro Expresión.

El código deberá ser similar al siguiente:

```
on(release) {
    c = new Color(colorTarget);
    c.setRGB(parseInt(string, radix));
}
```

16 Para el argumento de cadena `parseInt`, introduzca **input**.

La cadena que se va a analizar es el valor introducido en el campo de texto editable.

17 Para el argumento base `parseInt`, introduzca **16**.

La base es la base matemática del número del sistema que se va a analizar. En este caso, 16 es la base del sistema hexadecimal que utiliza el objeto Color. El código deberá ser similar al siguiente:

```
on(release) {
    c = new Color(colorTarget);
    c.setRGB(parseInt(input, 16));
}
```

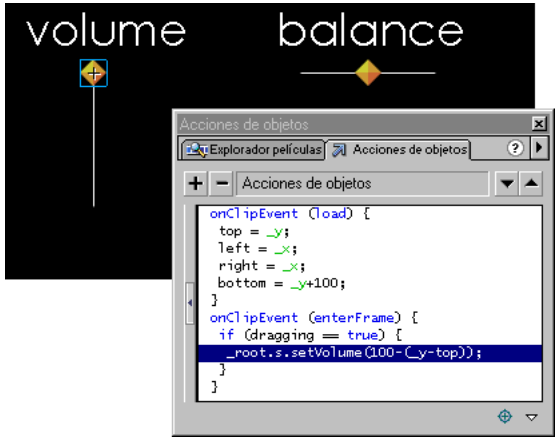
18 Seleccione Control > Probar película para cambiar el color del clip de película.

Si desea obtener más información sobre los métodos del objeto Color, consulte el Capítulo 7, “Diccionario de ActionScript”.

Creación de controles de sonido

Para controlar los sonidos de una película, debe utilizar el objeto predefinido Sound. Para utilizar los métodos del objeto Sound, deberá crear primero un nuevo objeto Sound. Después puede utilizar el método `attachSound` para insertar un sonido de la biblioteca en una película cuando la película se está reproduciendo.

El método `setVolume` del objeto Sound controla el volumen y el método `setPan` ajusta el balance izquierdo y derecho de un sonido. El siguiente ejemplo utiliza `setVolume` y `setPan` para crear controles de balance y de volumen que el usuario puede ajustar.



Cuando el usuario arrastra el deslizador de volumen, se llama al método `setVolume`.

Asignación de un sonido a una Línea de tiempo:

1. Seleccione Archivo > Importar para importar un sonido.
2. Seleccione el sonido de la biblioteca y elija Vinculación en el menú Opciones.
3. Seleccione Exportar este símbolo y asígnele el identificador **mySound**.
4. Seleccione el fotograma 1 en la Línea de tiempo y elija Ventana > Acciones.
5. Arrastre la acción `set variable` desde la caja de herramientas hasta la ventana Script.
6. Introduzca **s** en el cuadro Valor.
7. En la lista de la caja de herramientas, seleccione Objetos, después Sound y arrastre `new Sound` al cuadro Valor.

   El código deberá ser similar al siguiente:

   ```
   s = new Sound();
   ```

8 Haga doble clic sobre la acción `evaluate` de la caja de herramientas.

9 Introduzca **s** en el cuadro Expresión.

10 En la categoría Objetos de la lista de la Caja de herramientas, seleccione Sound, después arrastre `attachSound` al cuadro Expresión.

11 Introduzca "**mySound**" en el argumento ID de `attachSound`.

12 Haga doble clic sobre la acción `evaluate` de la caja de herramientas.

13 Introduzca **s** en el cuadro Expresión.

14 En la categoría Objetos, seleccione Sound, después arrastre `start` al cuadro Expresión.

El código deberá ser similar al siguiente:

```
s = new Sound();
s.attachSound("mySound");
s.start();
```

15 Seleccione Control > Probar película para oír el sonido.

Para crear un control deslizable de volumen:

1 Arrastre un botón al Escenario.

2 Seleccione el botón y elija Insertar > Convertir en símbolo. Seleccione el comportamiento del clip de película.

Así se crea un clip de película con el botón en su primer fotograma.

3 Seleccione el clip de película y elija Editar > Editar símbolo.

4 Seleccione el botón y elija Ventana > Acciones.

5 Introduzca las acciones siguientes:

```
on (press) {
    startDrag ("", false, left, top, right, bottom);
    dragging = true;
}
on (release, releaseOutside) {
    stopDrag ();
    dragging = false;
}
```

Los parámetros de `startDrag`, `left`, `top`, `right` y `bottom` son variables definidas en una acción de clip.

6 Elija Edición > Editar película para volver a la Línea de tiempo principal.

7 Seleccione el clip de película en el Escenario.

8 Introduzca las acciones siguientes:

```
onClipEvent (load) {
    top=_y;
    left=_x;
    right=_x;
    bottom=_y+100;
}

onClipEvent(enterFrame){
    if (dragging==true){
       _root.s.setVolume(100-(_y-top));
    }
}
```

9 Seleccione Control > Probar película para utilizar el deslizador de volumen.

Para crear un control deslizable de balance:

1 Arrastre un botón al Escenario.

2 Seleccione el botón y elija Insertar > Convertir en símbolo. Seleccione la propiedad del clip de película.

3 Seleccione el clip de película y elija Editar > Editar símbolo.

4 Seleccione el botón y elija Ventana > Acciones.

5 Introduzca las siguientes acciones:

```
on (press) {
    startDrag ("", false, left, top, right, bottom);
    dragging = true;
}
on (release, releaseOutside) {
    stopDrag ();
    dragging = false;
}
```

Los parámetros de `startDrag`, `left`, `top`, `right` y `bottom` son variables definidas en una acción de clip.

6 Elija Edición > Editar película para volver a la Línea de tiempo principal.

7 Seleccione el clip de película en el Escenario.

8 Introduzca las siguientes acciones:

```
onClipEvent(load){
    top=_y;
    bottom=_y;
    left=_x-50;
    right=_x+50;
    center=_x;
}

onClipEvent(enterFrame){
    if (dragging==true){
       _root.s.setPan((_x-center)*2);
    }
}
```

9 Seleccione Control > Probar película para utilizar el deslizador de balance.

Si desea obtener más información sobre los métodos del objeto Sound, consulte el Capítulo 7, “Diccionario de ActionScript”.

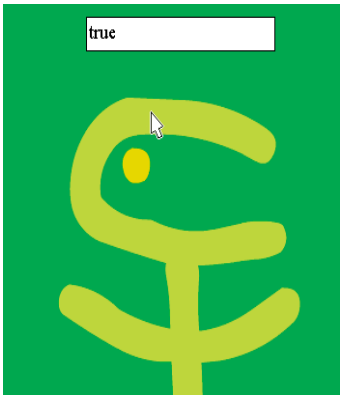
Detección de colisiones

Puede utilizar el método `hitTest` del objeto MovieClip para detectar colisiones en una película. El método `hitTest` comprueba si el objeto ha colisionado con un clip de película y devuelve un valor Booleano (`true` o `false`). Puede utilizar los parámetros del método `hitTest` para especificar las coordenadas *x* e *y* de un área de impacto en el Escenario, o utilizar la ruta de destino de otro clip de película como área de impacto.

Cada clip de película de una película es una instancia del objeto MovieClip. Esto le permite llamar a los métodos del objeto desde cualquier instancia, como se muestra a continuación:

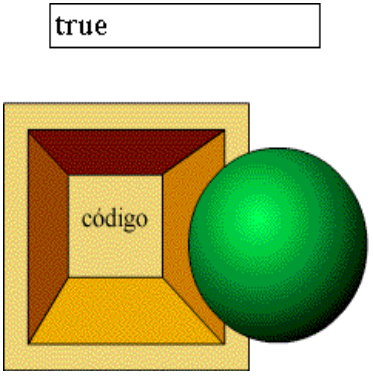
```
myMovieClip.hitTest(target);
```

Puede utilizar el método `hitTest` para comprobar la colisión en un clip de película y un solo punto



"True" aparece en el campo de texto siempre que el cursor esté sobre el área de color.

También puede utilizar el método `hitTest` para comprobar la colisión entre dos clips de película.



"True" aparece en el campo de texto siempre que un clip de película haga contacto con otro clip.

Para realizar una detección de colisión entre un clip de película y un punto del Escenario:

1. Seleccione un clip de película en el Escenario.
2. Seleccione Ventana > Acciones para abrir el panel Acciones de objeto.
3. Haga doble clic sobre la acción `trace` en la categoría Acciones de la caja de herramientas.
4. Active la casilla de verificación Expresión e introduzca lo siguiente en el cuadro del mismo nombre:

   ```
   trace (this.hitTest(_root._xmouse, _root._ymouse, true);
   ```

   Este ejemplo utiliza las propiedades `_xmouse` e `_ymouse` como las coordenadas *x* e *y* para el área de impacto y envía los resultados a la ventana Salida en el modo de prueba de película. También puede establecer un campo de texto en el Escenario para visualizar los resultados o utilizar los resultados en una sentencia `if`.
5. Seleccione Control > Probar película y mueva el ratón sobre el clip de película para comprobar la colisión.

Para realizar la detección de colisión en dos clips de película:

1. Arrastre dos clips de película al Escenario y asígneles los nombres de instancia **mcHitArea** y **mcDrag**.
2. Cree un campo de texto en el Escenario e introduzca **status** en el cuadro Variable opciones de texto.
3. Seleccione `mcHitArea` y elija Ventana > Acciones.
4. Haga doble clic sobre `evaluate` en la caja de herramientas.
5. Introduzca el siguiente código en el cuadro Expresiones seleccionando elementos de la caja de herramientas:

   ```
   _root.status=this.hitTest(_root.mcDrag);
   ```

6 Seleccione la acción `onClipEvent` en la ventana Script y elija `enterFrame` como el evento.

7 Seleccione `mcDrag` y elija Ventana > Acciones.

8 Haga doble clic sobre `startDrag` en la caja de herramientas.

9 Active la casilla de verificación Bloquear ratón en el centro.

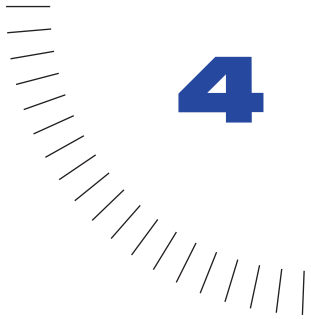
10 Seleccione la acción `onClipEvent` en la ventana Script y elija el evento `Mouse down`.

11 Haga doble clic sobre `stoptDrag` en la caja de herramientas.

12 Seleccione la acción `onClipEvent` en la ventana Script y elija el evento `Mouse up`.

13 Seleccione Control > Probar película y arrastre el clip de película para comprobar la detección de colisión.

Si desea obtener más información sobre el método `hitTest`, consulte el Capítulo 7, “Diccionario de ActionScript”.



CAPÍTULO 4
Trabajo con clips de película

Un clip de película es una mini película de Flash: tiene su propia Línea de tiempo y sus propias propiedades. Un símbolo de clip de película de la Biblioteca puede utilizarse varias veces en una película de Flash, cada utilización recibe el nombre de *instancia* de clip de película. Los clips de película pueden anidarse unos dentro de otros. Para distinguir unas instancias de otras, puede asignar a cada utilización de un clip de película un nombre de instancia.

Puede colocarse cualquier objeto en la Línea de tiempo de un clip de película, incluidos otros clips de película. Las películas que se cargan en Flash Player mediante `loadMovie` también son mini películas de Flash. Cada clip de película, la película cargada y la Línea de tiempo principal de una película de Flash son objetos con propiedades y métodos que se pueden manejar a través de ActionScript para crear una animación no lineal compleja y para programar poderosas interacciones.

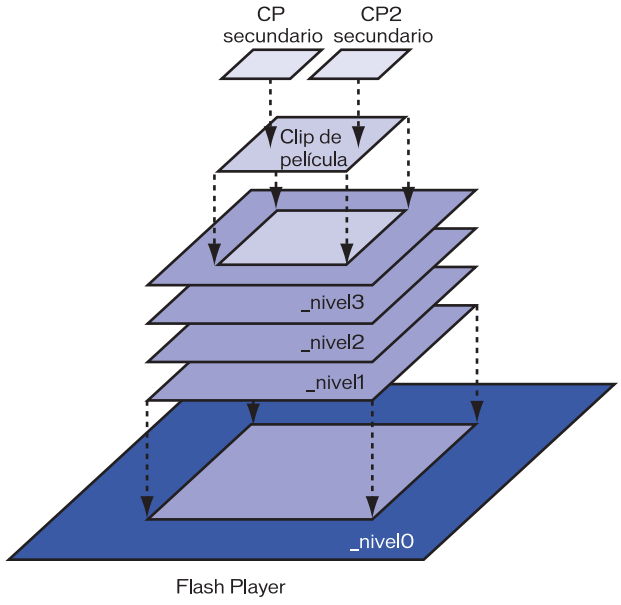
Controle los clips de película utilizando acciones y métodos del objeto MovieClip. Las acciones y los métodos pueden anexarse a fotogramas o a botones en un clip de película (acciones de fotograma y de botón) o a una instancia de clip de película específica (acciones de clip). Las acciones de un clip de película pueden controlar cualquier Línea de tiempo de una película. Para controlar una Línea de tiempo debe dirigirse a ella utilizando una ruta de destino. Una ruta de destino indica la ubicación de la Línea de tiempo en la película.

También puede convertir un clip de película en un Clip inteligente, un clip de película con ActionScript que puede volver a programarse sin utilizar el panel Acciones. Los clips inteligentes facilitan pasar objetos con lógica compleja de ActionScript de un programador a un diseñador.

Líneas de tiempo múltiples

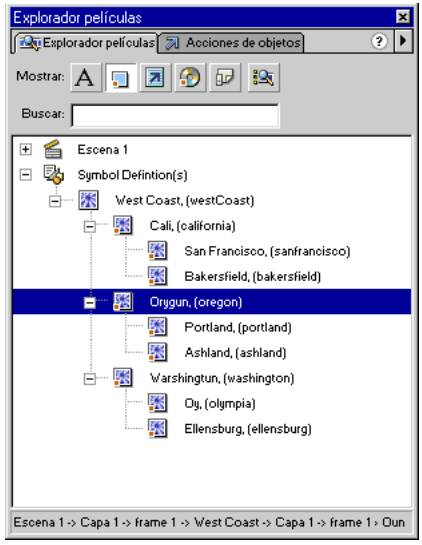
Todas las películas de Flash tienen una Línea de tiempo principal situada en el nivel 0 en Flash Player. Puede utilizar la acción `loadMovie` para cargar otras películas de Flash (archivos SWF) en Flash Player en cualquier nivel superior al nivel 0 (por ejemplo, nivel 1, nivel 2, nivel 15). Cada película cargada en un nivel de Flash Player tiene su propia Línea de tiempo.

Las películas de Flash en cualquier nivel pueden tener instancias de clips de película en sus Líneas de tiempo. Cada instancia de clip de película también tiene una Línea de tiempo que puede contener otros clips de película que también tienen Líneas de tiempo. Las Líneas de tiempo de los clips de película y de los niveles de Flash Player están organizadas de manera jerárquica para poder organizar y controlar fácilmente los objetos de su película.



La jerarquía de niveles y de clips de película en Flash Player.

En Flash, esta jerarquía de niveles y de clips de película recibe el nombre de *lista de visualización*. Puede ver la lista de visualización en el Explorador de películas cuando realice la autoría en Flash. Puede ver la lista de visualización en el Depurador cuando esté reproduciendo la película en el modo de prueba de película, en Flash Player independiente o en un navegador Web.



El Explorador de películas muestra la jerarquía de las Líneas de tiempo, denominada lista de visualización.

Las Líneas de tiempo en una película de Flash son objetos y todos tienen características (propiedades) y capacidades (métodos) del objeto predefinido MovieClip. Las Líneas de tiempo tienen relaciones específicas unas con otras dependiendo de su ubicación en la lista de visualización. Las Líneas de tiempo que se encuentran anidadas dentro de otras Líneas de tiempo se ven afectadas por los cambios realizados en la Línea de tiempo en la que residen. Por ejemplo, si `portland` es un elemento secundario de `oregon` y se cambia la propiedad `_xscale` de `oregon`, `portland` también cambiará de escala.

Las Líneas de tiempo también pueden enviar mensajes unas a otras. Por ejemplo, una acción del último fotograma de un clip de película podría decir a otro clip de película que se reproduzca.

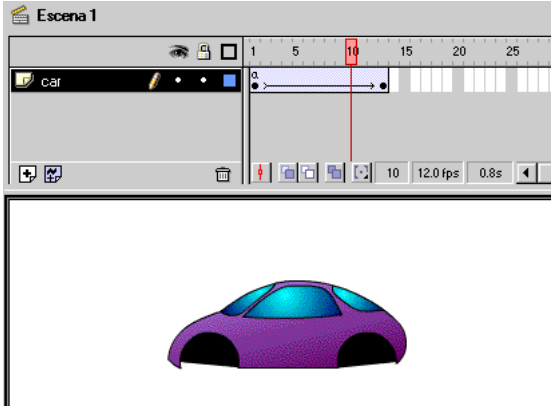
Relación jerárquica de las Líneas de tiempo

Cuando coloca una instancia de clip de película en la Línea de tiempo de otro clip de película, un símbolo de clip de película contiene la instancia del otro clip de película (el primer clip de película es *secundario* y el segundo clip de película es el *principal*). La Línea de tiempo principal de una película de Flash es la principal de todos los clips de película que estén en su nivel.

Las relaciones principal-secundario de los clips de película son jerárquicas. Para comprender esta jerarquía, considere la jerarquía de un PC: la unidad de disco duro tiene un directorio raíz (o carpeta) y subdirectorios. El directorio raíz es análogo a la Línea de tiempo principal de una película de Flash: es el nivel principal de todo lo demás. Los subdirectorios son análogos a los clips de película. Puede utilizar los subdirectorios para organizar los contenidos relacionados.

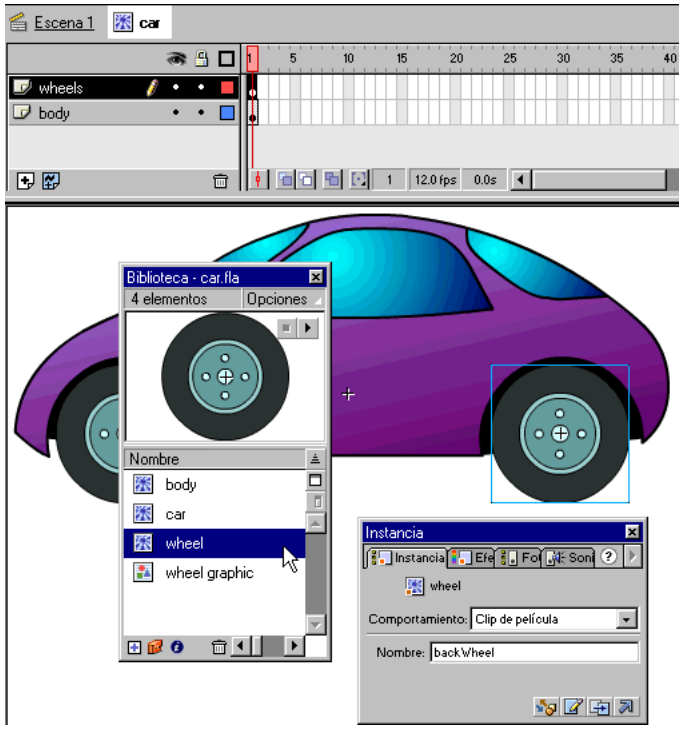
Igualmente, puede utilizar la jerarquía de clips de película de Flash para organizar los objetos visuales relacionados, con frecuencia de modo similar al comportamiento de los objetos en la realidad. Cualquier cambio que realice en un clip de película principal también se llevará a cabo en sus clips secundarios.

Por ejemplo, podría crear una película de Flash que tenga un coche que se mueve a través del Escenario. Podría utilizar un símbolo de clip de película para representar el coche y establecer una interpolación de movimiento para moverlo a través del Escenario.



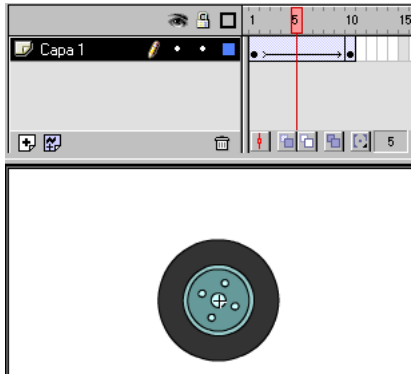
Una interpolación de movimiento mueve el clip de película del coche en la Línea de tiempo principal.

El coche se ve desde el lateral, con dos ruedas visibles. Una vez que tiene el coche moviéndose, puede agregar ruedas que giran. Así que cree un clip de película para una rueda del coche y cree dos instancias de este clip de película, llamadas `frontWheel` y `backWheel`. Entonces coloque las ruedas en la Línea de tiempo del clip de película coche, no en la Línea de tiempo principal. Como instancias secundarias de `car`, `frontWheel` y `backWheel` se ven afectadas por cualquiera de los cambios realizados en `car`. Esto quiere decir que se moverán con el coche conforme éste se interpola a través del Escenario.



Las instancias `frontWheel` y `backWheel` se colocan en la Línea de tiempo del clip de película `car`.

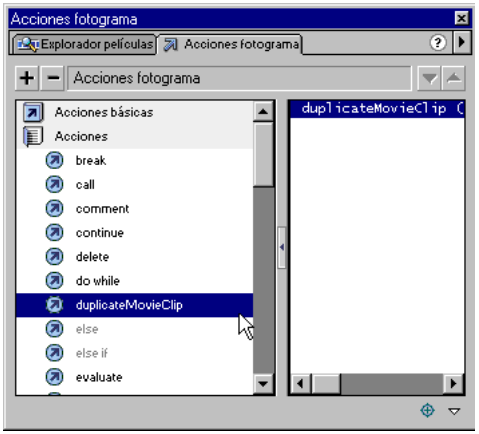
Para hacer que las ruedas giren, puede establecer una interpolación de movimiento para girar el símbolo de rueda y hacer que giren ambas instancias. Incluso después de haber cambiado `frontWheel` y `backWheel`, seguirá viéndose afectadas por la interpolación del su clip de película principal, `car`; las ruedas girarán, pero también se moverán con el clip de película principal `car` a través del Escenario.



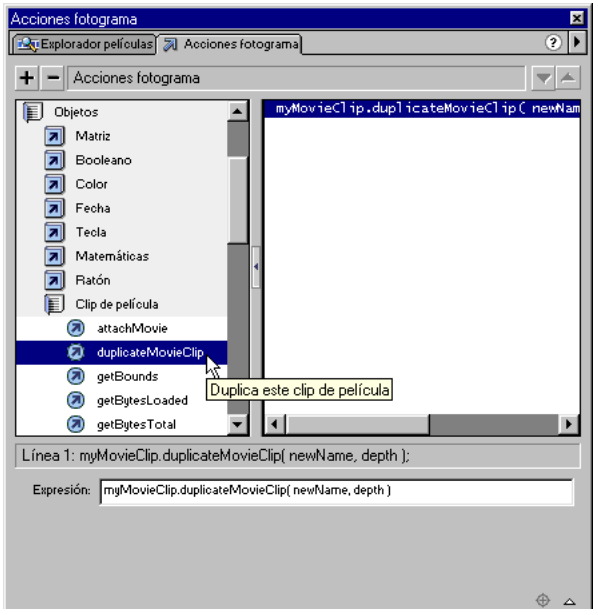
Envío de mensajes entre Líneas de tiempo

Puede enviar mensajes desde una Línea de tiempo a otra. Una Línea de tiempo contiene la acción, denominada *controlador*, y otro recibe la acción, llamada *destino*. Puede asignar una acción a un fotograma o botón en una Línea de tiempo, o bien, si la Línea de tiempo es un clip de película, al propio clip de película.

Para especificar el destino de las Líneas de tiempo, puede utilizar las acciones de la categoría Acciones o puede utilizar métodos del objeto MovieClip de la categoría Objetos en el panel Acciones. Por ejemplo, puede utilizar la acción `duplicateMovieClip` para especificar el destino y hacer copias de las instancias de clip de película mientras se reproduce una película.



Puede utilizar acciones de la categoría Acciones para seleccionar el destino de una Línea de tiempo.



Puede utilizar los métodos del objeto MovieClip para seleccionar el destino de una Línea de tiempo.

Para realizar varias acciones sobre el mismo destino, puede utilizar la acción `with`. Similar a la sentencia `with` de JavaScript, la acción `with` de ActionScript es una acción envolvente que le permite dirigir la Línea de tiempo de destino una vez, y después hacer que se ejecuten una serie de acciones sobre ese clip, no tiene que dirigir la Línea de tiempo destino en cada acción.

También puede utilizar la acción `tellTarget` para realizar varias acciones en el mismo destino.

La comunicación entre Líneas de tiempo debe hacerse como se muestra a continuación:

- Introduzca un nombre de instancia para el clip de película de destino.

  Para nombrar una instancia de clip de película, utilice el panel Instancia (Ventana > Paneles > Instancia). Las Líneas de tiempo cargadas en niveles utilizan su número de nivel como nombre de instancia, por ejemplo, `_level6`.

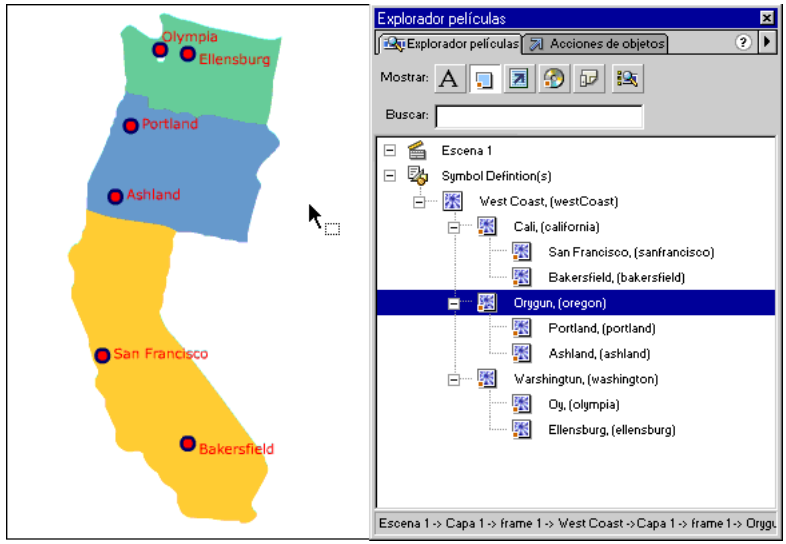
- Introduzca la ruta de destino del nombre de instancia en el panel Acciones.

  Puede introducir la ruta de destino manualmente, o puede utilizar el cuadro de diálogo Insertar ruta de destino para seleccionar el destino de un clip de película. (Consulte “Especificación de rutas de destino” a pagina 121).

Nota: Durante la reproducción, la Línea de tiempo de un clip de película debe estar en el escenario de destino.

Rutas de destino absolutas y relativas

Una ruta de destino es la dirección de la Línea de tiempo cuyo destino desea especificar. La lista de visualización de las Líneas de tiempo en Flash es similar a la jerarquía de archivos y carpetas de un servidor Web.



El Explorador de películas muestra la lista de visualización de los clips de película en modo de autor.

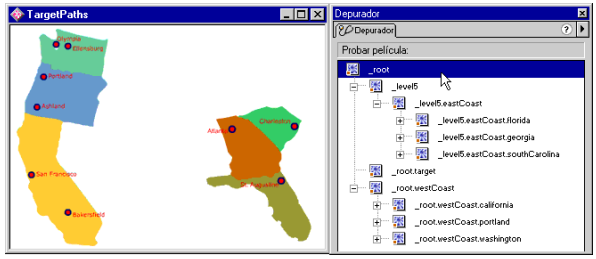
Al igual que en un servidor Web, puede ubicar cada Línea de tiempo en Flash de dos maneras: con una ruta absoluta o con una ruta relativa. La ruta absoluta de una instancia siempre es la misma, sin tener en cuenta qué Línea de tiempo llama a la acción; por ejemplo, la ruta absoluta a la instancia `california` es siempre `_level0.westCoast.california`. Una ruta relativa es diferente cuando se llama desde diferentes ubicaciones, por ejemplo, la ruta relativa hasta `california` desde `sanfrancisco` es `_parent`, pero desde `portland`, es`_parent._parent.california`.

Nota: Si desea obtener más información sobre el Explorador de películas, consulte la sección *Utilización de Flash*.

Una ruta absoluta comienza con el nombre del nivel en el que está cargada la película y continúa a lo largo de la lista de visualización hasta que alcanza la instancia de destino.

La primera película que se va a abrir en Flash Player se carga en el nivel 0. Debe asignar a cada película que se carga adicionalmente un número de nivel. El nombre de destino para un nivel es `_levelX` donde `X` es el número de nivel en el que está cargada la película. Por ejemplo, la primera película abierta en Flash Player se llama `_level0`, una película cargada en el nivel 3 se llama `_level3`.

En el ejemplo siguiente, dos películas se han cargado en el reproductor, `TargetPaths.swf` en el nivel 0 y `EastCoast.swf` en el nivel 5. Los niveles están indicados en el Depurador, con el nivel 0 indicado como `_root`.



El Depurador muestra la rutas absolutas de todas las Líneas de tiempo en la lista de visualización en modo de prueba de película.

Una instancia siempre tiene la misma ruta absoluta, ya sea llamada desde una acción en una instancia en el mismo nivel o desde otra acción en un nivel diferente. Por ejemplo, la instancia `bakersfield` en el nivel 0 siempre tiene la ruta absoluta siguiente en sintaxis de punto:

```
_level0.california.bakersfield
```

En sintaxis de barras, la ruta absoluta sustituye las barras por puntos, como se muestra a continuación:

```
_level0/california/bakersfield
```

Para comunicarse entre películas en diferentes niveles, debe utilizar el nombre de nivel en la ruta de destino. Por ejemplo, la instancia `portland` dirigiría la instancia `atlanta` como se muestra a continuación:

```
_level5.georgia.atlanta
```

En sintaxis de punto, puede utilizar el alias `_root` para hacer referencia a la Línea de tiempo principal del nivel actual. Para la Línea de tiempo principal, o `_level0`, el alias _root significa `_level0` cuando se selecciona como destino desde un clip también en `_level0`. Para una película cargada en `_level5`, `_root` es igual a `_level5` cuando se selecciona como destino por un clip de película también en el nivel 1. Por ejemplo, una acción llamada desde la instancia `southcarolina` podría utilizar la ruta absoluta siguiente para seleccionar el destino de la instancia `florida`:

```
_root.eastCoast.florida
```

En sintaxis de barras, puede utilizar `/` para hacer referencia a la Línea de tiempo principal del nivel actual, como se muestra a continuación:

```
/eastCoast/florida
```

En sintaxis de punto ya sea en modo relativo o absoluto, puede utilizar las mismas reglas de ruta de destino para identificar una variable en una Línea de tiempo o una propiedad de un objeto. Por ejemplo, la sentencia siguiente establece el nombre de variable en el formulario de instancia en el valor `"Gilbert"`:

```
_root.form.name = "Gilbert";
```

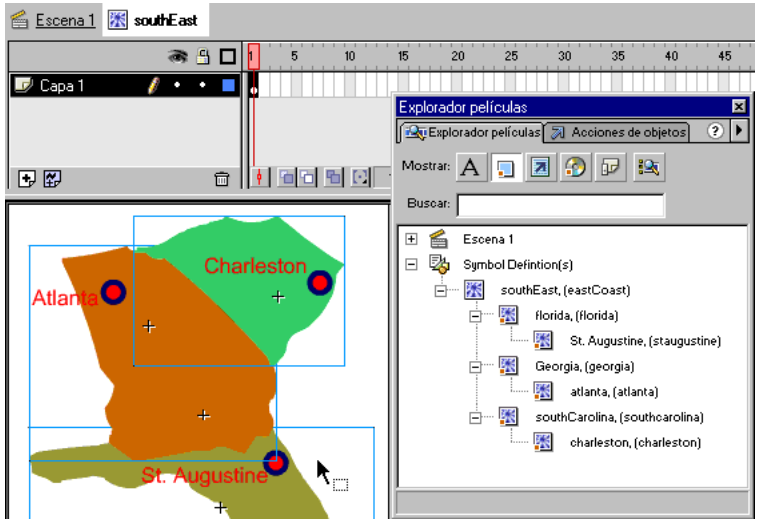
En sintaxis de barras ya sea en modo absoluto o relativo, puede identificar una variable en una Línea de tiempo precediendo el nombre de la variable por dos puntos (`:`), como se muestra a continuación:

```
/form:name = "Gilbert";
```

Una ruta relativa es dependiente de la relación entre la Línea de tiempo de controlador y la Línea de tiempo de destino. Puede utilizar una ruta relativa para volver a utilizar acciones debido a que la misma acción puede servir para diferentes Líneas de tiempo dependiendo de donde esté colocada la acción. Las rutas relativas se pueden dirigir solamente a destinos dentro de su propio nivel de Flash Player, no pueden dirigir películas cargadas en otros niveles. Por ejemplo, no puede utilizar una ruta relativa en una acción en `_level0` que especifica un destino para una Línea de tiempo en `_level5`.

En sintaxis de punto, puede utilizar la palabra clave `this` en una ruta de destino relativa para hacer referencia a la Línea de tiempo principal. Puede utilizar el alias `_parent` en una ruta de destino relativa para indicar la Línea de tiempo principal de la Línea de tiempo actual. El alias `_parent` puede utilizarse repetidamente para subir un nivel en la jerarquía de clip de película dentro del mismo nivel de Flash Player. Por ejemplo, `_parent._parent` controla un clip de película hasta dos niveles en la jerarquía.

En el ejemplo siguiente, cada ciudad (`charleston`, `atlanta`, y `staugustine`) son niveles secundarios de una instancia estado y cada estado (`southcarolina`, `georgia`, y `florida`) es un nivel secundario de la instancia `eastCoast`.



El Explorador de películas muestra las relaciones principal-secundario de los clips de película.

Una acción de la Línea de tiempo de la instancia `charleston` podría utilizar la ruta de destino siguiente para seleccionar el destino para la instancia `southcarolina`:

```
_parent
```

Para especificar el destino de la instancia `eastCoast` desde una acción en `charleston`, podría utilizar la ruta relativa siguiente:

```
_parent._parent
```

En sintaxis de barras, puede utilizar dos puntos (`..`) para subir un nivel en la jerarquía. Para seleccionar destino de `eastCoast` desde una acción en `charleston`, podría utilizar la ruta siguiente:

```
../..
```

Para especificar el destino de la instancia `atlanta` desde una acción en la Línea de tiempo de `charleston`, podría utilizar la ruta relativa siguiente mediante sintaxis de punto:

```
_parent._parent.georgia.atlanta
```

Las rutas relativas son útiles para volver a utilizar scripts. Por ejemplo, podría anexar un script a una película que amplifica el clip de película en 150%, como se muestra a continuación:

```
onClipEvent (load) {
   _parent._xscale = 150;
   _parent._yscale = 150;
}
```

Entonces podría volver a utilizar este script colocándolo en la Línea de tiempo de cualquier clip de película.

Si desea obtener más información sobre la dirección y la sintaxis de punto, consulte "Programación de Scripts en ActionScript" a pagina 51.

Si desea obtener más información sobre la sintaxis de punto y la sintaxis de barras, consulte "Utilización de la sintaxis de ActionScript" a pagina 52.

Especificación de rutas de destino

Para controlar un clip de película o una película cargada, debe utilizar una ruta de destino para especificar un destino. Un clip de película debe tener un nombre de instancia que se utilizará para asignarlo como destino. Puede especificar un destino de varias maneras diferentes:

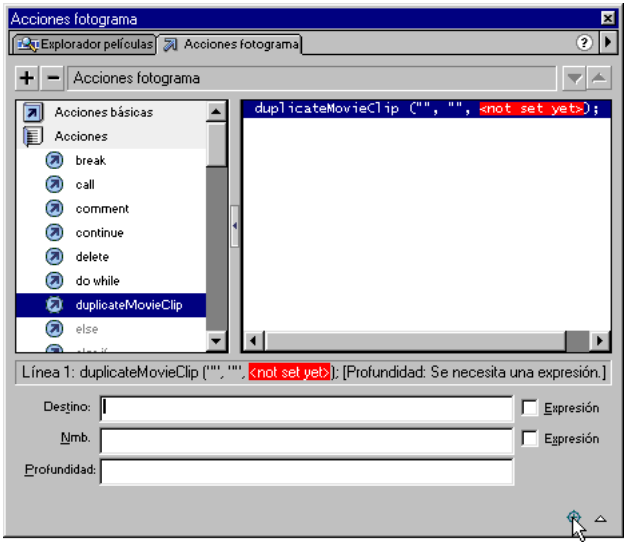
- Introduzca una ruta de destino utilizando el botón y el cuadro de diálogo Insertar ruta de destino en el panel Acciones.
- Introduzca la ruta de destino del clip de película en su script manualmente.
- Cree una expresión utilizando una referencia a un clip de película, o utilizando las funciones predefinidas `targetPath` y `eval`.

Para insertar una ruta de destino mediante el cuadro de diálogo Insertar ruta de destino:

1. Seleccione la instancia de clip de película, de fotograma o de botón a la que desea asignar la acción.

   Esta será la Línea de tiempo de controlador.
2. Seleccione Ventana> Acciones para ver el panel Acciones.
3. En la lista Caja de herramientas, elija una acción de la categoría Acciones o un método de la categoría MovieClip dentro de la categoría Objetos.
4. Haga clic sobre el campo o ubicación Destino en el script para insertar la ruta de destino.

5 Haga clic en el botón Insertar ruta de destino, situado en la esquina inferior derecha del panel Acciones para ver el cuadro de diálogo Insertar ruta de destino.



6 En el cuadro de diálogo Insertar ruta de destino, elija una sintaxis: De puntos (predeterminada) o de barras.

7 Elija Absoluto o Relativo para el modo de ruta de destino.

Consulte la sección “Rutas de destino absolutas y relativas” a pagina 117.

8 Especifique su destino realizando uno de los pasos que se detallan a continuación:

- Seleccione un clip de película en la lista de visualización Insertar ruta de destino.
- Introduzca un destino manualmente en el campo Destino utilizando una ruta absoluta o relativa y sintaxis de punto.

9 Haga clic en Aceptar.

Para insertar una ruta de destino manualmente:

Siga los pasos de 1 a 4 anteriores e introduzca una ruta de destino absoluta o relativa en el panel Acciones.

Para utilizar una expresión como ruta de destino:

1 Siga los pasos de 1 a 4 detallados anteriormente.

2 Siga uno de los pasos detallados a continuación:

- Introduzca manualmente una referencia como una ruta de destino. Una referencia se evalúa para determinar la ruta de destino. Puede utilizar una referencia como un parámetro para la acción `with`. En el ejemplo siguiente, la variable `index` se evalúa y multiplica por 2. El valor resultante se utiliza como el nombre del clip de película dentro de la instancia `Block` a la que se le dice que se reproduzca:

```
with (Board.Block[index*2]) {
    play();
}
```

- En la categoría Funciones de la lista de la Caja de herramientas, elija la función `targetPath`.

  La función `targetPath` convierte una referencia a un clip de película en una cadena que puede ser utilizada por acciones como `tellTarget`.

  En el ejemplo siguiente, la función `targetPath` convierte la referencia `Board.Block[index*2+1]` en una cadena:

```
tellTarget (targetPath (Board.Block[index*2+1])) {
    play();
}
```

  El ejemplo anterior es equivalente a la sintaxis de barras siguiente:

```
tellTarget ("Board/Block:" + index*2+1)) {
    play();
}
```

- En la categoría Funciones de la lista de la Caja de herramientas, elija la función `eval`.

  La función `eval` convierte una cadena en una referencia a un clip de película en una cadena que puede ser utilizada por acciones como `with`.

  El script siguiente evalúa la variable `i`, la agrega a la cadena `"cat"` y asigna el valor resultante a la variable `x` . La variable x es ahora una referencia a una instancia de clip de película y puede llamar a los métodos del objeto MovieClip, como se muestra a continuación:

  ```
  x = eval ("cat" + i)
  x.play()
  ```

  También puede utilizar la función `eval` para llamar métodos directamente, como se muestra a continuación:

  ```
  eval ("cat" + i).play()
  ```

Utilización de acciones y métodos para controlar Líneas de tiempo

Puede utilizar ciertas acciones y métodos del objeto MovieClip para *seleccionar como destino*, o llevar a cabo tareas en, un clip de película o un nivel cargado. Por ejemplo, la acción `setProperty` establece una propiedad (como `_width`) de una Línea de tiempo en un valor (como `100`). Algunos métodos del objeto MovieClip duplican la función de todas las acciones que tienen como destino a otras Líneas de tiempo. También hay métodos adicionales, como `hitTest`, y `swapDepths`. Ya se utilice una acción o un método, la Línea de tiempo de destino deberá estar cargada en Flash Player cuando se llame a la acción o al método.

Las acciones siguientes pueden seleccionar clips de película como destino: `loadMovie`, `unloadMovie`, `setProperty`, `startDrag`, `duplicateMovieClip`, y `removeMovieClip`. Para utilizar estas acciones, debe introducir una ruta de destino en el parámetro Target de la acción para indicar el receptor de la acción. Algunas de estas acciones pueden seleccionar como destino a clips de película o niveles cargados y otras solamente pueden seleccionar como destino a de clips de película.

Los siguientes métodos del objeto MovieClip pueden controlar clips de película o niveles cargados y no tienen acciones equivalentes: `attachMovie`, `getBounds`, `getBytesLoaded`, `getBytesTotal`, `globalToLocal`, `localToGlobal`, `hitTest` y `swapDepths`.

Cuando una acción o un método ofrecen funciones similares, puede elegir controlar clips de película utilizando cualquiera de ellos. La elección depende de lo que prefiera y de lo familiarizado que esté con la programación de scripts en ActionScript.

Si desea obtener más información sobre los métodos del objeto MovieClip e información sobre cada acción, consulte el Capítulo 7, “Diccionario de ActionScript” a pagina 175.

Acerca de métodos versus acciones

Para utilizar un método, se le debe invocar utilizando la ruta de destino al nombre de instancia, seguida de un punto y después el nombre del método y los argumentos, como en las sentencias siguientes:

```
myMovieClip.play();
parentClip.childClip.gotoAndPlay(3);
```

En la primera sentencia, el método `play` hace que la instancia `myMovieClip` se reproduzca. En la segunda sentencia, el método `gotoAndPlay` envía la cabeza lectora de `childClip` (que es secundaria de la instancia `parentClip`) al fotograma 3 y se reproduce.

Las acciones que controlan una Línea de tiempo tienen un parámetro Target que especifica la ruta de destino. Por ejemplo, en el script siguiente la acción `startDrag` selecciona como destino a la instancia `customCursor` y la hace arrastrable:

```
on(press){
   startDrag("customCursor");
}
```

Cuando se utiliza un método, se llama al método al final de la ruta de destino. Por ejemplo, la sentencia siguiente realiza la misma función `startDrag`:

```
customCursor.startDrag();
```

Las sentencias escritas utilizando los métodos del objeto MovieClip tienden a ser más breves ya que no requieren la acción `tellTarget`. No se recomienda la utilización de la acción `tellTarget` debido a que no es compatible con el estándar ECMA-262.

Por ejemplo, para indicar al clip de película `myMovieClip` que comience a reproducir utilizando los métodos del objeto MovieClip, se utilizaría el código que se muestra a continuación:

```
myMovieClip.play();
```

El código siguiente produce los mismos resultados utilizando la acción `tellTarget`:

```
tellTarget ("myMovieClip") {
   play();
}
```

Utilización de varios métodos o acciones para seleccionar como destino a una Línea de tiempo

Puede utilizar la acción `with` para referirse a un clip de película que ha sido asignado como destino una vez y después ejecutar una serie de acciones en ese clip. La acción `with` funciona en todos los objetos de ActionScript, (por ejemplo Array, Color y Sound) no solamente en clips de película. La acción `tellTarget` es similar a la acción `with`. Sin embargo, la acción `tellTarget` es menos adecuada, debido a que no funciona con todos los objetos de ActionScript y no sigue las especificaciones ECMA-262.

La acción `with` toma un objeto como parámetro. El objeto que especifique se agregará al final de la ruta de destino actual. Todas las acciones anidadas dentro de una acción `with` se llevan a cabo dentro de la nueva ruta de acceso, o *ámbito*. Por ejemplo, en el siguiente script de la Línea de tiempo principal, a la acción `with` se le pasa el objeto `donut.hole` para cambiar las propiedades de `hole`:

```
with (donut.hole) {
   _alpha = 20;
   _xscale = 150;
   _yscale = 150;
}
```

Es como si las sentencias dentro de la acción with fueran llamadas desde la Línea de tiempo de la instancia `hole`.

En el ejemplo siguiente, observe la economía de utilizar la acción `with` y los métodos del objeto MovieClip para especificar varias instrucciones:

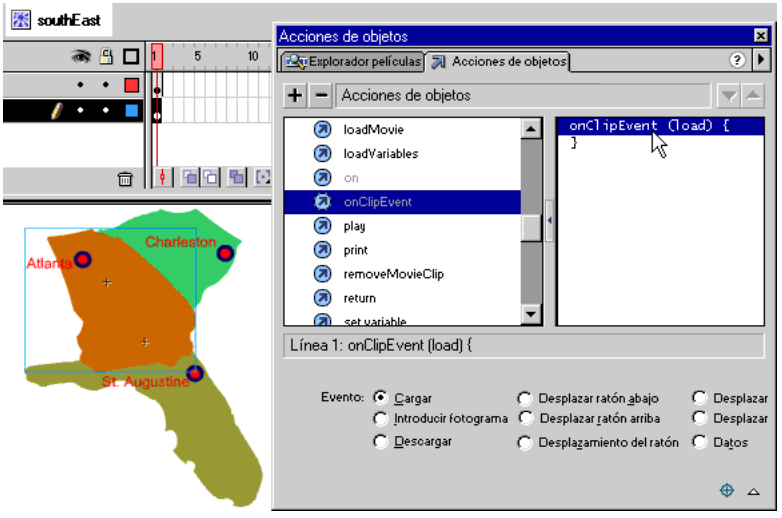
```
with (myMovieClip) {
   _x -= 10;
   _y += 10;
   gotoAndPlay(3);
}
```

Si desea obtener más información sobre la acción `tellTarget`, consulte *Utilización de Flash.*

Asignación de una acción o método

Las acciones y los métodos pueden asignarse a un botón o a un fotograma en la Línea de tiempo o a una instancia de clip de película.

Para asignar una acción o un método a una instancia de clip de película debe utilizar un controlador `onClipEvent`. Todas las acciones vinculadas a la instancia se anidan dentro de un controlador `onClipEvent` y se ejecutan una vez que éste se ha activado. La acción `onClipEvent` se activa mediante cualquiera de los eventos de Línea de tiempo (como cargar una película) o eventos de usuario (como hacer clic con el ratón o presionar una tecla). Por ejemplo, `onClipEvent(mouseMove)` activa una acción cada vez que el usuario mueve el ratón.



A la acción `onClipEvent` se le asigna a una instancia en el Escenario. Los eventos `onClipEvent` aparecen en la lista del panel de parámetros en el panel Acciones.

Carga y descarga de películas adicionales

Puede utilizar la acción o el método `loadMovie` para reproducir películas adicionales sin cerrar Flash Player, o intercambiar películas sin cargar otra página HTML. Puede utilizar `loadMovie` para enviar variables a un script CGI que genera un archivo SWF como resultado del CGI. Cuando carga una película, puede especificar un nivel o un clip de película de destino en el que se cargará la película.

La acción y el método `unloadMovie` elimina una película cargada previamente por `loadMovie`. La descarga de películas de modo explícito con `unloadMovie` asegura una transición fluida entre las películas y puede reducir la memoria requerida por Flash Player. Utilice la acción `loadMovie` para realizar cualquiera de las acciones siguientes:

- Reproducción de una secuencia de anuncios publicitarios que son archivos SWF, colocando una acción `Load Movie` al final de cada archivo SWF para cargar la película siguiente.
- Desarrollo de una interfaz ramificada que permite al usuario elegir entre varios archivos SWF diferentes.
- Creación de una interfaz de navegador con controles de navegación en el nivel 0 que carga otros niveles. Cargar niveles produce transiciones más suaves que cargar nuevas páginas HTML en un explorador.

Cambio de posición y aspecto de clips de película

Para cambiar las propiedades de un clip de película a medida que se reproduce, puede utilizar la acción `setProperty` o escribir una sentencia que asigna un valor a una propiedad. Si carga una película en un destino, la película cargada heredará las propiedades del clip de película de destino. Una vez que se ha cargado la película, puede cambiar dichas propiedades.

Algunas propiedades, llamadas propiedades de *sólo lectura*, tienen valores que se pueden leer pero no establecer. Puede escribir sentencias para establecer cualquier propiedad que no sea de sólo lectura. Las sentencias siguientes establecen la propiedad _alpha de la instancia del clip de película `wheel` que es secundaria de la instancia `car`:

```
car.wheel._alpha = 50;
```

Además, puede escribir sentencias que obtienen el valor de una propiedad de clip de película. Por ejemplo, la sentencia siguiente obtiene el valor de la propiedad `_xmouse` de la Línea de tiempo principal y establece la propiedad `_x` de la instancia `customCursor` en ese valor:

```
onClipEvent(enterFrame){
   customCursor._x = _root._xmouse;
}
```

También puede utilizar la función `getProperty` para recuperar las propiedades del clip de película.

Las propiedades `_x`, `_y`, `_rotation`, `_xscale`, `_yscale`, `_height`, `_width`, `_alpha` y `_visible` se ven afectadas por las transformaciones del elemento principal del clip de película y modifican el clip de película y cualquiera de los elementos secundarios del clip. Las propiedades `_focusrect`, `_highquality`, `_quality`, y `_soundbuftime` son globales, solamente pertenecen al nivel 0 de la Línea de tiempo. Todas las demás propiedades pertenecen a cada clip de película o nivel cargado. En la tabla siguiente se muestran todas las propiedades de clip de película:

Propiedades			
_alpha	_highquality	_totalframes	_xscale
_currentframe	_name	_url	_y
_droptarget	_quality	_visible	_ymouse
_focusrect	_rotation	_width	_yscale
_framesloaded	_soundbuftime	_x	
_height	_target	_xmouse	

Clips de película que se pueden arrastrar

Puede utilizar la acción o el método `startDrag` para hacer que se pueda arrastrar un clip de película mientras se reproduce una película. Por ejemplo, puede crear un clip de película que se pueda arrastrar para juegos, para funciones arrastrar y soltar, para interfaces personalizables, para barras de desplazamiento y para deslizadores.

Un clip de película puede ser arrastrado hasta que se detiene explícitamente por `stopDrag`, o hasta que se selecciona otro clip de película como destino de `startDrag`. Sólo se puede arrastrar un clip de película a la vez.

Para crear comportamientos de arrastrar y soltar más complejos, puede probar la propiedad `_droptarget` del clip de película que se está arrastrando. Por ejemplo, puede examinar la propiedad `_droptarget` para ver si la película se arrastró a un clip de película específico (como un clip de película tipo “papelera”) y, a continuación, desencadenar otra acción. Consulte “Utilización de las sentencias if” a pagina 73 y “Utilización de operadores para manipular los valores de las expresiones” a pagina 64.

Duplicación y eliminación de clips de película

Puede crear o eliminar instancias de clip de película según se reproduce su película utilizando `duplicateMovieClip` o `removeMovieClip`, respectivamente. La acción y el método `duplicateMovieClip` crean dinámicamente una nueva instancia del clip de película, asignándole un nuevo nombre de instancia y dándole una profundidad. Un clip de película duplicado siempre comienza en el fotograma 1 incluso si el clip de película original se encontraba en otro fotograma cuando se duplicó y siempre se encuentra encima de todos los clips de película situados en la Línea de tiempo. Las variables no se copian en el clip de película duplicado.

Para borrar un clip de película creado con `duplicateMovieClip`, utilice `removeMovieClip`. Los clips de película duplicados se eliminan si el clip de película principal se borra.

Anexión de clips de película

Puede recuperar una copia de un clip de película de la biblioteca y reproducirlo como parte de su película utilizando el método `attachMovie`. Este método carga otro clip de película en su clip de película y lo reproduce según se ejecuta la película.

Para utilizar el método `attachMovie`, al clip de película que se va a anexar se le debe asignar un nombre único en el cuadro de diálogo Propiedades de vínculos de símbolos.

Para asignar un clip de película para que se comparta

1. Seleccione el clip de película que desea anexar en la biblioteca de la película.
2. En la ventana Biblioteca, elija Vinculación en el menú Opciones.
3. En la sección Vínculo, elija Exportar este símbolo.
4. En el cuadro de diálogo Propiedades de vínculos de símbolos, en la sección Identificador, introduzca un nombre para el clip de película. El nombre debe ser distinto del nombre del símbolo en la biblioteca.
5. Haga clic en Aceptar.

Para anexar un clip de película a otro clip de película:

1. En el panel Acciones, especifique el destino al que desea anexar un clip de película.
2. En la lista de la Caja de herramientas, seleccione el objeto MovieClip y después elija el método `attachMovie`.
3. Establezca los siguientes argumentos:

- Para `idName`, especifique el nombre Identificador que introdujo en el cuadro de diálogo Propiedades de vínculos de símbolos.
- Para `newName`, introduzca un nombre de instancia para el clip anexado con el fin de que pueda ser capaz de asignarle un destino.
- Para `depth`, introduzca el nivel en el que se anexará la película duplicada al clip de película. Cada película anexada tiene su propio orden de apilamiento, con el nivel 0 como el nivel de la película que lo originó Los clips de película anexados siempre están encima del clip de película original.

  Por ejemplo:

```
myMovieClip.attachMovie("calif", "california", 10 );
```

Creación de clips inteligentes

Un Clip inteligente es un clip de película con parámetros de clip definidos que pueden cambiarse. Esos parámetros entonces se pasan a acciones en el Clip inteligente que cambia el comportamiento del clip.

Parea crear un Clip inteligente, asigne parámetros de clip a un símbolo de clip de película en la Biblioteca. Puede escribir sentencias de ActionScript en el Clip inteligente que operan en los parámetros del clip, de forma similar a utilizar argumentos en la definición de una función. Puede seleccionar una instancia del Clip inteligente en el Escenario y cambiar los valores de los parámetros en el panel Clip inteligente. Durante la reproducción, los valores establecidos en el panel se envían al Clip inteligente antes de que se ejecute cualquiera de las acciones de la película.

Los Clips inteligentes son útiles para pasar elementos complejos de Flash de un programador a un diseñador. El programador puede escribir las acciones en el Clip inteligente con variables que controlan el clip y la película. Entonces un diseñador puede cambiar los valores de esas variables en el panel Parámetros de clip sin tener que abrir el panel Acciones.

Puede utilizar los Clips inteligentes para crear elementos de interfaz (como botones de radio, menús emergentes, consejos de herramientas, encuestas. juegos y avatares). Cualquier clip de película que desee reutilizar de modo diferente sin tener que cambiar los scripts sería un buen Clip inteligente.

Además, puede crear una interfaz personalizada en Flash desde el panel Parámetros de clip para facilitar el trabajo a los diseñadores que están personalizando el clip.

Definición de los parámetros de clip

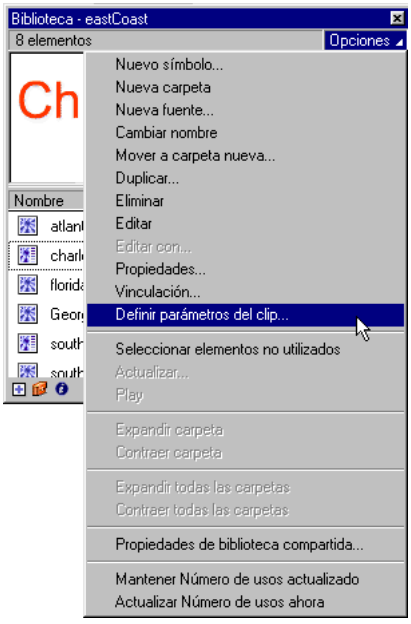
Los parámetros de clip son datos que se pasan a un clip de película cuando se carga en una película. Puede definir los parámetros de clip cuando está en el proceso de autoría de la película. Puede utilizar estos parámetros en acciones para cambiar el aspecto y el comportamiento del Clip inteligente mientras se reproduce la película. Un clip de película con parámetros de clip definidos se identifica a través de un icono especial en la ventana Biblioteca.

Para definir los parámetros de clip para un clip de película:

1. Seleccione el símbolo de un clip de película en su biblioteca de película y realice uno de los pasos siguientes para ver el cuadro de diálogo Parámetros de clip:

- Haga clic con el botón derecho del ratón (en Windows), o Control-Clic (en Macintosh), y elija Definir parámetros de clip en el menú contextual.

- Elija Definir parámetros del clip en el menú Opciones en la parte superior derecha de la ventana Biblioteca.



2 Utilice los controles del cuadro de diálogo Parámetros de clip como se detalla a continuación:

- Haga clic sobre el botón Agregar (+) para añadir un nuevo nombre/valor o parámetros adicionales para un par nombre/valor.
- Haga clic sobre el botón Menos (-) para borrar un par nombre/valor.
- Utilice los botones de flecha para cambiar el orden de los parámetros de la lista.
- Seleccione un campo haciendo doble clic sobre él e introduzca un valor.

3 En la sección Nombre, introduzca un identificador único para el parámetro.

4 En la sección Tipo, elija la clase de datos que contendrá el parámetro en el menú emergente:

- Seleccione Predeterminado para utilizar una cadena o un valor numérico.
- Seleccione Matriz para obtener una lista de elementos dinámica que puede aumentar o decrecer.
- Seleccione Objeto para declarar varios elementos relacionados con nombres y valores, como un objeto de punto con elementos *x* e *y*.
- Seleccione Lista para limitar la selección a varias elecciones, como `true` o `false` o `Red`, `Green` o `Blue`.

5 En la sección Valor, elija el valor predeterminado que contendrá el parámetro en el menú emergente.

6 Si desea utilizar una interfaz personalizada para el panel Parámetros de clip, realice uno de los pasos que se detallan a continuación:

- Introduzca una ruta relativa al archivo SWF de la interfaz personalizada en el campo Vínculo a IU personalizada.
- Haga clic sobre la carpeta Vínculo a IU personalizada y examine la interfaz personalizada.

Consulte la sección “Creación de una interfaz personalizada” a pagina 136.

7 Para Descripción, introduzca notas que aparecerán en el panel Parámetros de clip que describen lo que hace cada parámetro.

Puede incluir cualquier información en la Descripción que desee comunicar a quien utilice el Clip inteligente. Por ejemplo, una explicación de los métodos que ha definido.

8 Elija Bloquear en instancia para evitar que los usuarios cambien el nombre del parámetro en el panel Parámetros de clip.

Se le recomienda que deje bloqueados los nombres de parámetro.

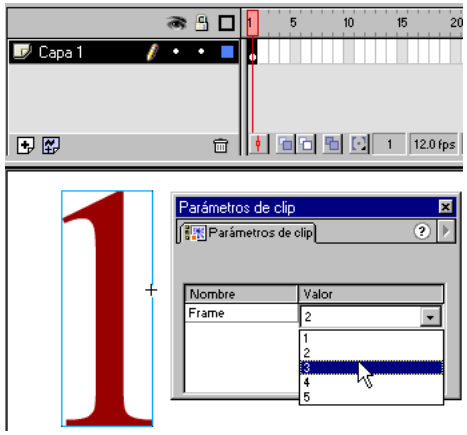
9 Haga clic en Aceptar.

Establecimiento de los parámetros de clip

Puede escribir acciones en el Clip inteligente que utilicen los parámetros definidos para cambiar el comportamiento de un Clip inteligente. Como ejemplo sencillo, si define un parámetro de clip con el nombre `Frame`, podría escribir el script siguiente en el Clip inteligente que utilice el parámetro `Frame`:

```
onClipEvent(load){
   gotoAndStop(Frame);
}
```

Entonces puede seleccionar el Clip inteligente en el Escenario y establecer el valor del parámetro `Frame` en el panel Parámetros de clip para cambiar que fotograma se reproduce.



Para establecer los parámetros de clip de un Clip inteligente:

1. Seleccione una instancia del Clip inteligente en el Escenario.

   Los Clips inteligentes son clips de película, así que sólo se mostrará el primer fotograma en el modo de autor.

2. Elija Ventana > Paneles > Parámetros de clip para ver el panel Parámetros de clip.

3. En el panel Parámetros de clip, realice uno de los pasos que se detallan a continuación:

- Haga doble clic sobre el campo Valor para seleccionarlo e introduzca un valor para cada parámetro.

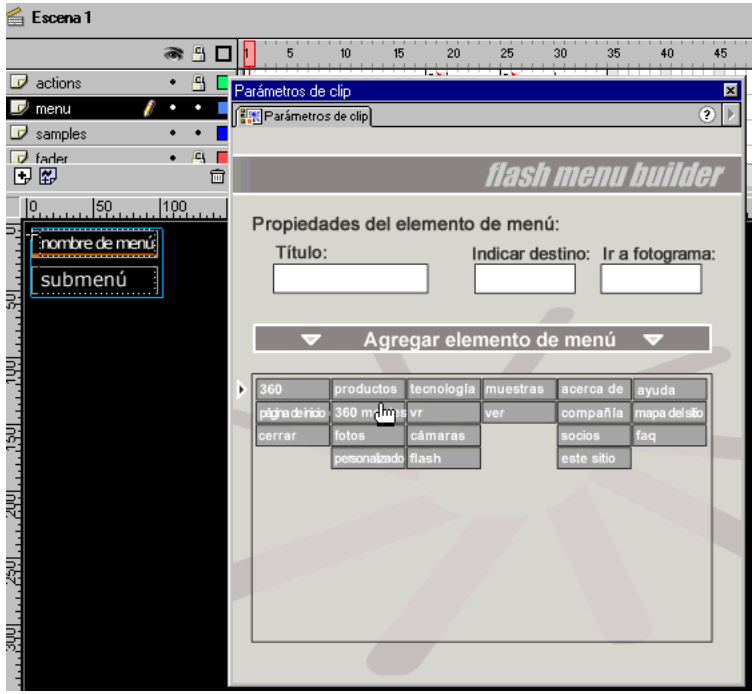
  Si se ha definido el parámetro como Lista, aparecerá un menú emergente.

- Si se ha definido una interfaz personalizada, utilice los elementos de interfaz que se proporcionan.

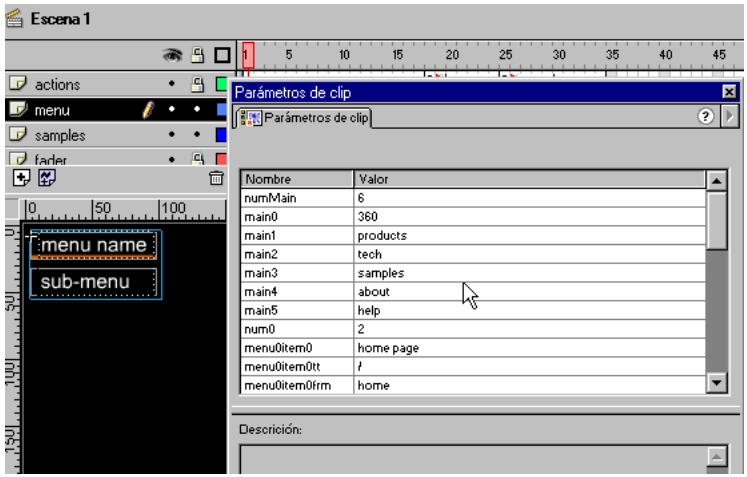
4. Elija Control > Probar película para ver cambio de comportamiento del Clip inteligente.

Creación de una interfaz personalizada

Una interfaz personalizada es una película de Flash que le permite introducir valores que se pasarán al Clip inteligente. La interfaz personalizada sustituye a la interfaz del panel Parámetros de clip.



El panel Parámetros de clip con una película de interfaz personalizada.



El mismo Clip inteligente sin una interfaz personalizada en el panel Parámetros de clip.

Cualquiera de los valores que introduzca mediante una interfaz personalizada se pasan del panel Parámetros de clip al Clip inteligente por medio de un intermediario o un clip de película de intercambio en la interfaz personalizada. El clip de película de intercambio debe tener el nombre de instancia `xch`. Si está seleccionada una interfaz personalizada en el cuadro de diálogo Definir parámetros de clip, la instancia del Clip inteligente pasa los parámetros definidos al clip de película`xch` y cualquier nuevo valor introducido en la interfaz personalizada se copia en `xch` y se pasa de vuelta al Clip inteligente.

Debe colocar el clip `xch` en la Línea de tiempo principal de la película de interfaz y `xch` siempre debe estar cargado. El clip de película `xch` debería contener solamente los valores que se van a pasar al Clip inteligente. No debería contener gráficos, otros clips de película o sentencias de ActionScript; `xch` simplemente es un contenedor por medio del cual se pasan los valores. Puede transferir objetos de nivel superior, como Matrices y Objetos, por medio del clip `xch`. Sin embargo, no debería pasar Matrices u Objetos anidados.

Para crear una interfaz personalizada para un Clip inteligente:

1. Elija Archivo > Nuevo para crear una nueva película de Flash.
2. Elija Insertar > Nuevo Símbolo para crear el clip de película de intercambio.
3. Cree una nueva capa llamada “Clip de intercambio”.
4. Con la capa “Clip de intercambio” seleccionada, arrastre el clip de película de intercambio desde la ventana Biblioteca hasta el Escenario en el fotograma 1.
5. Seleccione el clip de película de intercambio en el Escenario, elija Ventana > Paneles > Instancia e introduzca el nombre `xch`.
6. Cree los elementos de interfaz con los que el autor interactuará para establecer los parámetros de clip. Por ejemplo, un menú emergente, botones de radio o elementos de menú de arrastrar y soltar.
7. Utilice la acción `set variable` para copiar valores de variable y de objeto en la instancia `xch`.

   Por ejemplo, si un botón se utiliza como un elemento de interfaz, el botón podría tener una acción que establece el valor de la variable `vertical` y la pasa a `xch`, como se muestra a continuación:

   ```
   on (release){
       _root.xch.vertical = true;
   }
   ```

8. Exportar la película como un archivo SWF.

Para utilizar un archivo SWF de interfaz personalizado con un Clip inteligente, necesita vincularlos en el cuadro de diálogo Definir parámetros de clip en la Biblioteca que contiene el Clip inteligente. Es buena idea guardar el archivo SWF en el mismo directorio que el FLA que contiene el Clip inteligente. Si vuelve a utilizar el Clip inteligente en otro archivo o pasa el Clip inteligente a otro autor, el Clip inteligente y el SWF de interfaz personalizada deberán permanecer en las mismas ubicaciones relativas.

CAPÍTULO 5
Integración de Flash con aplicaciones Web

Las películas de Flash pueden enviar información a archivos remotos y también cargarla de éstos. Para enviar y cargar variables, utilice la acción `loadVariables` o `getURL`. Para cargar una película de Flash Player desde una ubicación remota, utilice la acción `loadMovie`. Para enviar y cargar datos XML, utilice el objeto XML o el objeto XMLsocket. Puede estructurar los datos XML utilizando los métodos de objeto XML predefinidos.

También puede crear formularios de Flash compuestos por elementos de interfaz comunes como campos de texto y menús emergentes para la captura de datos que se enviarán a una aplicación de servidor.

Para ampliar Flash de modo que pueda enviar y recibir mensajes desde el entorno del anfitrión de la película (por ejemplo, Flash Player o una función de JavaScript en un navegador Web) puede utilizar `fscommand` y los métodos de Flash Player.

Envío y carga de variables a y desde un archivo remoto

Una película de Flash es una ventana para capturar y visualizar información parecida a una página HTML. Las películas de Flash, a diferencia de las páginas HTML, pueden permanecer cargadas en el navegador y actualizarse continuamente con nueva información sin necesidad de refrescarlas. Puede utilizar las acciones de Flash y los métodos de objeto para enviar y recibir información de scripts, archivos de texto y archivos XML de servidor.

Los scripts de servidor pueden solicitar información de una base de datos y enviarla varias veces entre la base de datos y una película de Flash. Los scripts de servidor pueden estar escritos en muchos lenguajes diferentes: algunos de los más comunes son Perl, ASP (Microsoft Active Server Pages) y PHP.

Almacenar información en una base de datos y recuperarla le permite crear contenido dinámico y personalizado para su película. Por ejemplo, podría crear un tablero de mensajes, perfiles personales de los usuarios o un carrito de la compra que recuerde lo que ha adquirido el usuario de modo que se puedan determinar sus preferencias.

Puede utilizar varias acciones de ActionScript y métodos de objeto para pasar información dentro y fuera de la película. Cada acción y cada método utiliza un protocolo para transferir la información. Cada uno de ellos también requiere que se le dé formato a la información de alguna manera.

Las acciones siguientes utilizan el protocolo HTTP o HTTPS para enviar información en el formato codificado de URL: `getURL`, `loadVariables`, `loadMovie`.

Los métodos siguientes utilizan el protocolo HTTP o HTTPS para enviar información como XML: `XML.send`, `XML.load`, `XML.sendAndLoad`.

Los métodos siguientes crean y utilizan una conexión socket TCP/IP para enviar información como XML: `XMLSocket.connect`, `XMLSocket.send`.

Aspectos relativos a la seguridad

Cuando se reproduce una película de Flash en un navegador Web, puede cargar datos en la película solamente desde un archivo que se encuentre en un servidor en el mismo subdominio. Esto evita que las películas de Flash puedan descargar información de los servidores de otras personas.

Para determinar el subdominio de una URL que consta de uno o dos componentes, utilice el dominio completo:

Dominio	Subdominio
http://macromedia	macromedia
http://macromedia.com	macromedia.com

Para determinar el subdominio de una URL que consta de más de dos componentes, elimine el último nivel:

Dominio	Subdominio
http://x.y.macromedia.com	y.macromedia.com
http://www.macromedia.com	macromedia.com

El siguiente gráfico muestra cómo determina si Flash Player permite una solicitud HTTP:

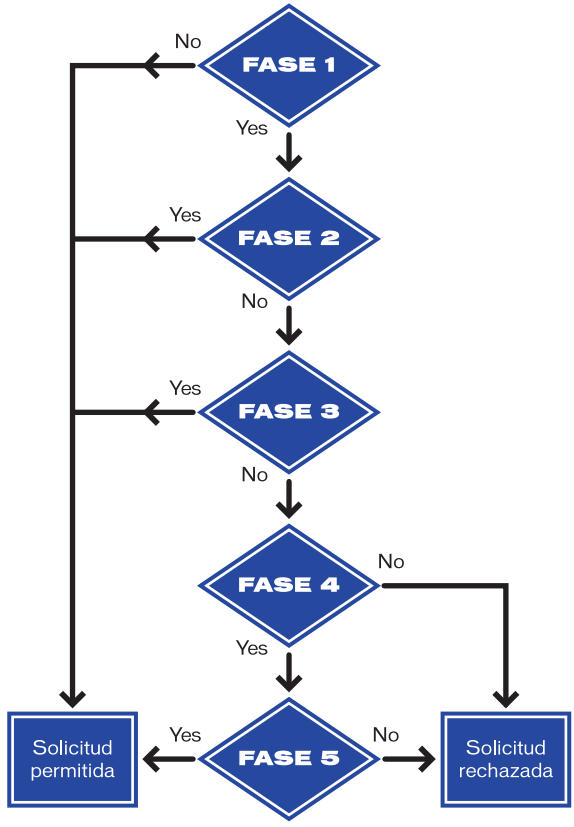
FASE 1
¿Se solicita para: loadVariables, xml.load, xml.sendAndLoad o xmlsocket.connect?

FASE 2
¿Se solicita para una dirección URL relativa?

FASE 3
¿Se ha cargado la película que realiza la solicitud desde un disco local? (su URL empieza por file: o res:)

FASE 4
¿Empieza la dirección URL solicitada por http://, https:// o ftp://?

FASE 5
¿Coincide el nombre de dominio de la película que realiza la solicitud con el de la URL solicitada?



Cuando utilice el objeto XMLSocket para crear una conexión socket con un servidor, debe utilizar un puerto con el número 1024 o superior. (Los puertos con números inferiores se usan comunmente para Telnet, FTP, la World Wide Web o Finger).

Flash se basa en las funciones de seguridad HTTP y HTTPS y de los navegadores estándar. En esencia, Flash ofrece el mismo nivel de seguridad que en un HTML estándar. Siga las mismas reglas que sigue al construir sitios Web HTML seguros. Por ejemplo, si desea utilizar contraseñas seguras en Flash, es necesario establecer la autenticación de la contraseña mediante una solicitud a un servidor Web.

Para crear una contraseña, utilice un campo de texto para solicitar que el usuario introduzca una contraseña. Envíela a un servidor en una acción `loadVariables` o en un método `XML.sendAndLoad` utilizando una URL HTTPS con el método `POST`. El servidor Web puede verificar si la contraseña es válida. De este modo, la contraseña nunca estará disponible en el archivo SWF.

Comprobación de los datos cargados

Cada acción y cada método que carga datos en una película (excepto `XMLSocket.send`) es *asíncrono*; los resultados de una acción se devuelven en un tiempo no determinado.

Antes de que pueda utilizar los datos cargados en una película, debe comprobar que se han cargado. Por ejemplo, no puede cargar variables y manipular los valores de esas variables en el mismo script. En el script siguiente, puede utilizar la variable `lastFrameVisited` hasta que esté seguro de que la variable se ha cargado desde el archivo `myData.txt`:

```
loadVariables("myData.txt", 0);
gotoAndPlay(lastFrameVisited);
```

Cada acción y método tiene una técnica específica que puede utilizar para comprobar que se han cargado los datos. Si utiliza las acciones `loadVariables` o `loadMovie` puede cargar información en un destino del clip de película y utilizar el evento `data` de la acción `onClipEvent` para ejecutar un script. Si utiliza la acción `loadVariables` para cargar los datos, la acción `onClipEvent(data)` se ejecutará al cargarse la última variable. Si utiliza la acción `loadMovie` para cargar los datos, la acción `onClipEvent(data)` se ejecuta cada vez que se envía un fragmento de la película a Flash Player.

Por ejemplo, la siguiente acción de botón carga las variables del archivo `myData.txt` en el clip de película `loadTargetMC`:

```
on(release){
   loadVariables("myData.txt", _root.loadTargetMC);
}
```

Una acción asignada a la instancia `loadTargetMC` utiliza la variable `lastFrameVisited` que se carga del archivo `myData.txt`. La siguiente acción se ejecutará solamente cuando se hayan cargado todas las variables, incluida `lastFrameVisited`:

```
onClipEvent(data) {
   goToAndPlay(lastFrameVisited);
}
```

Si utiliza los métodos `XML.load` y `XMLSocket.connect`, puede definir un controlador que procesará los datos cuando lleguen. Un controlador es una propiedad del objeto XML o XMLSocket a la que se le asigna una función que ha definido el usuario. Se llama automáticamente a los controladores cuando se recibe la información. Para el objeto XML, utilice `XML.onLoad`. Para el objeto XMLSocket, utilice `XMLSocket.onConnect`.

Si desea obtener más información, consulte las secciones "Utilización del objeto XML" a pagina 146 y "Utilización del objeto XMLSocket" a pagina 150.

Utilización de loadVariables, getURL y loadMovie

Todas las acciones `loadVariables`, `getURL` y `loadMovie` se comunican con los scripts de servidor utilizando el protocolo HTTP. Cada acción envía todas las variables de la Línea de tiempo a la que está vinculada la acción; cada acción gestiona su respuesta como se muestra a continuación:

- `getURL` devuelve cualquier información a una ventana del navegador, no al Flash Player.
- `loadVariables` carga variables en una Línea de tiempo específica en Flash Player.
- `LoadMovie` carga una película en un nivel específico en Flash Player.

Cuando utilice las acciones `loadVariables`, `getURL` o `loadMovie`, puede especificar varios argumentos:

- *`URL`* es el archivo en el que residen las variables remotas.
- *`Location`* es el nivel o destino en la película que recibe las variables.

  Si desea obtener más información sobre los niveles y los destinos, consulte la sección "Líneas de tiempo múltiples" a pagina 110

  Nota: La acción `getURL` no toma este argumento.

- *`Variables`* establece el método HTTP, ya sea con `GET` o con `POST`, por medio de los que se enviarán las variables.

Por ejemplo, si deseara realizar el seguimiento de las puntuaciones más altas de un juego podría almacenar las puntuaciones en un servidor y utilizar una acción `loadVariables` para cargarlas en la película cada vez que alguien juegue una partida. La acción puede ser parecida a esta:

```
loadVariables("http://www.mySite.com/scripts/high_score.php",
_root.scoreClip, GET)
```

De esta forma se cargan las variables del script PHP llamado `high_score.php` en la instancia del clip de película `scoreClip` utilizando el método HTTP `GET`.

Cualquiera de las variables cargadas con la acción `loadVariables` deben encontrarse en el formato MIME estándar aplicación/x-www-formulario_URL_codificado (un formato estándar que se utiliza en los scripts CGI). El archivo que especifique en el argumento URL de la acción `loadVariables` debe escribir la variable y los pares de valores en este formato para que Flash pueda leerlos.

El archivo puede especificar cualquier número de variables; las variables y los pares de valores deben estar separados por un signo `&` y las palabras dentro de un valor deben estar separadas con un signo más (+). Por ejemplo, la siguiente frase define varias variables:

```
highScore1=54000&playerName1=rockin+good&highScore2=53455&playerN
ame2=bonehelmet&highScore3=42885&playerName3=soda+pop
```

Si desea obtener más información sobre `loadVariables`, `getURL` y `loadMovie`, consulte Capítulo 7, “Diccionario de ActionScript”.

Lenguaje XML

XML (*Extensible Markup Language*, Lenguaje de marca extensible) se está convirtiendo en el estándar para el intercambio de datos estructurados en las aplicaciones de Internet. Puede integrar datos en Flash con los servidores que utilizan tecnología XML para crear aplicaciones muy complejas como un sistema de chat o un sistema de corretaje.

En XML, como con HTML, puede utilizar etiquetas para *marcar*, o especificar, un cuerpo de texto. En HTML, puede utilizar las etiquetas predefinidas para indicar cómo debería aparecer el texto en un navegador Web (por ejemplo, la etiqueta `<b>` indica que el texto debería aparecer en negrita). En XML, debe definir las etiquetas que identifican el tipo de una parte de datos (por ejemplo, `<password>VerySecret</password>`). XML separa la estructura de la información del modo en el que ésta se muestra. Esto permite que el mismo documento XML se utilice y reutilice en diferentes entornos.

Cada etiqueta XML se denomina *nodo*, o elemento. Cada nodo tiene un tipo (1–elemento XML, o 3–nodo de texto) y los elementos también pueden tener atributos. Un nodo anidado en otro nodo se denomina *secundario* o *Nodosecundario*. La estructura de árbol jerárquico de los nodos se llama XML DOM (Modelo de Objetos de Documento), parecido a JavaScript DOM, que es la estructura de los elementos de un navegador Web.

En el ejemplo siguiente, `<PORTFOLIO>` es el nodo principal, no tiene atributos y contiene el Nodosecundario `<HOLDING>` que tiene los atributos `SYMBOL`, `QTY`, `PRICE` y `VALUE`:

```
<PORTFOLIO>
   <HOLDING SYMBOL="RICH"
      QTY="75"
      PRICE="245.50"
      VALUE="18412.50" />
</PORTFOLIO>
```

Utilización del objeto XML

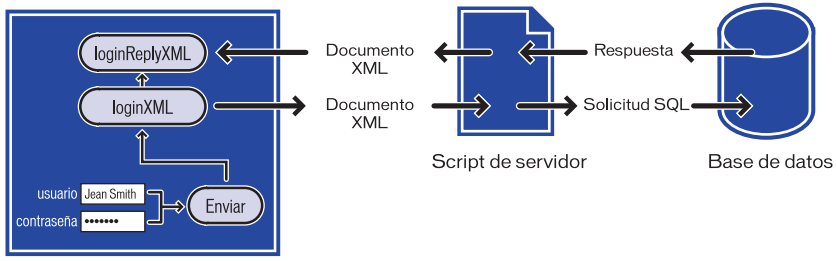
Puede utilizar los métodos del objeto XML de ActionScript (por ejemplo, `appendChild`, `removeNode`, y `insertBefore`) para estructurar los datos XML en Flash para enviarlos a un servidor y para manipular e interpretar los datos XML descargados.

Puede utilizar los métodos de objeto XML siguientes para enviar y cargar datos XML a un servidor por medio del método HTTP `POST`:

- `load` descarga XML de una URL y lo coloca en un objeto XML de ActionScript.
- `send` pasa un objeto XML a una URL. Cualquier información devuelta se envía a otra ventana del navegador.
- `sendAndLoad` envía un objeto XML a una URL. Cualquier información devuelta se coloca en un objeto XML de ActionScript.

Por ejemplo, podría crear un sistema de corretaje para comerciar con títulos que almacene toda su información (nombres de usuario, contraseñas, IDs de sesión, valores de la cartera e información de transacciones) en una base de datos.

El script de servidor que pasa la información entre Flash y la base de datos lee y escribe los datos en formato XML. Puede utilizar ActionScript para convertir la información recogida en la película de Flash (por ejemplo, un nombre de usuario y una contraseña) a un objeto XML y después enviar los datos al script de servidor como un documento XML. Puede utilizar ActionScript para cargar el documento XML que el servidor devuelve a un objeto XML para ser utilizado en la película.



El flujo y la conversión de datos entre una película de Flash Player, un documento de script de servidor y una base de datos.

La validación de la contraseña para el sistema de corretaje requiere dos scripts: una función definida en el fotograma uno y un script que crea y envía los objetos XML anexados al botón SUBMIT del formulario.

Cuando un usuario introduce su información en los campos de texto de la película de Flash con las variables `username` y `password`, las variables deben convertirse a XML antes de pasarlas al servidor. La primera sección del script carga las variables en un objeto XML recién creado llamado `loginXML`. Cuando un usuario presiona el botón SUBMIT, el objeto `loginXML` se convierte en una cadena de XML y se envía al servidor.

El script siguiente se encuentra anexado al botón SUBMIT. Para comprender el script, lea las líneas comentadas de cada script, indicadas por los caracteres `//`:

```
on (release) {
   // A. Construir un documento XML con un elemento LOGIN
   loginXML = new XML();
   loginElement = loginXML.createElement("LOGIN");
   loginElement.attributes.username = username;
   loginElement.attributes.password = password;
   loginXML.appendChild(loginElement);

   // B. Construir un documento XML para albergar la respuesta del
servidor
   loginReplyXML = new XML();
   loginReplyXML.onLoad = onLoginReply;

   // C. Enviar el elemento LOGIN al servidor,
   //    colocar la respuesta en loginReplyXML
   loginXML.sendAndLoad("https://www.imexstocks.com/main.cgi",
            loginReplyXML);
}
```

La primera sección del script genera el siguiente código XML cuando el usuario pulsa el botón SUBMIT:

```
<LOGIN USERNAME="JeanSmith" PASSWORD="VerySecret" />
```

El servidor recibe el XML, genera una respuesta XML y lo envía de regreso a la película Flash. Si se acepta la contraseña, el servidor responde con el mensaje que se muestra a continuación:

```
<LOGINREPLY STATUS="OK" SESSION="rnr6f7vkj2oe14m7jkkycilb" />
```

Este XML incluye un atributo `SESSION` que contiene un Id de sesión exclusivo generado al azar que se utilizará en todas las comunicaciones entre el cliente y el servidor para el resto de la sesión. Si se rechaza la contraseña, el servidor responde con el mensaje que se muestra a continuación:

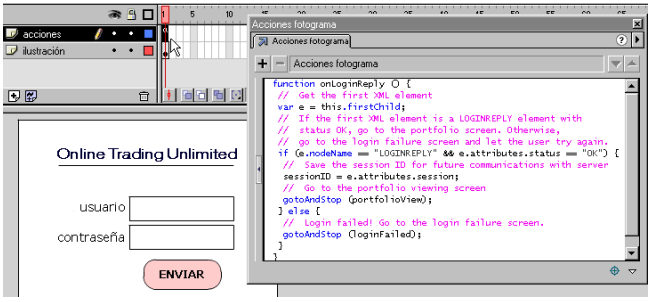
```
<LOGINREPLY STATUS="FAILED" />
```

El nodo XML `LOGINREPLY` debe cargarse en un objeto XML vacío en la película de Flash. La sentencia siguiente crea el objeto XML `loginreplyXML` para recibir el nodo XML:

```
// B. Construir un objeto XML para albergar la respuesta del
servidor
loginReplyXML = new XML();
loginReplyXML.onLoad = onLoginReply;
```

La segunda sentencia asigna la función `onLoginReply` al controlador `loginReplyXML.onLoad`.

El elemento XML `LOGINREPLY` llega de modo asíncrono, al igual que los datos de una acción `loadVariables`, y lo carga en el objeto `loginReplyXML`. Cuando llegan los datos, se llama al método `onLoad` del objeto `loginReplyXML`. Debe definir la función `onLoginReply` y asignarla al controlador `loginReplyXML.onLoad` para que pueda procesar el elemento `LOGINREPLY`. La función `onLoginReply` se asigna al fotograma que contiene el botón Submit.



La función `onLoginReply` está definida en el primer fotograma de la película.

La función `onLoginReply` está definida en el primer fotograma de la película. Para comprender el script, lea las líneas de comentario de cada script indicadas por los caracteres `//`:

```
function onLoginReply() {
   // Obtener el primer elemento XML
   var e = this.firstChild;
   // Si el primer elemento XML es un elemento LOGINREPLY con
   // estado OK, ir a la pantalla del portafolios.  De lo
contrario,
   // ir a la pantalla que indica fallo de conexión y dejar que
el usuario lo intente de nuevo.
   if (e.nodeName == "LOGINREPLY" && e.attributes.status == "OK") {
// Guardar el ID de la sesión para futuras comunicaciones con el
servidor
   sessionID = e.attributes.session;
// Ir a la pantalla de visualización del portafolios
      gotoAndStop("portfolioView");
   } else {
      // Conexión fallida  Ir a la pantalla de fallo de conexión.
      gotoAndStop("loginFailed");
   }
}
```

La primera línea de esta función , `var e = this.firstChild`, utiliza la palabra clave `this` para referirse al objeto XML `loginReplyXML` que acaba de cargarse con XML del servidor. Puede utilizar `this` debido a que `onLoginReply` ha sido invocada como `loginReplyXML.onLoad`, así que aunque `onLoginReply` parezca ser un función normal, en realidad se comporta como un método de `loginReplyXML`.

Para enviar el nombre de usuario y la contraseña como XML al servidor y para cargar una respuesta XML de nuevo en la película de Flash, puede utilizar el método `sendAndLoad`, como se muestra a continuación:

```
// C. Enviar el elemento LOGIN al servidor,
//    colocar la respuesta en loginReplyXML
   loginXML.sendAndLoad("https://www.imexstocks.com/main.cgi",
loginReplyXML);
```

Si desea obtener más información sobre los métodos XML, consulte sus entradas individuales en el Capítulo 7, “Diccionario de ActionScript”.

Nota: Este diseño es solamente un ejemplo y no garantizamos el nivel de seguridad que proporciona. Si está implementando un sistema de seguridad protegido mediante contraseña, asegúrese de que comprende perfectamente la seguridad de la red.

Utilización del objeto XMLSocket

ActionScript proporciona un objeto XMLSocket predefinido que le permite abrir una conexión continua con un servidor. Una conexión de socket hembra permite al servidor enviar la información al cliente tan pronto como la información se encuentra disponible. Sin una conexión continua, el servidor debe esperar una solicitud HTTP. Esta conexión abierta elimina los problemas de latencia y se usa con frecuencia para las aplicaciones en tiempo real como los chats. Los datos se envían por la conexión de socket como una cadena y deberán estar en formato XML. Puede utilizar el objeto XML para estructurar los datos.

Para crear una conexión de socket, debe crear una aplicación de servidor que espere la conexión de socket y envíe una respuesta a la película de Flash. Este tipo de aplicación de servidor puede escribirse en un lenguaje de programación como Java.

Puede utilizar los métodos del objeto XMLSocket de ActionScript `connect` y `send` para transferir XML desde y hasta un servidor por una conexión de socket. El método `connect` establece una conexión de socket con un puerto de servidor Web. El método `send` pasa un objeto XML al servidor especificado en la conexión de socket.

Cuando se invoca al método del objeto XMLSocket `connect`, Flash Player abre una conexión TCP/IP con el servidor y la mantiene abierta hasta que sucede una de las condiciones que se muestran a continuación:

- Se llama al método `close` del objeto XMLSocket.
- No existen más referencias al objeto XMLSocket.
- Se sale de Flash Player.
- Se interrumpe la conexión (por ejemplo, se desconecta el módem).

El ejemplo siguiente crea una conexión de socket XML y envía los datos desde el objeto XML `myXML`. Para comprender el script, lea las líneas comentadas de cada script, indicadas por los caracteres `//`:

```
// crear un nuevo objeto Date
sock = new XMLSocket();
//llamar a su método de conexión para establecer una conexión con
el puerto 1024
//del servidor en al URL
sock.connect("http://www.myserver.com", 1024);
//definir una función para asignarla al objeto sock que gestiona
//la respuesta del servidor. Si la conexión tiene éxito, enviar el
//objeto myXML. Si falla, muestra un mensaje de error en un campo
de texto.
function onSockConnect(success){
   if (success){
      sock.send(myXML);
   } else {
      msg="Ha ocurrido un error al conectar con "+serverName;
   }
}
//asignar la función SockConnect a la propiedad onConnect
sock.onConnect = onSockConnect;
```

Si desea obtener más información, consulte la entrada correspondiente a XMLSocket en el Capítulo 7, “Diccionario de ActionScript”.

Creación de formularios

Los formularios de Flash proporcionan un método avanzado de interactividad: una combinación de botones, películas y campos de texto que le permiten pasar información a otra aplicación situada en un servidor local o remoto. Todos los elementos comunes de un formulario (como botones de radio, listas desplegables y casillas de verificación) pueden crearse como películas o botones con el mismo aspecto y estilo del diseño global de su sitio Web. El elemento más común de un formulario es un campo de introducción de texto.

Los tipos de formulario más habituales que utilizan dichos elementos de interfaz incluyen interfaces de chat, notas de pedidos e interfaces de búsqueda. Por ejemplo, un formulario de Flash puede recoger y enviar datos de direcciones a otra aplicación que compila la información en un archivo de correo electrónico o de base de datos. Incluso un solo campo de texto se considera un formulario y puede utilizarse para recoger la información introducida por el usuario y mostrar los resultados.

Los formularios requieren dos componentes principales: los elementos del interfaz de Flash que componen el formulario y una aplicación de servidor o un script de cliente para procesar la información introducida por el usuario. En los siguientes pasos se esboza el procedimiento general para la creación de elementos de formulario con Flash.

Para crear un formulario:

1. Coloque los elementos del interfaz en la película utilizando el diseño que desee.

   Puede utilizar los elementos de interfaz de la biblioteca común Botones-Avanzados o crear los suyos propios.

2. En el Panel opciones de texto, establezca los campos de texto en Entrada y asigne un nombre de variable exclusivo.

   Si desea obtener más información acerca de la creación de campos de texto editables, consulte la sección *Utilización de Flash*.

3. Asigne una acción que envíe o cargue, o bien envíe y cargue los datos.

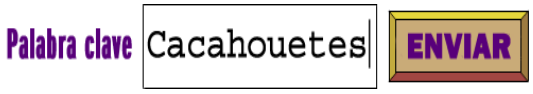
Creación de un formulario de búsqueda

Un ejemplo de un formulario sencillo es un campo de búsqueda con un botón Submit. Como introducción a la creación de formularios, el ejemplo siguiente proporciona instrucciones para la creación de una interfaz de búsqueda utilizando una acción `getURL`. Al introducir la información necesaria, los usuarios pueden pasar una palabra clave a una herramienta de búsqueda situada en un servidor Web remoto.

Para crear un formulario de búsqueda sencillo:

1. Cree un botón para enviar los datos introducidos.
2. Cree una etiqueta, un campo de texto vacío y una instancia del botón en el Escenario.

   La pantalla debería tener un aspecto similar a este:



3. Seleccione el campo de texto y elija Ventana > Paneles > Opciones de texto.
4. En el panel Opciones de texto, establezca las siguientes opciones:

- Elija Introducción de texto en el menú emergente.
- Seleccione Borde/Fondo.
- Especifique un nombre de variable.

   Nota: Las herramientas de búsqueda individuales pueden requerir un nombre de variable específico. Vaya al sitio Web de la herramienta de búsqueda para obtener más detalles.

5 En el Escenario, seleccione el botón y elija Ventana > Acciones.

Aparecerá el panel Acciones de objeto.

Nota: Una marca de verificación junto a Acciones en el menú Ventana indica que el panel está abierto.

6 Arrastre la acción `getURL` desde la caja de herramientas hasta la ventana Script.

7 En el panel Parámetros, establezca las siguientes opciones:

- En URL, introduzca la URL de la herramienta de búsqueda.
- En Ventana, seleccione `_blank`. Esto abrirá una nueva ventana que mostrará los resultados de la búsqueda.
- En Variables, seleccione Send Using GET.

8 Para comprobar el funcionamiento del formulario, elija Archivo > Previsualización de publicación > HTML.

Utilización de variables en formularios

Puede utilizar variables en un formulario para almacenar la información introducida por el usuario. Para establecer variables, utilice los campos de texto editables o asigne acciones a botones en elementos de interfaz. Por ejemplo, cada elemento en un menú emergente es un botón con una acción que establece una variable para indicar el elemento seleccionado. Puede asignar un nombre de variable a un campo de introducción de texto. El campo de texto actúa como una ventana que muestra el valor de esa variable.

Cuando pasa información desde y hasta un script de servidor, las variables de la película de Flash deben coincidir con las variables del script. Por ejemplo, si el script espera una variable llamada `password`, el campo de texto en el que los usuarios introducen la contraseña deberá tener el nombre de variable `password`.

Algunos scripts requieren variables ocultas, que son las variables que el usuario nunca ve. Para crear una variable oculta en Flash, puede establecer una variable en un fotograma en el clip de película que contenga los otros elementos del formulario. Las variables ocultas se envían al script de servidor junto con otras variables establecidas en la Línea de tiempo que contiene la acción que realiza la acción de enviar los datos en el formulario.

Verificación de datos introducidos

En un formulario que pasa las variables a una aplicación de un servidor Web, deseará verificar que los usuarios introducen la información correcta. Por ejemplo, para evitar que los usuarios introduzcan texto en un campo del número de teléfono. Utilice una serie de acciones `set variable` junto con `for` y `if` para evaluar los datos introducidos.

En la siguiente acción de ejemplo comprueba si los datos introducidos son un número y que el número esté en el formato *###-###-####*. Si los datos son válidos, aparece el mensaje "Correcto, número de teléfono válido". Si los datos no son válidos, aparece el mensaje "Número de teléfono no válido".

Para utilizar este script en una película, cree dos campos de texto en el Escenario y elija Entrada en el panel Opciones de texto para cada uno. Asigne la variable `phoneNumber` a uno de los campos de texto y asigne la variable `message` al otro. Anexe la siguiente acción a un botón en el Escenario al lado de los campos de texto:

```
on (release) {
   valid = validPhoneNumber(phoneNumber);
   if (valid) {
      mensaje = "Correcto, número de teléfono válido";
   } else {
      mensaje = "Número de teléfono no válido";
   }
   function isdigit(ch) {
      return ch.length == 1 && ch >= '0' && ch <= '9';
   }
   function validPhoneNumber(phoneNumber) {
      if (phoneNumber.length != 12) {
         return false;
      }
      for (var index = 0; index < 12; index++) {
         var ch = phoneNumber.charAt(index);
         if (index == 3 || index == 7) {
            if (ch != "-") {
            return false;
            }
         } else if (!isdigit(ch)) {
            return false;
         }
      }
      return true;
   }
}
```

Para enviar los datos, cree un botón con una acción similar a la siguiente. (Sustituya los argumentos `getURL` con argumentos apropiados para su película.)

```
on (release) {
   if (valid) {
      getURL("http://www.webserver.com", "_self", "GET");
   }
}
```

Para obtener más información sobre estas sentencias de ActionScript, consulte `set`, `for` y `if` en el Capítulo 7, “Diccionario de ActionScript” a pagina 175”.

Envío de mensajes desde y hasta el Reproductor de Flash

Para enviar mensajes desde una película de Flash a su entorno anfitrión (por ejemplo, un explorador Web, una película de Director o al reproductor Flash Player independiente), puede utilizar la acción `fscommand`. Esto le permite ampliar su película utilizando la capacidades del anfitrión. Por ejemplo, podría pasar una acción `fscommand` a una función de JavaScript en una página HTML que abra un nueva ventana del navegador con propiedades específicas.

Para controlar una película en Flash Player desde los lenguajes de script de navegador Web como JavaScript, VBScript y Microsoft JScript, puede utilizar los métodos de Flash Player (funciones que envían mensajes desde un entorno anfitrión a la película de Flash). Por ejemplo, podría tener un vínculo a una página HTML que envíe su película de Flash a un fotograma específico.

Utilización de fscommand

Utilice la acción `fscommand` para enviar un mensaje a cualquiera que sea el programa anfitrión de Flash Player. La acción `fscommand` tiene dos parámetros: *`command`* y *`arguments`*. Para enviar un mensaje a la versión independiente de Flash Player, debe utilizar comandos y argumentos predefinidos. Por ejemplo, la acción siguiente establece que el reproductor independiente cambie la escala de la película al tamaño de pantalla completa cuando se suelte un botón:

```
on(release){
   fscommand("fullscreen", "true");
}
```

La tabla siguiente muestra los valores que puede especificar para los parámetros *command* y *arguments* de la acción `fscommand` para controlar una película que se ejecuta en el reproductor independiente (incluidos los proyectores):

comando	*argumentos*	**Propósito**
`salir`	Ninguno	Cierra el proyector.
`fullscreen`	`true` o `false`	Especificar `true` establece Flash Player en el modo de pantalla completa. Especificar `false` devuelve el reproductor a la vista de menú normal.
`allowscale`	`true` o `false`	Si se especifica `false` el reproductor se establece para que la película se dibuje siempre a su tamaño original y nunca se cambie su escala. Si se especifica `true` obliga a la película a cambiar su escala al 100% del reproductor.
`showmenu`	`true` o `false`	Especificar `true` habilita el conjunto completo de elementos de menú de contexto. Si se especifica `false` se atenúan todos los elementos de menú de contexto excepto Acerca de Flash Player.
`exec`	Ruta de acceso a la aplicación	Ejecuta una aplicación desde el interior del proyector.

Para utilizar `fscommand` para enviar un mensaje a un lenguaje de script como JavaScript en un navegador Web, puede pasar cualquiera de los dos argumentos de los parámetros *command* y *arguments*. Estos argumentos pueden ser cadenas o expresiones y se utilizarán en una función de JavaScript que "captura" o gestiona la acción `fscommand`.

Una acción `fscommand` invoca a la función de JavaScript *moviename*`_DoFSCommand` en la página HTML en la que está insertada la película de Flash, donde *moviename* es el nombre de Flash Player según fue asignado por el atributo `NAME` del identificador `EMBED` o por el atributo `ID` del identificador `OBJECT`. Si Flash Player tiene asignado el nombre `myMovie`, la función de JavaScript invocada es `myMovie_DoFSCommand`.

Para utilizar la acción fscommand para abrir un cuadro de mensaje desde una película de Flash en la página HTML a través de JavaScript:

1 En la página HTML que incorpora la película Flash, agregue el siguiente código de Javascript:

```
function theMovie_DoFSCommand(command, args) {
    if (command == "messagebox") {
       alert(args);
    }
}
```

Si publica su película utilizando Flash con la plantilla FSCommand en Configuración de publicación en HTML, el código se inserta automáticamente. Los atributos `NAME` y `ID` serán el nombre del archivo. Por ejemplo, para el archivo `myMovie.fla`, los atributos se establecerían en `myMovie`. Si desea obtener más información sobre la publicación, consulte *Utilización de Flash.*

2 En la película de Flash, agregue la acción `fscommand` a un botón:

```
fscommand("messagebox", "Es un cuadro de mensajes invocado desde
Flash.")
```

También puede utilizar expresiones para la acción `fscommand` y argumentos, como en el ejemplo siguiente:

```
fscommand("messagebox", "Hello, " & name & ", welcome to
our Web site!")
```

3 Elija Archivo > Previsualización de la publicación > HTML para probar la película.

La acción `fscommand` puede enviar mensajes a Macromedia Director que son interpretados por Lingo como cadenas, eventos o código Lingo ejecutable. Si el mensaje es una cadena o un evento, debe escribir el código Lingo para recibirlo de la acción `fscommand` y llevar a cabo una acción en Director. Si desea obtener más información, consulte el Centro de Soporte de Director en la dirección http://www.macromedia.com/support/director.

En Visual Basic, Visual C++, y otros programas que son anfitriones para los controles ActiveX, `fscommand` envía un evento VB con dos cadenas que pueden gestionarse en el lenguaje de programación del entorno. Si desea obtener más información, utilice las palabras clave `Flash method` para buscar el Centro de Soporte Flash en la dirección http://www.macromedia.com/support/flash.

Acerca de los métodos de Flash Player

Puede utilizar los métodos de Flash Player para controlar una película en Flash Player mediante los lenguajes de script de navegador Web como JavaScript y VBScript. Al igual que con otros métodos, puede utilizar los métodos de Flash Player para enviar llamadas a las películas de Flash Player desde un entorno de scripts diferente a ActionScript. Cada método tiene un nombre y la mayoría de los métodos toman argumentos. Un argumento especifica un valor sobre el que opera el método. El cálculo realizado por algunos de los métodos devuelve un valor que puede ser utilizado por el entorno de scripts.

Existen dos tecnologías diferentes que permiten la comunicación entre el navegador y Flash Player. LiveConnect (Netscape Navigator 3.0 o posteriores bajo Windows 95/98/2000/NT o Power Macintosh) y ActiveX (Microsoft Internet Explorer 3.0 y posteriores bajo Windows 95/98/2000/NT). Aunque las técnicas de script son similares para todos los navegadores y lenguajes, existen otras propiedades adicionales y eventos disponibles para su utilización con los controles ActiveX.

Si desea obtener más información, incluida una lista completa de los métodos de script de Flash Player, utilice las palabras clave `Flash method` para buscar el Centro de Soporte de Flash en la dirección http://www.macromedia.com/support/flash.

CAPÍTULO 6
Solución de problemas de ActionScript

El grado de minuciosidad de algunas acciones, especialmente cuando se combinan con otras, puede hacer que las películas de Flash sean extraordinariamente complejas. Como en cualquier otro lenguaje de programación, el programador puede escribir código de ActionScript incorrecto que provoque errores en los scripts. La utilización de técnicas de creación apropiadas facilita la posterior solución de problemas en las películas cuando algo se comporta de modo inesperado.

Flash dispone de varias herramientas que ayudan a verificar las películas en modo de comprobación o mediante un navegador Web. El Depurador muestra una lista jerárquica de clips de película cargados actualmente en Flash Player. También permite mostrar y modificar los valores de las variables a medida que se reproduce la película. En el modo de comprobación de película, la ventana Salida muestra mensajes de error y listas de variables y objetos. Puede utilizarse la acción `trace` en los scripts para enviar notas de programación y resultados de expresiones a la ventana Salida.

Directrices para la creación y solución de problemas

La utilización de técnicas de creación adecuadas durante el proceso de programación de scripts, da como resultado películas con menor cantidad de errores de programación. Las siguientes directrices le ayudarán a evitar problemas y a solucionarlos rápidamente en caso de que se presenten.

Utilización de técnicas de creación adecuadas

Es aconsejable guardar varias versiones de la película a medida que se trabaja. Seleccione Archivo > Guardar como para guardar una versión con un nombre diferente cada media hora. Puede utilizar versiones anteriores para determinar a partir de qué momento empezó a surgir un determinado problema buscando el archivo más reciente que no producía dicho problema. Utilizando esta técnica, siempre podrá contar con una versión que funcione, incluso aunque algún archivo resulte dañado.

Otra regla importante durante la creación es probar pronto, probar a menudo y probar en todas las plataformas de destino, con el fin de detectar los problemas tan pronto como aparezcan. Seleccione Control > Probar película para ejecutar la película en modo de prueba siempre que realice cambios significativos o antes de guardar una versión. En el modo de prueba, la película se ejecuta en una versión del reproductor independiente.

Si la audiencia a la que va dirigida la película pretende verla desde la Web, será importante probarla también en un navegador. En determinadas circunstancias (por ejemplo, cuando se esté desarrollando una intranet) podrá conocerse de antemano el tipo de navegador y la plataforma que usará la audiencia. En cambio, cuando se desarrollan películas para un sitio Web, conviene probarlas en todos los navegadores y en todas las plataformas posibles.

Las siguientes técnicas de creación podrían ser de gran ayuda:

- Utilice la acción `trace` para enviar comentarios a la ventana Salida (Consulte “Utilización de la acción trace” a pagina 173).
- Utilice la acción `comment` para incluir notas aclaratorias que aparecerán únicamente en el panel Acciones (Consulte “Comentarios” a pagina 55).
- Emplee criterios coherentes de asignación de nombres para identificar elementos en un script. Por ejemplo, es conveniente evitar espacios en los nombres. Escriba la primera letra de los nombres de variables y funciones en minúsculas y escriba cada nueva palabra con su inicial en mayúsculas (`miNombreVariable`, `miNombreFuncion`). Escriba la primera letra de las funciones constructoras en mayúsculas (`MiFuncionConstructora`). Es importante adecuarse a un estilo que tenga sentido y utilizarlo de forma coherente.

- Utilice nombres de variables descriptivos que reflejen el tipo de información que contiene la variable. Por ejemplo, una variable que contenga información acerca de el último botón presionado podría llamarse `ultimoBotonPresionado`. Un nombre como `foo` haría difícil recordar el tipo de información que contiene la variable.
- Utilice campos de texto editables en capas guía para controlar los valores de las variables, como alternativa a la utilización del Depurador.
- Utilice el Explorador de películas en modo de edición de películas para ver la lista de visualización y todas las acciones de una película. Consulte Ayuda de Flash.
- Utilice la acción `for...in` para realizar un bucle continuo por las propiedades de los clips de película, incluidos los clips de película secundarios. Puede hacer uso de la acción `for...in` junto con la acción `trace` para enviar una lista de propiedades a la ventana Salida. Consulte "Repetición de una acción" a pagina 74.

Utilización de una lista de verificación

Al igual que sucede con otros entornos de programación de scripts, existen una serie de errores que los programadores cometen con cierta frecuencia. La siguiente lista constituye un buen punto de partida para solucionar posibles problemas de la película:

- Asegúrese de que se encuentra en el modo de prueba de película.

  En el modo de creación sólo funcionarán los botones y las acciones de fotograma simples (por ejemplo, `gotoAndPlay` y `stop`). Seleccione Control > Habilitar acciones de fotogramas simples, o bien Control > Habilitar botones simples para activar las citadas acciones.
- Asegúrese de que no existan acciones de fotograma en varias capas que entren en conflicto entre sí.
- Si utiliza el panel Acciones en Modo Normal, asegúrese de que las sentencias están preparadas para utilizar expresiones.

  Si intenta pasar una expresión en una acción sin seleccionar el cuadro Expresión, el valor se pasará como cadena. Consulte "Utilización de operadores para manipular los valores de las expresiones" a pagina 64.
- Asegúrese de no existan varios elementos de ActionScript con el mismo nombre.

  Es conveniente asignar nombres únicos a las variables, funciones, objetos y propiedades. No obstante, las variables locales constituyen una excepción: sólo es necesario que sean únicas dentro de su ámbito de utilización y suelen reutilizarse con frecuencia como contadores. Consulte "Ámbito de una variable" a pagina 61.

Si desea obtener sugerencias adicionales acerca de la solución de problemas en películas de Flash, consulte el Centro de soporte de Flash en la dirección http://www.macromedia.com/support/flash.

Utilización del Depurador

El Depurador permite detectar errores en una película a medida que ésta se reproduce en Flash Player. Es posible acceder a la lista de visualización de clips de película y de películas cargadas y cambiar los valores de las variables y propiedades con el fin de determinar los valores correctos. Después pueden volver a editarse los scripts de modo que produzcan los resultados correctos. Para utilizar el Depurador es necesario ejecutar el Reproductor de depuración de Flash, una versión especial de Flash Player.

El Reproductor de depuración de Flash se instala automáticamente con la aplicación de creación de Flash 5. Permite descargar la lista de visualización, los pares nombre de variable y valor, así como los pares nombre de propiedad y valor al Depurador en la aplicación de creación de Flash.

Para mostrar el Depurador:

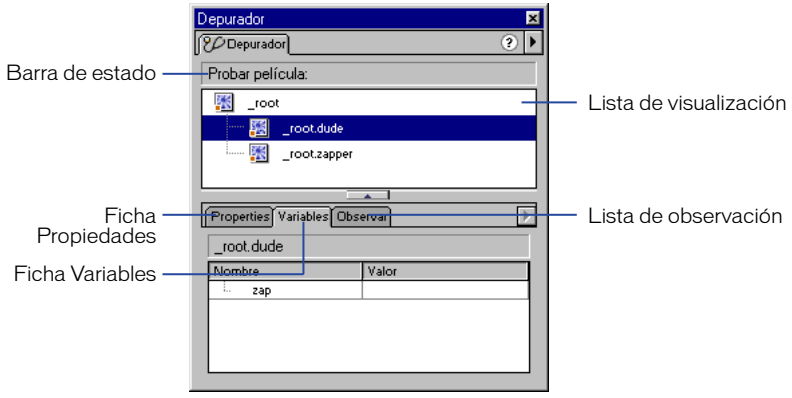
Seleccione Ventana > Depurador.

El Depurador se abrirá en estado inactivo. No aparecerá información alguna en la lista de visualización hasta que se genere un comando desde Flash Player.

Para activar el Depurador en modo de prueba de película:

Seleccione Control > Depurar película.

El Depurador se abrirá en estado activo.



Habilitación de la depuración de películas

Durante la exportación de películas de Flash Player, puede optarse por habilitar la depuración para la película y crear una contraseña de depuración. Si la depuración no se habilita, el Depurador no se activará.

Al igual que sucede en JavaScript o HTML, cualquier variable de ActionScript de cliente puede, teóricamente, ser vista por el usuario. Para guardar variables de un modo seguro, debe enviarlas a una aplicación de servidor en lugar de almacenarlas en una película.

No obstante, como creador de Flash, podría tener otros secretos comerciales, como estructuras de clips de películas, que no desee revelar. Para asegurar que sólo los usuarios de confianza pueden ver sus películas con el Reproductor de depuración de Flash, puede publicar las películas con una contraseña de Depurador.

Para habilitar la depuración y crear una contraseña:

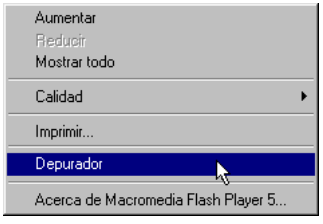
1. Seleccione Archivo > Configuración de publicación.
2. Haga clic en la ficha Flash.
3. Seleccione Depuración permitida
4. Para establecer una contraseña, especifíquela en el cuadro Contraseña.

   Sin esta contraseña no será posible descargar información al Depurador. Si deja vacío el campo de contraseña, no será necesaria ninguna.

Para activar el Depurador en un navegador Web:

1. Haga clic con el botón derecho del ratón (Windows), o Control-clic (Macintosh), para abrir el menú contextual del Reproductor de depuración de Flash.
2. Seleccione Depurador.

Nota: El Depurador sólo puede controlar una película cada vez. Para utilizar el Depurador, Flash debe estar abierto.



Menú contextual del Reproductor de depuración de Flash.

La barra de estado

Una vez activada, la barra de estado del Depurador mostrará la URL o la ruta del archivo local correspondiente a la película. Flash Player se implementa de diferentes formas, dependiendo del entorno de reproducción. La barra de estado del Depurador muestra el tipo de reproductor de Flash que ejecuta la película:

- Modo de prueba de película
- Reproductor independiente
- complemento de Netscape

  El complemento de Netscape se utiliza con Netscape Navigator en Windows y Macintosh y con Microsoft Internet Explorer en Macintosh.

- ActiveX, control

  El control ActiveX se utiliza con Internet Explorer en Windows.

La lista de visualización

Cuando el Depurador está activo muestra una vista dinámica de la lista de visualización del clip de película. Pueden expandirse y contraerse ramas con objeto de ver todos los clips de película que estén en el Escenario en un momento determinado. Cuando se agregan o eliminan clips de película, la lista de visualización refleja los cambios al instante. Puede cambiarse el tamaño de la lista de visualización desplazando el separador horizontal o arrastrando desde la esquina inferior derecha.

Visualización y modificación de variables

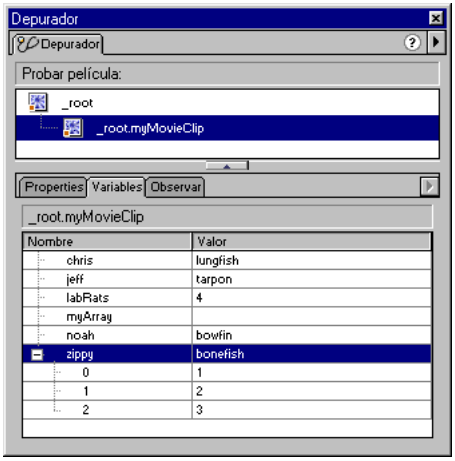
La ficha Variables del Depurador muestra los nombres y valores de cada una de las variables definidas en la película. Si se modifica el valor de una variable en la ficha Variables, el cambio se reflejará en la película mientras ésta se reproduce. Por ejemplo, para comprobar la detección de una colisión en un juego, podría introducir el valor de la variable para situar una bola en la ubicación oportuna junto a una pared.

Para ver una variable:

1 Seleccione el clip de película que contiene la variable en la lista de visualización.

2 Haga clic en la ficha Variables.

La lista de visualización se actualizará automáticamente según se vaya reproduciendo la película. Si se elimina de la película un clip de película en un fotograma determinado, dicho clip de película se eliminará también de la lista de visualización en el Depurador; de este modo se eliminará el nombre de la variable y su valor.



Para modificar el valor de una variable:

Seleccione el valor y especifique otro diferente.

El valor debe ser una constante (por ejemplo, `"Hola"`, `3523`, o `"http://www.macromedia.com"`) y no una expresión (como `x + 2` o `eval("name:" +i)`). El valor puede ser una cadena (cualquier valor encerrado entre comillas (`""`)), un número o un valor booleano (`true` o `false`).

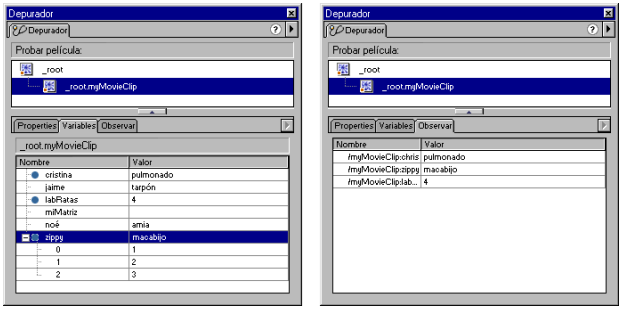
Las variables de tipo matriz y los objetos se muestran en la ficha Variables. Haga clic en el botón Agregar (+) para ver sus propiedades y valores. Sin embargo, no es posible introducir valores de variables de objetos ni variables de tipo matriz (por ejemplo, `{name: "Soy un objeto"}` o `[1, 2, 3]`) en los campos de valores.

Nota: Para mostrar el valor de una expresión en modo de prueba de película, utilice la acción `trace`. Consulte “Utilización de la acción trace” a pagina 173.

Utilización de la lista Observar

Para controlar un conjunto de variables críticas de un modo razonable, deben marcarse las variables que se desee que aparezcan en la lista Observar. La lista Observar muestra la ruta absoluta de la variable, así como su valor. También puede especificarse un nuevo valor para una variable desde la lista Observar.

Sólo pueden agregarse variables a la lista Observar, no propiedades ni funciones.



Variables marcadas para la lista Observar y variables en la lista Observar.

Utilice uno de los siguientes procedimientos para agregar variables a la lista Observar:

- En la ficha Variables, haga clic con el botón derecho del ratón (Windows) o Control-clic (Macintosh) en una variable seleccionada y elija Observar en el menú contextual. Aparecerá un punto azul junto a la variable.
- En la ficha Observar, haga clic con el botón derecho del ratón (Windows) o Control-clic (Macintosh) y seleccione Agregar en el menú contextual. Especifique el nombre y el valor de la variable en los campos correspondientes.

Para eliminar variables de la lista Observar:

En la ficha Observar, haga clic con el botón derecho del ratón (Windows) o Control-clic (Macintosh) y seleccione Eliminar en el menú contextual.

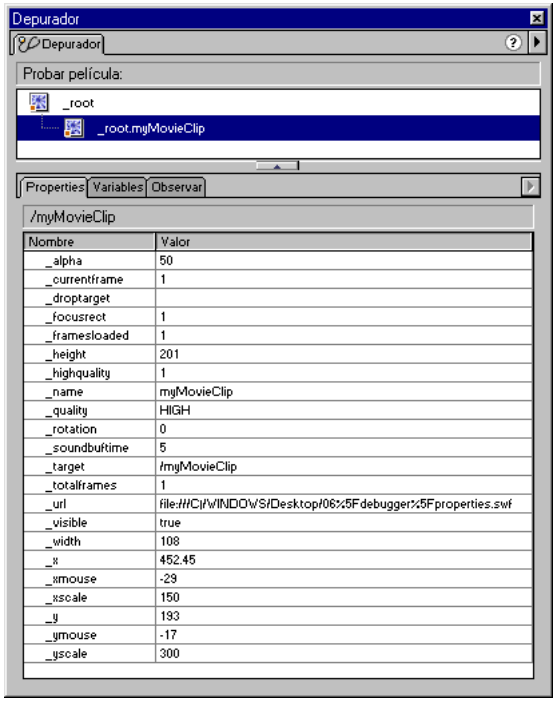
Visualización de las propiedades de las películas y modificación de las propiedades editables

La ficha Propiedades del Depurador muestra todos los valores de las propiedades de cualquier clip de película del Escenario. Al cambiar el valor de una propiedad, el resultado puede verse reflejado en la película a medida que se reproduce. Algunas propiedades de los clips de película son de sólo lectura y no pueden modificarse.

Para mostrar las propiedades de un clip de película:

1 Seleccione un clip de película de la lista de visualización.

2 Haga clic en la ficha Propiedades.



Nombre	Valor
_alpha	50
_currentframe	1
_droptarget	
_focusrect	1
_framesloaded	1
_height	201
_highquality	1
_name	myMovieClip
_quality	HIGH
_rotation	0
_soundbuftime	5
_target	/myMovieClip
_totalframes	1
_url	file:///C|/WINDOWS/Desktop/06%5Fdebugger%5Fproperties.swf
_visible	true
_width	108
_x	452.45
_xmouse	-29
_xscale	150
_y	193
_ymouse	-17
_yscale	300

Para modificar el valor de una propiedad:

Seleccione el valor y especifique otro diferente.

El valor debe ser una constante (por ejemplo, `50` o `"caramelo"`), pero no una expresión (como `x + 50`). El valor puede ser una cadena (cualquier valor encerrado entre comillas (`""`)), un número o un valor booleano (`true` o `false`). No es posible especificar objetos ni valores de tipo matriz (por ejemplo, `{id: "rogue"}` o `[1, 2, 3]`) en el Depurador.

Si desea obtener más información, consulte las secciones “Cadena” a pagina 56 y “Utilización de operadores para manipular los valores de las expresiones” a pagina 64.

Nota: Para mostrar el valor de una expresión en modo de prueba de película, utilice la acción `trace`. Consulte “Utilización de la acción trace” a pagina 173.

Utilización de la ventana Salida

En el modo de prueba de película, la ventana Salida muestra información que ayuda a solucionar problemas relacionados con la película. Algún tipo de información, como los errores sintácticos, se muestra automáticamente. Puede accederse a otro tipo de información mediante los comandos Mostrar objetos y Mostrar variables (Consulte las secciones “Utilización del comando Mostrar objetos” a pagina 171 y “Utilización del comando Mostrar variables” a pagina 172).

Si se utiliza la acción `trace` en los scripts, será posible enviar información específica a la ventana Salida durante la reproducción de la película. Por ejemplo, podrían enviarse notas acerca del estado de la película o el valor de una expresión. Consulte “Utilización de la acción trace” a pagina 173.

Para mostrar la ventana Salida:

1. Si la película no se está ejecutando en el modo de prueba, seleccione Control > Probar película.
2. Seleccione Ventana > Salida.

   Aparecerá la ventana Salida.

Nota: Cuando existen errores sintácticos en un script, la ventana Salida aparece automáticamente.

3 Para trabajar con el contenido de la ventana Salida utilice el menú Opciones:

- Seleccione Opciones > Copiar si desea copiar el contenido de la ventana Salida en el Portapapeles.
- Seleccione Opciones > Borrar si desea borrar el contenido de la ventana.
- Seleccione Opciones > Guardar en archivo si desea guardar en contenido de la ventana en un archivo de texto.
- Seleccione Opciones > Imprimir si desea imprimir el contenido de la ventana.

Utilización del comando Mostrar objetos

En el modo de prueba de películas, el comando Mostrar objetos indica en una lista jerárquica el nivel, fotograma, tipo de objeto (forma, clip de película o botón) y ruta de destino de una instancia de clip de película. Resulta especialmente útil para encontrar el nombre de instancia y la ruta de destino correctas. A diferencia del Depurador, la lista no se actualiza automáticamente a medida que se reproduce la película; es necesario ejecutar el comando Mostrar objetos cada vez que se desea enviar información a la ventana Salida.

Para mostrar una lista de objetos en una película:

1 Si la película no se está ejecutando en el modo de prueba, seleccione Control > Probar película.

2 Seleccione Depurar > Mostrar objetos.

La ventana Salida mostrará una lista de todos los objetos que se encuentren actualmente en el Escenario, como en el siguiente ejemplo:

```
Layer #0: Frame=3
Movie Clip: Frame=1 Target=_root.MC
      Shape:
   Movie Clip: Frame=1 Target=_root.instance3
      Shape:
   Button:
      Movie Clip: Frame=1 Target=_root.instance3.instance2
         Shape:
```

Nota: El comando Mostrar objetos no enumera todos los objetos de datos de ActionScript. En este contexto, un objeto se considera que es una forma o símbolo en el Escenario.

Utilización del comando Mostrar variables

En el modo de prueba de películas, el comando Mostrar variables enumera una lista de todas las variables definidas actualmente en la película. Resulta especialmente útil para encontrar el nombre de la variable y la ruta de destino correctos. A diferencia del Depurador, la lista no se actualiza automáticamente a medida que se reproduce la película; es necesario ejecutar el comando Mostrar variables cada vez que se desea enviar información a la ventana Salida.

Para mostrar una lista de variables en una película:

1. Si la película no se está ejecutando en el modo de prueba, seleccione Control > Probar película.
2. Seleccione Depurar > Mostrar variables.

La ventana Salida mostrará una lista de todas las variables definidas actualmente en la película, como en el siguiente ejemplo:

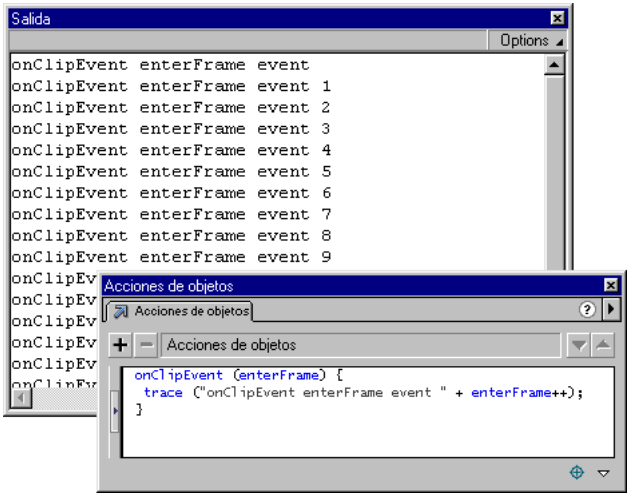
```
Level #0:
   Variable _root.country = "Sweden"
   Variable _root.city = "San Francisco"
Movie Clip:  Target=""
Variable _root.instance1.firstName = “Rick”
```

Utilización de la acción trace

Cuando se utiliza la acción `trace` en un script, puede enviarse información a la ventana Salida. Por ejemplo, cuando se prueba una película o una escena, pueden enviarse notas de programación determinadas a la ventana o hacer que aparezcan resultados concretos cuando se presiona un botón o se reproduce un fotograma. La acción `trace` es similar a la sentencia `alert` de JavaScript.

El uso de la acción `trace` en un a script, permite utilizar expresiones como argumentos. El valor de una expresión se mostrará en la ventana Salida en modo de prueba de películas, como en el siguiente ejemplo:

```
onClipEvent(enterFrame){
   trace("onClipEvent enterFrame " + enterFrame++)
}
```



La acción `trace` devuelve valores que se muestran en la ventana Salida.

CAPÍTULO 7
Diccionario de ActionScript

Esta parte de la *Guía de referencia de ActionScript* describe la sintaxis y la utilización de los elementos de ActionScript en Flash 5 y versiones posteriores. Las entradas de esta guía son las mismas que en la Ayuda del Diccionario de ActionScript. Para utilizar los ejemplos en un script, copie el texto de ejemplo desde la Ayuda del Diccionario de ActionScript y péguelo en el panel Acciones en modo Experto.

En el diccionario aparece una lista de todos los elementos de ActionScript (operadores, palabras clave, sentencias, acciones, propiedades, funciones, objetos y métodos). Para ver una introducción de todas las entradas del diccionario, consulte "Contenido del diccionario" a pagina 178; las tablas de esta sección son un buen punto de comienzo para buscar operadores simbólicos o métodos cuya clase de objeto no conoce.

ActionScript sigue el estándar ECMA-262 (la especificación escrita por la Asociación Europea de Fabricantes de Ordenadores, European Computer Manufacturers Association) a no ser que se indique lo contrario.

Existen dos tipos de entradas en este diccionario:

- Entradas individuales para operadores, palabras clave, funciones, variables, propiedades, métodos y sentencias
- Entradas de objetos, que proporcionan información general sobre los objetos predefinidos

Utilice esta información en las entradas de ejemplo para interpretar la estructura y las convenciones utilizadas en estos dos tipos de entradas.

Entrada de muestra para la mayoría de los elementos de ActionScript

El diccionario de muestras siguiente explica las convenciones utilizadas para los elementos de ActionScript que no son objetos.

Título de entrada

Todas las entradas aparecen en la lista en orden alfabético. El orden alfabético ignora las mayúsculas, los signos de subrayado iniciales, etc.

Sintaxis

La sección "Sintaxis" proporciona la sintaxis correcta para la utilización del elemento de ActionScript en su código. La parte de código de la sintaxis aparece en `fuente de código` y los argumentos que debe introducir aparecen en *`fuente de código en cursiva`*. Los corchetes indican argumentos opcionales.

Argumentos

En esta sección se describen los argumentos de la lista de sintaxis.

Descripción

En esta sección se identifica el elemento (por ejemplo, como un operador, método, función u otro elemento) y después se describe como se utiliza el elemento.

Reproductor

En esta sección se indica qué versiones del reproductor soportan el elemento. Ésta no es la misma que la versión de Flash utilizada para crear contenidos. Por ejemplo, si está creando contenido para Flash Player 4 utilizando la herramienta de autor de Flash 5, no puede utilizar elementos de ActionScript que sólo están disponibles en Flash Player 5.

Con la introducción de ActionScript de Flash 5, algunos de los elementos de ActionScript de Flash 4 (y de versiones anteriores) se han desestimado. Aunque los elementos desestimados aún los admite Flash Player 5, se recomienda que utilice los nuevos elementos de Flash 5.

Además, se ha aumentado mucho la funcionalidad de los operadores en Flash 5. No sólo se han introducido bastantes operadores matemáticos nuevos, sino que algunos de los operadores más antiguos ahora son capaces de manejar tipos de datos adicionales. Para mantener la coherencia de los datos, los archivos de Flash 4 se modifican automáticamente cuando se importan al entorno de autoría de Flash 5, pero estas modificaciones no afectarán a la funcionalidad del script original. Si desea obtener más información, consulte las entradas de + (suma), < (menor que), > (mayor que), <= (menor o igual que), >= (mayor o igual que), != (no igualdad) y = (igualdad).

Ejemplo
En esta sección se aparece una muestra de código que demuestra como utilizar el elemento.

Véase también
En esta sección se muestra una lista de las entradas del diccionario de ActionScript relacionadas.

Entrada de muestra para objetos

La entrada de muestra del diccionario siguiente explica las convenciones utilizadas para los objetos de ActionScript predefinidos. Los objetos aparecen en la lista en orden alfabético con todos los demás elementos del diccionario.

Título de entrada

El título de entrada proporciona el nombre del objeto. El nombre del objeto va seguido de un párrafo que contiene información general sobre el objeto.

Tablas de resumen sobre métodos y propiedades

La entrada de cada objeto contiene una tabla con una lista de todos los métodos asociados con el objeto. Si el objeto tiene propiedades (a menudo constantes), estos elementos se resumen en una tabla adicional. Todos los métodos y propiedades que aparecen en la lista de estas tablas también tienen sus propias entradas en el diccionario, que siguen a la entrada del objeto.

Constructor

Si el objeto requiere la utilización de un constructor para acceder a sus métodos y propiedades, el constructor se describe al final de la entrada del objeto. Esta descripción tiene todos los elementos estándar (descripción de sintaxis, etc.) de las otras entradas del diccionario.

Listas de métodos y propiedades

Los métodos y propiedades de un objeto aparecen en una lista alfabética tras la entrada del objeto.

Contenido del diccionario

Todas las entradas del diccionario aparecen en la lista en orden alfabético. Sin embargo, algunos operadores son símbolos y se presentan en orden ASCII. Además, los métodos que están asociados con un objeto aparecen en la lista junto al nombre del objeto (por ejemplo, el método `abs` del objeto Math aparece en la lista como `Math.abs`).

Las dos tablas siguientes le ayudarán a localizar estos elementos. En la primera tabla aparece una lista de los operadores simbólicos en el orden en el que aparecen en el diccionario. En la segunda tabla aparecen todos los demás elementos de ActionScript.

Nota: Para ver la prioridad y la posibilidad de asociación de los operadores, consulte el Apéndice A.

Operadores simbólicos	
--	(disminución)
++	incremento
!	(valor NOT lógico)
!=	(no igualdad)
%	(módulo)
%=	(asignación de módulo)
&	(operador AND como bit)
&&	(operador AND de cortocircuito)
&=	(operador de asignación AND como bit)
( )	(paréntesis)
-	(menos)
*	(multiplicación)
*=	(asignación de multiplicación)
,	(coma)
.	(punto)
? :	(condicional)
/	(división)
//	(delimitador de comentario)
/*	(delimitador de comentario)

Operadores simbólicos	
/=	(asignación de división)
[]	(acceso a matriz)
^	(operador XOR como bit)
^=	(operador de asignación XOR como bit)
{}	(inicializador de objeto)
|	(operador OR como bit)
||	(valor OR lógico)
|=	(operador de asignación OR como bit)
~	(operador NOT como bit)
+	(suma)
+=	(asignación de suma)
<	(menor que)
<<	(desplazamiento a la izquierda como bit)
<<=	(desplazamiento a la izquierda como bit y asignación)
<=	(menor o igual que)
<>	(no igualdad)
=	(asignación)
-=	(asignación de negación)
==	(igualdad)
>	(mayor que)
>=	(mayor o igual que)
>>	(desplazamiento a la derecha como bit)
>>=	(desplazamiento a la derecha como bit y asignación)
>>>	(desplazamiento a la derecha como bit sin signo)
>>>=	(desplazamiento a la derecha como bit sin signo y asignación)

En la tabla siguiente se muestra una lista de todos los elementos de ActionScript que no son operadores simbólicos.

Elemento de ActionScript	Véase la entrada
abs	“Math.abs” a pagina 299
acos	“Math.acos” a pagina 299
add	“add” a pagina 223
and	“and” a pagina 224
_alpha	“_alpha” a pagina 224
appendChild	“XML.appendChild” a pagina 395
Array	“Array (objeto)” a pagina 225
asin	“Math.asin” a pagina 300
atan	“Math.atan” a pagina 300
atan2	“Math.atan2” a pagina 301
attachMovie	“MovieClip.attachMovie” a pagina 315
attachSound	“Sound.attachSound” a pagina 359
attributes	“XML.attributes” a pagina 396
BACKSPACE	“Key.BACKSPACE” a pagina 285
Boolean	“Boolean (función)” a pagina 234, “Boolean (objeto)” a pagina 234
break	“break” a pagina 236
call	“call” a pagina 237
CAPSLOCK	“Key.CAPSLOCK” a pagina 285
ceil	“Math.ceil” a pagina 301
charAt	“String.charAt” a pagina 373
charCodeAt	“String.charCodeAt” a pagina 373
childNodes	“XML.childNodes” a pagina 397
chr	“chr” a pagina 237
cloneNode	“XML.cloneNode” a pagina 397
close	“XMLSocket.close” a pagina 412
Color	“Color (objeto)” a pagina 238

Elemento de ActionScript	Véase la entrada
concat	“Array.concat” a pagina 227, “String.concat” a pagina 373
connect	“XMLSocket.connect” a pagina 413
constructor	Array, Boolean, Color, Date, Number, Object, Sound, String, XML, XMLSocket
continue	“continue” a pagina 242
CONTROL	“Key.CONTROL” a pagina 285
cos	“Math.cos” a pagina 301
createElement	“XML.createElement” a pagina 398
createTextNode	“XML.createTextNode” a pagina 398
_currentframe	“_currentframe” a pagina 242
Date	“Date (objeto)” a pagina 243
delete	“delete” a pagina 260
DELETEKEY	“Key.DELETEKEY” a pagina 286
docTypeDecl	“XML.docTypeDecl” a pagina 399
do...while	“do...while” a pagina 262
DOWN	“Key.DOWN” a pagina 286
_droptarget	“_droptarget” a pagina 263
duplicateMovieClip	“duplicateMovieClip” a pagina 264, “MovieClip.duplicateMovieClip” a pagina 315
E	“Math.E” a pagina 302
else	“else” a pagina 265
END	“Key.END” a pagina 286
ENTER	“Key.ENTER” a pagina 287
eq	“eq (equal–string specific)” a pagina 265
escape (function)	“escape” a pagina 266
ESCAPE (constant)	“Key.ESCAPE” a pagina 287
eval	“eval” a pagina 266
evaluate	“evaluate” a pagina 267
exp	“Math.exp” a pagina 302

Elemento de ActionScript	Véase la entrada
firstChild	“XML.firstChild” a pagina 400
floor	“Math.floor” a pagina 303
_focusrect	“_focusrect” a pagina 268
for	“for” a pagina 268
for... in	“for...in” a pagina 270
_framesloaded	“_framesloaded” a pagina 271
fromCharCode	“String.fromCharCode” a pagina 374
fscommand	“fscommand” a pagina 272
function	“function” a pagina 272
ge	“ge (mayor o igual que, específico de cadena)” a pagina 274
getAscii	“Key.getAscii” a pagina 287
getBeginIndex	“Selection.getBeginIndex” a pagina 354
getBounds	“MovieClip.getBounds” a pagina 316
getBytesLoaded	“MovieClip.getBytesLoaded” a pagina 317
getBytesTotal	“MovieClip.getBytesTotal” a pagina 317
getCaretIndex	“Selection.getCaretIndex” a pagina 354
getCode	“Key.getCode” a pagina 288
getDate	“Date.getDate” a pagina 246
getDay	“Date.getDay” a pagina 246
getEndIndex	“Selection.getEndIndex” a pagina 355
getFocus	“Selection.getFocus” a pagina 355
getFullYear	“Date.getFullYear” a pagina 247
getHours	“Date.getHours” a pagina 247
getMilliseconds	“Date.getMilliseconds” a pagina 248
getMinutes	“Date.getMinutes” a pagina 248
getMonth	“Date.getMonth” a pagina 248
getPan	“Sound.getPan” a pagina 360
getProperty	“getProperty” a pagina 274

Elemento de ActionScript	Véase la entrada
getRGB	“Color.setRGB” a pagina 240
getSeconds	“Date.getSeconds” a pagina 249
getTime	“Date.getTime” a pagina 249
getTimer	“getTimer” a pagina 275
getTimezoneOffset	“Date.getTimezoneOffset” a pagina 249
getTransform	“Color.getTransform” a pagina 239, “Sound.getTransform” a pagina 360
getURL	“getURL” a pagina 275, “MovieClip.getURL” a pagina 318
getUTCDate	“Date.getUTCDate” a pagina 250
getUTCDay	“Date.getUTCDay” a pagina 250
getUTCFullYear	“Date.getUTCFullYear” a pagina 251
getUTCHours	“Date.getUTCHours” a pagina 251
getUTCMilliseconds	“Date.getUTCMilliseconds” a pagina 251
getUTCMinutes	“Date.getUTCMinutes” a pagina 252
getUTCMonth	“Date.getUTCMonth” a pagina 252
getUTCSeconds	“Date.getUTCSeconds” a pagina 252
getVersion	“getVersion” a pagina 276
getVolume	“Sound.getVolume” a pagina 361
getYear	“Date.getYear” a pagina 253
globalToLocal	“MovieClip.globalToLocal” a pagina 318
gotoAndPlay	“gotoAndPlay” a pagina 277, “MovieClip.gotoAndPlay” a pagina 319
gotoAndStop	“gotoAndStop” a pagina 277, “MovieClip.gotoAndStop” a pagina 319
gt	“gt (greater than –string specific)” a pagina 278
hasChildNodes	“XML.haschildNodes” a pagina 400
_height	“_height” a pagina 278
hide	“Mouse.hide” a pagina 312
_highquality	“_highquality” a pagina 279

Elemento de ActionScript	Véase la entrada
hitTest	“MovieClip.hitTest” a pagina 320
HOME	“Key.HOME” a pagina 288
if	“if” a pagina 279
ifFrameLoaded	“ifFrameLoaded” a pagina 280
#include	“#include” a pagina 280
indexOf	“String.indexOf” a pagina 374
Infinity	“Infinity” a pagina 281
INSERT	“Key.INSERT” a pagina 288
insertBefore	“XML.insertBefore” a pagina 401
int	“int” a pagina 281
isDown	“Key.isDown” a pagina 289
isFinite	“isFinite” a pagina 281
isNaN	“isNaN” a pagina 282
isToggled	“Key.isToggled” a pagina 289
join	“Array.join” a pagina 227
Key	“Key (objeto)” a pagina 283
lastChild	“XML.lastChild” a pagina 401
lastIndexOf	“String.indexOf” a pagina 374
le	“le (menor o igual que–específico de cadena)” a pagina 292
LEFT	“Key.LEFT” a pagina 289
length	“length” a pagina 292, “Array.length” a pagina 228, “String.length” a pagina 375
LN2	“Math.LN2” a pagina 304
LN10	“Math.LN10” a pagina 305
load	“XML.load” a pagina 402
loaded	“XML.loaded” a pagina 402
loadMovie	“loadMovie” a pagina 294, “MovieClip.loadMovie” a pagina 321

Elemento de ActionScript	Véase la entrada
loadVariables	“loadVariables” a pagina 296, “MovieClip.loadVariables” a pagina 322
localToGlobal	“MovieClip.localToGlobal” a pagina 323
log	“Math.log” a pagina 303
LOG2E	“Math.LOG2E” a pagina 303
LOG10E	“Math.LOG10E” a pagina 304
lt	“le (menor o igual que—específico de cadena)” a pagina 292
Math	“Matemáticas (objeto)” a pagina 297
max	“Math.max” a pagina 305
maxscroll	“maxscroll” a pagina 309
MAX_VALUE	“Number.MAX_VALUE” a pagina 334
mbchr	“mbchr” a pagina 310
mblength	“mblength” a pagina 310
mbord	“mbord” a pagina 311
mbsubstring	“mbsubstring” a pagina 311
min	“Math.min” a pagina 305
MIN_VALUE	“Number.MIN_VALUE” a pagina 334
Mouse	“Mouse (objeto)” a pagina 312
MovieClip	“MovieClip (objeto)” a pagina 313
_name	“_name” a pagina 327
NaN	“NaN” a pagina 328, “Number.NaN” a pagina 335
ne	“ne (no igual, específico de cadena)” a pagina 328
NEGATIVE_INFINITY	“Number.NEGATIVE_INFINITY” a pagina 335
new **(operator)**	“new” a pagina 328
newline	“newline” a pagina 329
nextFrame	“nextFrame” a pagina 330, “MovieClip.nextFrame” a pagina 324
nextScene	“nextScene” a pagina 330
nextSibling	“XML.nextSibling” a pagina 403

Elemento de ActionScript	Véase la entrada
nodeName	“XML.nodeName” a pagina 403
nodeType	“XML.nodeType” a pagina 404
nodeValue	“XML.nodeValue” a pagina 404
not	“not” a pagina 331
null	“null” a pagina 331
Number	“Number (función)” a pagina 332, “Number (objeto)” a pagina 332
Object	“Object (objeto)” a pagina 337
On	“on(mouseEvent)” a pagina 340
onClipEvent	“onClipEvent” a pagina 338
onClose	“XMLSocket.onClose” a pagina 414
onConnect	“XMLSocket.onConnect” a pagina 415
OnLoad	“XML.onLoad” a pagina 405
onXML	“XMLSocket.onXML” a pagina 416
or (logical OR)	“or” a pagina 342
ord	“ord” a pagina 342
_parent	“_parent” a pagina 342
parentNode	“XML.parentNode” a pagina 406
parseFloat	“parseFloat” a pagina 343
parseInt	“parseInt” a pagina 344
parseXML	“XML.parseXML” a pagina 406
PGDN	“Key.PGDN” a pagina 290
PGUP	“Key.PGUP” a pagina 290
PI	“Math.PI” a pagina 306
play	“play” a pagina 345, “MovieClip.play” a pagina 324
pop	“Array.pop” a pagina 229
POSITIVE_INFINITY	“Number.POSITIVE_INFINITY” a pagina 335
pow	“Math.pow” a pagina 306

Elemento de ActionScript	Véase la entrada
prevFrame	“prevFrame” a pagina 345, “MovieClip.prevFrame” a pagina 324
previousSibling	“XML.previousSibling” a pagina 406
prevScene	“prevScene” a pagina 346
print	“print” a pagina 346
printAsBitmap	“printAsBitmap” a pagina 348
push	“Array.push” a pagina 229
_quality	“_quality” a pagina 349
random	“random” a pagina 350, “Math.random” a pagina 307
removeMovieClip	“removeMovieClip” a pagina 350, “MovieClip.removeMovieClip” a pagina 325
removeNode	“XML.removeNode” a pagina 407
return	“return” a pagina 351
reverse	“Array.reverse” a pagina 230
RIGHT	“Key.RIGHT” a pagina 290
_root	“_root” a pagina 351
_rotation	“_rotation” a pagina 352
round	“Math.round” a pagina 307
scroll	“scroll” a pagina 353
Selection	“Selection (objeto)” a pagina 353
send	“XML.send” a pagina 407, “XMLSocket.send” a pagina 417
sendAndLoad	“XML.sendAndLoad” a pagina 407
set	“set” a pagina 356
setDate	“Date.setDate” a pagina 253
setFocus	“Selection.setFocus” a pagina 356
setFullYear	“Date.setFullYear” a pagina 253
setHours	“Date.setHours” a pagina 254
setMilliseconds	“Date.setMilliseconds” a pagina 254
setMinutes	“Date.setMinutes” a pagina 255

Elemento de ActionScript	Véase la entrada
setMonth	“Date.setMonth” a pagina 255
setPan	“Sound.setPan” a pagina 361
setProperty	“setProperty” a pagina 357
setRGB	“Color.setRGB” a pagina 240
setSeconds	“Date.setSeconds” a pagina 255
setSelection	“Selection.setSelection” a pagina 356
setTime	“Date.setTime” a pagina 256
setTransform	“Color.setTransform” a pagina 240, “Sound.setTransform” a pagina 362
setUTCDate	“Date.setUTCDate” a pagina 256
setUTCFullYear	“Date.setUTCFullYear” a pagina 256
setUTCHours	“Date.setUTCHours” a pagina 257
setUTCMilliseconds	“Date.setUTCMilliseconds” a pagina 257
setUTCMinutes	“Date.setUTCMinutes” a pagina 258
setUTCMonth	“Date.setUTCMonth” a pagina 258
setUTCSeconds	“Date.setUTCSeconds” a pagina 258
setVolume	“Sound.setVolume” a pagina 365
setYear	“Date.setYear” a pagina 259
shift (method)	“Array.shift” a pagina 230
SHIFT (constant)	“Key.SHIFT” a pagina 291
show	“Mouse.show” a pagina 313
sin	“Math.sin” a pagina 307
slice	“Array.slice” a pagina 231, “String.slice” a pagina 375
sort	“Array.sort” a pagina 231
Sound	“Sound (objeto)” a pagina 358
_soundbuftime	“_soundbuftime” a pagina 366
SPACE	“Key.SPACE” a pagina 291
splice	“Array.splice” a pagina 233
split	“String.split” a pagina 376

Elemento de ActionScript	Véase la entrada
sqrt	“Math.sqrt” a pagina 308
SQRT1_2	“Math.SQRT1_2” a pagina 308
SQRT2	“Math.SQRT2” a pagina 309
start	“Sound.start” a pagina 365
startDrag	“startDrag” a pagina 367, “MovieClip.startDrag” a pagina 325
status	“XML.status” a pagina 408
stop	“stop” a pagina 368,“MovieClip.stop” a pagina 326,“Sound.stop” a pagina 366
stopAllSounds	“stopAllSounds” a pagina 368
stopDrag	“stopDrag” a pagina 369, “MovieClip.stopDrag” a pagina 326
String	“String (función)” a pagina 369, “Cadena (objeto)” a pagina 371,“" " (delimitador de cadena)” a pagina 370
substr	“String.substr” a pagina 376
substring	“substring” a pagina 378, “String.substring” a pagina 377
swapDepths	“MovieClip.swapDepths” a pagina 326
TAB	“Key.TAB” a pagina 291
tan	“Math.tan” a pagina 309
_target	“_target” a pagina 378
targetPath	“targetPath” a pagina 379
tellTarget	“tellTarget” a pagina 379
this	“this” a pagina 380
toggleHighQuality	“toggleHighQuality” a pagina 381
toLowerCase	“String.toLowerCase” a pagina 377
toString	“Array.toString” a pagina 233,“Boolean.toString” a pagina 235, “Date.toString” a pagina 259, “Number.toString” a pagina 336, “Object.toString” a pagina 338, “XML.toString” a pagina 409
_totalframes	“_totalframes” a pagina 382
toUpperCase	“String.toUpperCase” a pagina 378

Elemento de ActionScript	Véase la entrada
trace	“trace” a pagina 382
typeof	“typeof” a pagina 383
unescape	“unescape” a pagina 384
unloadMovie	“unloadMovie” a pagina 384, “MovieClip.unloadMovie” a pagina 327
unshift	“Array.shift” a pagina 230
UP	“Key.UP” a pagina 292
updateAfterEvent	“updateAfterEvent” a pagina 385
_url	“_url” a pagina 386
UTC	“Date.UTC” a pagina 260
valueOf	“Boolean.valueOf” a pagina 236, “Number.valueOf” a pagina 336, “Object.valueOf” a pagina 338
var	“var” a pagina 386
_visible	“_visible” a pagina 386
void	“void” a pagina 387
while	“while” a pagina 387
_width	“_width” a pagina 389
with	“with” a pagina 389
_x	“_x” a pagina 392
XML	“XML (objeto)” a pagina 393
xmlDecl	“XML.xmlDecl” a pagina 409
XMLSocket	“XMLSocket (objeto)” a pagina 410
_xmouse	“_xmouse” a pagina 418
_xscale	“_xscale” a pagina 418
_y	“_y” a pagina 419
_ymouse	“_ymouse” a pagina 420
_yscale	“_yscale” a pagina 420

-- (disminución)

Sintaxis

--expresión
expresión--

Argumentos

expresión Una variable, número, elemento en una matriz, o la propiedad de un objeto.

Descripción

Operador; un operador unario predisminución y postdisminución que resta 1 de la *expresión*. La forma predisminución del operador (*--expresión*) resta 1 de *expresión* y devuelve el resultado. La forma postdisminución del operador (*expresión--*) resta 1 de la *expresión* y devuelve el valor inicial de la *expresión* (el resultado anterior a la resta).

Reproductor

Flash 4 o posterior.

Ejemplo

La forma predisminución del operador resta `x` a 2 (`x - 1 = 2`) y devuelve el resultado como `y`:

```
x = 3;
y = --x
```

La forma postdisminución del operador resta `x` a 2 (`x - 1 = 2`) y devuelve el valor original (`x = 3`) como resultado `y`:

```
Si x = 3;
y = x--
```

++ (incremento)

Sintaxis

++expresión
expresión++

Argumentos

expresión Una variable, número, elemento en una matriz, o la propiedad de un objeto.

Descripción

Operador; un operador unario preincremento y postincremento que agrega 1 a la *expresión*. La forma preincremento del operador (*++expresión*) agrega 1 a la *expresión* y devuelve el resultado. La forma postincremento del operador (*expresión++*) agrega 1 a la *expresión* y devuelve el valor inicial de la *expresión* (el resultado anterior a la suma).

La forma preincremento del operador aumenta `x` a 2 (`x + 1 = 2`) y devuelve el resultado como `y`:

```
x = 1;
y = ++x
```

La forma postincremento del operador aumenta `x` a 2 (`x + 1 = 2`) y devuelve el valor original (`x = 1`) como resultado `y`:

```
x = 1;
y = x++
```

Reproductor
Flash 4 o posterior.

Ejemplo
El ejemplo siguiente utiliza ++ como operador preincremento con una sentencia `while`.

```
i = 0
while(i++ < 5){
// this section will execute five times
}
```

El ejemplo siguiente utiliza ++ como operador preincremento:

```
var a = [];
var i = 0;
while (i < 10) {
   a.push(++i);
}
trace(a.join());
```

Este script imprime lo que se muestra a continuación:

```
1,2,3,4,5,6,7,8,9
```

El ejemplo siguiente utiliza ++ como operador postincremento:

```
var a = [];
var i = 0;
while (i < 10) {
a.push(i++);
   }
trace(a.join());
```

Este script imprime lo que se muestra a continuación:

```
0,1,2,3,4,5,6,7,8,9
```

! (valor NOT lógico)

Sintaxis
`!expresión`

Argumentos
expresión Una variable o expresión evaluada.

Descripción
Operador (lógico); invierte el valor Booleano de una variable o expresión. Si *expresión* es una variable con un valor absoluto o convertido `true`, `!variable` el valor de `! expresión` es `false`. Si la expresión `x && y` evalúa como `false`, la expresión `!(x && y)` evalúa `true`. Este operador es idéntico al operador `not` que se utilizaba en Flash 4.

Reproductor
Flash 4 o posterior.

Ejemplo
En el ejemplo siguiente la variable `happy` está establecida en `false`, la condición `if` evalúa la condición `!happy`, y si la condición es `true`, `trace` envía una cadena a la ventana Salida.

```
happy = false;
if (!happy){
trace("don't worry be happy");
}
```

A continuación se muestran los resultados del operador `!`:

`! true` devuelve `false`

`! false` devuelve `true`

!= (no igualdad)

Sintaxis
`expresión1 != expresión2`

Argumentos
expresión1, expresión2 Números, cadenas, Booleanos, variables, objetos, matrices o funciones.

Descripción
Operador (de igualdad); comprueba el opuesto exacto del operador ==. Si *expresión1* es igual a *expresión2*, el resultado es `false`. Al igual que con el operador ==, la definición de *igual* depende de los tipos de datos que se comparan.

- Los números, cadenas y los valores Booleanos se comparan por valor.
- Las variables, objetos, matrices y funciones se comparan por referencia.

Reproductor
Flash 5 o posterior.

Ejemplo
El ejemplo siguiente muestra los resultados del operador `!=`:

`5 != 8` devuelve `true`

`5 != 5` devuelve `false`

El ejemplo siguiente muestra la utilización del operador `!=` en una sentencia `if`:

```
a = "David";
b = "Fool"
if (a != b)
trace("David is not a fool");
```

Véase también
"`==` (igualdad)" a pagina 218

% (módulo)

Sintaxis
`expresión1 % expresión2`

Argumentos
`expresión1, expresión2` Números, números enteros, números con coma flotante o cadenas que convierten en un valor numérico.

Descripción
Operador (aritmético); calcula el resto de *`expresión1`* dividido por *`expresión2`*. Si cualquiera de los argumentos de la *`expresión`* son no numéricos, el operador módulo intenta convertirlos en números.

Reproductor
Flash 4 o posterior. En archivos de Flash 4, el operador `%` se expande en el archivo SWF como `x - int(x/y) * y` y puede no ser tan rápido o preciso que en Flash Player 5.

Ejemplo
A continuación se muestra un ejemplo numérico de la utilización del operador `%`:

`12 % 5` devuelve `2`

`4.3 % 2.1` devuelve `0.1`

%= (asignación de módulo)

Sintaxis
expresión1 %= *expresión2*

Argumentos
expresión1, *expresión2* Números enteros y variables.

Descripción
Operador (de asignación); asigna *expresión1* el valor de *expresión1* % *expresión2*.

Reproductor
Flash 4 o posterior.

Ejemplo
A continuación se muestra la utilización del operador %= con variables y números:

x %= y es lo mismo que x = x % y

Si x = 14 y y = 5 entonces

x %= 5 devuelve 4

Véase también
"% (módulo)" a pagina 194

& (operador AND como bit)

Sintaxis
expresión1 & *expresión2*

Argumentos
expresión1, *expresión2* Cualquier número.

Descripción
Operador (como bit); convierte *expresión1* y *expresión2* en números enteros de 32 bits sin signo y realiza una operación Booleana AND en cada bit de los argumentos del número entero. El resultado es un nuevo número entero de 32 bits sin signo.

Reproductor
Flash 5 o posterior. Si se ha utilizado el operador & en Flash 4 para concatenar cadenas. En Flash 5 el operador & es un AND como bit y los operadores add y + concatenan cadenas. Los archivos de Flash 4 que utilizan el operador & se actualizan automáticamente para utilizar add cuando se traen al entorno de autoría de Flash 5.

&& (operador AND de cortocircuito)

Sintaxis
expresión1 && *expresión2*

Argumentos
expresión1, expresión2 Números, cadenas, variables o funciones.

Descripción
Operador (lógico); realiza una operación Booleana en los valores de una o ambas expresiones. Hace que el interpretador de Flash evalúe *expresión1* (la expresión de la izquierda) y devuelve `false` si la expresión evalúa como `false`. Si *expresión1* evalúa como `true`, *expresión2* (a la derecha) se evalúa. Si *expresión2* evalúa como `true`, el resultado final es `true`; en caso contrario, es `false`.

Reproductor
Flash 4 o posterior.

Ejemplo
Este ejemplo asigna los valores de las expresiones evaluadas a las variables `winner` y `loser` para realizar una comprobación:

```
winner = (chocolateEggs >=10) && (jellyBeans >=25);
loser = (chocolateEggs <=1) && (jellyBeans <= 5);
if (winner) {
   alert = "You Win the Hunt!";
   if (loser) {
      alert = "Now THAT'S Unhappy Hunting!";
   }
} else {
   alert = "We're all winners!";
}
```

&= (operador de asignación AND como bit)

Sintaxis
expresión1 &= *expresión2*

Argumentos
expresión1, expresión2 Números enteros y variables.

Descripción
Operador (de asignación como bit); asigna *expresión1* el valor de *expresión1* & *expresión2*.

Reproductor
Flash 5 o posterior.

Ejemplo
A continuación se muestra la utilización del operador `&=`con variables y números:

`x &= y` es lo mismo que `x = x & y`

Si `x = 15` y `y = 9` entonces

`x &= 9` devuelve `9`

Véase también
"& (operador AND como bit)" a pagina 195

() (paréntesis)

Sintaxis
```
(expresión1, expresión2);
function(functionCall1, ..., functionCallN);
```

Argumentos
expresión1, expresión2 Números, cadenas, variables o texto.

function La función que se va a realizar en el contenido de los paréntesis.

functionCall1...functionCallN Una serie de funciones que se ejecutan antes de que el resultado se pase a la función fuera de los paréntesis.

Descripción
Operador (general); realiza una operación de agrupamiento sobre uno o más argumentos, o rodea uno o más argumentos y pasa el resultado como parámetro a una función fuera de los paréntesis.

Sintaxis 1: Realiza una operación de agrupamiento sobre una o más expresiones para controlar el orden de ejecución de los operadores en la expresión. Este operador ignora el orden de precedencia automático y hace que las expresiones dentro de los paréntesis se evalúen primero. Cuando los paréntesis están anidados, Flash evalúa el contenido de los paréntesis interiores antes que el contenido de los exteriores.

Sintaxis 2: Rodea uno o más argumentos y los pasa como parámetro a la función de fuera de los paréntesis.

Reproductor
Flash 4 o posterior.

Ejemplo
(Sintaxis 1) Las sentencias siguientes muestran la utilización de los paréntesis para controlar el orden de ejecución de las expresiones. (El resultado aparece debajo de cada sentencia.)

```
(2 + 3) * (4 + 5)
45
2 + (3 * (4 + 5))
29
2 + (3 * 4) + 5
19
```

(Sintaxis 2) El ejemplo siguiente muestra la utilización de los paréntesis con una función:

```
getDate();
invoice(item, amount);
```

Véase también
"with" a pagina 389

- (menos)

Sintaxis
(Negación) –*expresión*

(Resta) *expresión1* - *expresión2*

Argumentos
expresión1, expresión2 Cualquier número.

Descripción
Operador (aritmético); utilizado para negación o resta. Cuando se utiliza para negar, invierte el signo de la *expresión* numérica. Cuando se utiliza para restar, realiza una resta aritmética sobre dos expresiones numéricas, restando *expresión2* de *expresión1*. Cuando ambas expresiones son números enteros, la diferencia es un número entero. Cuando una o ambas expresiones son números con coma flotante, la diferencia es un número con coma flotante.

Reproductor
Flash 4 o posterior.

Ejemplo
(Negación) Esta sentencia invierte el signo de la expresión 2 + 3:

```
-(2 + 3)
```

El resultado es -5.

(Resta) Esta sentencia resta el número entero 2 del número entero 5:

```
5 - 2
```

El resultado es 3, que es un número entero.

(Resta): Esta sentencia resta el número con coma flotante 1,5 del número con coma flotante 3,25:

```
put 3.25 - 1.5
```

El resultado es 1,75, que es un número con coma flotante.

* (multiplicación)

Sintaxis
*expresión1 * expresión2*

Argumentos
expresión1, expresión2 Números enteros o números con coma flotante.

Descripción
Operador (aritmético); multiplica dos expresiones numéricas. Cuando ambas expresiones son números enteros, el producto es un número entero. Cuando una o ambas expresiones son números con coma flotante, el producto es un número con coma flotante.

Reproductor
Flash 4 o posterior.

Ejemplo
Esta sentencia multiplica los números enteros 2 y 3:

```
2 * 3
```

El resultado es 6, que es un número entero.

Ejemplo
Esta sentencia multiplica los números con coma flotante 2,0 y 3,1416:

```
2.0 * 3.1416
```

El resultado es 6,2832, que es un número con coma flotante.

*= (asignación de multiplicación)

Sintaxis
*expresión1 *= expresión2*

Argumentos
expresión1, expresión2 Números enteros, números con coma flotante, o cadenas.

Descripción
Operador (de asignación); asigna a *expresión1* el valor de *expresión1 * expresión2*.

Reproductor
Flash 4 o posterior.

Ejemplo
A continuación se muestra la utilización del operador *= con variables y números:

`x *= y` es lo mismo que `x = x * y`

Si `x = 5` y `y = 10` entonces

`x *= 10` devuelve `50`

Véase también
"* (multiplicación)" a pagina 199

, (coma)

Sintaxis
expresión1, *expresión2*

Argumentos
expresión Cualquier número, variable, cadena, elemento de matriz u otros datos.

Descripción
Operador; da instrucciones a Flash para que evalúe *expresión1*, después *expresión2* y devuelva el valor de *expresión2*. Este operador se utiliza principalmente con la sentencia de bucle `for`.

Reproductor
Flash 4 o posterior.

Ejemplo
El siguiente código de muestra utiliza el operador coma:

```
var a=1, b=2, c=3;
```

Esto es equivalente a escribir lo que se muestra a continuación:

```
var a=1;
var b=2;
var c=3;
```

. (operador punto)

Sintaxis

objeto.propiedad_o_método
nombre_instancia.variable
nombre_instancia.instancia_secundaria.variable

Argumentos

objeto Una instancia de un objeto. Algunos objetos requieren que las instancias se creen utilizando el constructor de ese objeto. El objeto puede ser cualquiera de los objetos predefinidos de ActionScript o un objeto personalizado. Este argumento siempre se encuentra a la izquierda del operador punto (.).

propiedad_o_método El nombre de una propiedad o método asociado con un objeto. Todos los métodos y propiedades válidas de los objetos predefinidos aparecen en la lista en las tablas de resumen de Método y Propiedad de ese objeto. Este argumento siempre se encuentra a la derecha del operador punto (.).

nombre_instancia El nombre de una instancia de clip de película.

instancia_secundaria Una instancia de clip de película que secundaria del clip de película principal.

variable Una variable en un clip de película.

Descripción

Operador; utilizado para desplazarse por las jerarquías del clip de película para acceder a los clips de película anidados, a variables o a propiedades. El operador punto también se utiliza para comprobar o establecer las propiedades de un objeto, ejecutar un método de un objeto o crear una estructura de datos.

Reproductor

Flash 4 o posterior.

Véase también

"[] (operador de acceso a matriz)" a pagina 205

Ejemplo

Esta sentencia identifica el valor actual de la variable `hairColor` por el clip de película `person`:

```
person.hairColor
```

Esto es equivalente a la sintaxis de Flash 4 que se muestra a continuación:

```
/person:hairColor
```

Ejemplo

El código siguiente muestra como el operador punto puede utilizarse para crear una estructura de una matriz.

```
account.name = "Gary Smith";
account.address = "123 Main St ";
account.city = "Any Town";
account.state = "CA";
account.zip = "12345";
```

?: (condicional)

Sintaxis

expresión1 ? *expresión2* : *expresión3*

Argumentos

expresión1 Una expresión que evalúa como un valor Booleano, normalmente una expresión de comparación.

expresión2, expresión3 Valores de cualquier tipo.

Descripción

Operador (condicional); da instrucciones a Flash para que evalúe *expresión1*, y devuelva el valor de *expresión2* si *expresión1* es `true`; en caso contrario devuelva el valor de la *expresión3*.

Reproductor

Flash 4 o posterior.

/ (división)

Sintaxis

expresión1 / expresión2

Argumentos

expresión Cualquier número.

Descripción

Operador (aritmético); divide *expresión1* por *expresión2*. Los argumentos de la expresión y el resultado de la operación de división se tratan o expresan como números con coma flotante de doble precisión.

Reproductor

Flash 4 o posterior.

Ejemplo

Esta sentencia divide el número con coma flotante 22,0 por 7,0 y a continuación muestra el resultado en la ventana de Salida:

```
trace(22.0 / 7.0);
```

El resultado es 3,1429, que es un número con coma flotante.

// (delimitador de comentario)

Sintaxis

```
// comentario
```

Argumentos

comentario Texto que no es parte del código y el interpretador debería ignorar.

Descripción

Comentario; indica el comienzo de un comentario de script. Cualquier texto que aparezca entre el delimitador de comentario `//` y el carácter de final de línea se interpreta como un comentario y el interpretador de ActionScript lo ignora.

Reproductor

Flash 1 o posterior.

Ejemplo

Este script utiliza delimitadores de comentario de barra para identificar la primera, la tercera y la séptima línea como comentarios.

```
// set the X position of the ball movie clip
ball = getProperty(ball._x);
// set the Y position of the ball movie clip
ball = getProperty(ball._y);
// set the X position of the kitty movie clip
kitty = getProperty(kitty._x);
// set the Y position of the kitty movie clip
kitty_y = getProperty(kitty._y);
```

Véase también

"/* (delimitador de comentario)" a pagina 203

/* (delimitador de comentario)

Sintaxis

```
/* comentario */
/*
* comentario
* comentario
*/
```

Argumentos

comentario Cualquier texto

Descripción
Comentario; indica una o más líneas de comentarios del script. Cualquier texto que aparezca entre la etiqueta de apertura de comentario `/*` y la etiqueta de cierre de comentario `*/`, se interpreta como un comentario y el interpretador de ActionScript lo ignora. Utilice la primera sintaxis para identificar comentarios de una sola línea y utilice la segunda sintaxis para identificar comentarios de varias líneas sucesivas. Si no se incluye la etiqueta de cierre `*/` cuando se utiliza esta forma de delimitador de comentario, hace que el compilador de ActionScript devuelva un mensaje de error.

Reproductor
Flash 5 o posterior.

Véase también
"// (delimitador de comentario)" a pagina 203

/= (asignación de división)

Sintaxis
expresión1 /= expresión2

Argumentos
expresión1, expresión2 Números enteros, números con coma flotante, o cadenas.

Descripción
Operador (de asignación); asigna a *expresión1* el valor de *expresión1 / expresión2*.

Reproductor
Flash 4 o posterior.

Ejemplo
A continuación se muestra la utilización del operador `/=` con variables y números:

```
x /= y es lo mismo que x = x /y
x = 10;
y = 2;
x /= y;
// x now contains the value 5
```

[] (operador de acceso a matriz)

Sintaxis

```
myArray["a0", "a1",..."aN"];
object[value1, value2, ...valueN];
```

Argumentos

myArray El nombre de una matriz.

a0, a1,...aN Elementos en una matriz.

value1, 2,...N Nombres de propiedades.

Descripción

Operador; crea un nuevo objeto que inicializa las propiedades especificadas en los argumentos o inicializa una nueva matriz con los elementos (*a0*) especificados en los argumentos.

El objeto creado tiene un objeto `Objeto` genérico como su prototipo. Utilizar este operador es lo mismo que llamar a `new Object` y rellenar las propiedades utilizando el operador de asignación. Utilizar este operador es una alternativa a utilizar el operador `new`, que permite la creación rápida y práctica de objetos.

Reproductor

Flash 4 o posterior.

Ejemplo

Las siguientes muestras de código de ejemplo son dos modos diferentes de crear un nuevo objeto Array vacío.

```
myArray =[];
myArray = new Array();
```

A continuación se muestra un ejemplo de una matriz simple.

```
myArray = ["red", "orange", "yellow", "green", "blue", "purple"]
myArray[0]="red"
myArray[1]="yellow"
myArray[2]="green"
myArray[3]="blue"
myArray[4]="purple"
```

^(operador XOR como bit)

Sintaxis
expresión1 ^ *expresión2*

Argumentos
expresión1, expresión2 Cualquier número.

Descripción
Operador (como bit); convierte *expresión1* y *expresión2* en números enteros de 32 bits no firmados y devuelve un 1 en cada posición de bit donde los bits correspondientes en *expresión1* o *expresión1*, pero no ambos, son 1.

Reproductor
Flash 5 o posterior.

Ejemplo
```
15 ^ 9 returns 6
(1111 ^ 1001 = 0110)
```

^= (operador de asignación XOR como bit)

Sintaxis
expresión1 ^= expresión2

Argumentos
expresión1, expresión2 Números enteros y variables.

Descripción
Operador (de asignación como bit); asigna a *expresión1* el valor de *expresión1* ^ *expresión2*.

Reproductor
Flash 5 o posterior.

Ejemplo
A continuación se muestra un ejemplo de una operación ^=.

```
// 15 decimal = 1111 binary
x = 15;
// 9 decimal = 1001 binary
x ^= y;
returns
x ^ y (0110 binary)
```

A continuación se muestra la utilización del operador ^= con variables y números:

```
x ^= y es lo mismo que x = x ^ y
Si x = 15  y y = 9 entonces
15 ^= 9 devuelve 6
```

Véase también
"^(operador XOR como bit)" a pagina 206

{} (inicializador de objeto)

Sintaxis

```
object {name1: value1,
   name1: value2,
...
   nameN: valueN };
```

Argumentos

object El objeto que se va a crear.

name1,2,...N El nombre de la propiedad.

value1,2,...N El valor correspondiente para cada propiedad *name*.

Descripción

Operador; crea un nuevo objeto y lo inicializa con el *name* especificado y los pares de propiedades *value*. El objeto creado tiene un objeto `Objeto` genérico como su prototipo. Utilizar este operador es lo mismo que llamar a `new Object` y rellenar las propiedades utilizando el operador de asignación. Utilizar este operador es una alternativa a utilizar el operador `new`, que permite la creación rápida y práctica de objetos.

Reproductor

Flash 5 o posterior.

Ejemplo

El código siguiente muestra como puede crearse un objeto vacío utilizando el operador inicializador de objeto y utilizando el `new Object`.

```
object = {};
object = new Object();
```

El ejemplo siguiente crea un objeto `account` que inicializa las propiedades `name`, `address`, `city`, `state`, `zip`, y `balance`.

```
account = { name: "John Smith",
   address: "123 Main Street",
   city: "Blossomville",
   state: "California",
   zip: "12345",
   balance: "1000" };
```

El ejemplo siguiente muestra como los inicializadores de matriz y de objeto pueden anidarse unos dentro de otros.

```
person = { name: "Peter Piper",
   children: [ "Jack", "Jill", "Moe",] };
```

El ejemplo siguiente muestra otro modo de utilizar la información del ejemplo anterior, con los mismos resultados.

```
person = new Person();
person.name = 'John Smith';
person.children = new Array();
person.children[0] = 'Jack';
person.children[1] = 'Jill';
person.children[2] = 'Moe';
```

Véase también

"[] (operador de acceso a matriz)" a pagina 205
"new" a pagina 328
"Object (objeto)" a pagina 337

| (operador OR como bit)

Sintaxis

expresión1|*expresión2*

Argumentos

expresión1, expresión2 Cualquier número.

Descripción

Operador (como bit); convierte *expresión1* y *expresión2* en números enteros de 32 bits no firmados y devuelve un 1 en cada posición de bit donde los bits correspondientes en *expresión1* o *expresión2* son 1.

Reproductor

Flash 5 o posterior.

Ejemplo

A continuación se muestra un ejemplo de una operación OR como bit. Observe que 15 es binario 1111.

```
// 15 decimal = 1111 binary
x = 15;
// 9 decimal = 1001 binary
y = 9;
// x | y = binary
z = x | y;
z = 15
```

A continuación se muestra otro modo de expresar el ejemplo anterior.

```
15 | 9 returns 15
(1111 | 0011 = 1111)
```

|| (operador OR)

Sintaxis
expresión1 || *expresión2*

Argumentos
expresión1, *expresión2* Un valor o expresión Booleana que convierte aun valor Booleano.

Descripción
Operador (lógico); evalúa *expresión1* y *expresión2*. El resultado es (true) si cualquiera de ellas o ambas expresiones evalúan como true; el resultado es (false) solamente si ambas expresiones evalúan como false.

Con expresiones no Booleanas, el operador lógico OR hace que Flash evalúe la expresión de la izquierda; si puede convertirse en true, el resultado es true. En caso contrario, evalúa la expresión de la derecha y el resultado es el valor de esa expresión.

Reproductor
Flash 4 o posterior.

Ejemplo
El ejemplo siguiente utiliza el operador || en una sentencia if:

```
want = true;
need = true;
love = false;
if (want || need || love){
trace("two out of 3 ain't bad");
}
```

|= (operador de asignación OR como bit)

Sintaxis
expresión1 |= *expresión2*

Argumentos
expresión1, *expresión2* Números enteros y variables.

Descripción
Operador (de asignación); asigna a *expresión1* el valor de *expresión1* | *expresión2*.

Reproductor
Flash 5 o posterior.

Ejemplo
A continuación se muestra la utilización del operador |= con variables y números:

```
x |= y es lo mismo que x = x | y
Si x = 15  y y = 9 entonces
x |= 9 devuelve 15
```

Véase también
"| (operador OR como bit)" a pagina 208

~ (operador NOT como bit)

Sintaxis
~ *expresión*

Argumentos
expresión Cualquier número.

Descripción
Operador (como bit); convierte la *expresión* en un número entero de 32 bits no firmado, después invierte los bits. O bien, dicho de modo sencillo, cambia el signo de un número y le resta 1.

Una operación de NOT como bit cambia el signo de un número y le resta 1.

Reproductor
Flash 5 o posterior.

Ejemplo
A continuación se muestra una explicación numérica de una operación de NOT como bit realizada en una variable:

```
~a, returns -1 if a = 0, and returns -2 if a = 1, thus:
~0=-1 and ~1=-2
```

+ (suma)

Sintaxis
expresión1 + expresión2

Argumentos
expresión1, expresión2 Números enteros, números, números con coma flotante, o cadenas.

Descripción
Operador; agrega expresiones numéricas o concatena cadenas. Si una expresión es una cadena, todas las demás expresiones se convierten en cadenas y se concatenan.

Si ambas expresiones son números enteros, la suma es un número entero, si cualquiera de ellas o ambas expresiones son números con coma flotante, la suma es un número con coma flotante.

Reproductor
Flash 4; Flash 5 o posterior. En Flash 5, + es un operador numérico o concatenador de cadena dependiendo del tipo de datos del argumento. En Flash 4, + es solamente un operador numérico. Los archivos de Flash 4 que se incorporen a un entorno de creación de Flash 5 sufren un proceso de conversión para mantener la integridad de los tipos de datos. El primer ejemplo a continuación muestra el proceso de conversión.

Ejemplo
A continuación se muestra un ejemplo de conversión de un archivo de Flash 4 que contenga una comparación de cualidad numérica.

Archivo de Flash 4:

```
x + y
```

Archivo de Flash 5 convertido:

```
Number(x) + Number(y)
```

Esta sentencia agrega los números enteros 2 y 3 y después muestra el número entero 5 resultante, en la ventana de Salida.

```
trace (2 + 3);
```

Esta sentencia añade los números con coma flotante 2,5 y 3,25 y muestra el resultado, 5,7500, un número con coma flotante, en la ventana de Salida:

```
trace (2.5 + 3.25);
```

Esta sentencia concatena dos cadenas y muestra el resultado, "today is my birthday," en la ventana de Salida.

```
"today is my" + "birthday"
```

Véase también
"add" a pagina 223

+= (asignación de suma)

Sintaxis
expresión1 += *expresión2*

Argumentos
expresión1, expresión2 Números enteros, números con coma flotante o cadenas.

Descripción
Operador (de asignación compuesta); asigna a *expresión1* el valor de *expresión1* + *expresión2*. Este operador también realiza la concatenación de cadenas.

Reproductor
Flash 4 o posterior.

Ejemplo
El ejemplo siguiente muestra una utilización numérica del operador +=:

```
x += y es lo mismo que x = x + y
If x = 5 and y = 10 then
x += 10 returns 15
```

Este ejemplo muestra la utilización del operador += con una expresión de cadena.

```
x = "My name is"
x += "Mary"
```

El resultado del código anterior es lo que se muestra a continuación:

```
"My name is Mary"
```

Véase también
"+ (suma)" a pagina 210

< (menor que)

Sintaxis
expresión1 < *expresión2*

Argumentos
expresión1, expresión2 Números o cadenas.

Descripción
Operador (de comparación); compara dos expresiones y determina si *expresión1* es menor que *expresión2* (`true`), o si *expresión1* es mayor o igual que *expresión2* (`false`). Las expresiones de cadena se evalúan y comparan basándose en el número de caracteres de la cadena.

Reproductor
Flash 4; Flash 5 o posterior. En Flash 5 < es un operador de comparación capaz de manejar varios tipos de datos. En Flash 4, < es un operador numérico. Los archivos de Flash 4 que se incorporen a un entorno de creación de Flash 5 sufren un proceso de conversión para mantener la integridad de los tipos de datos. El primer ejemplo a continuación muestra el proceso de conversión.

Ejemplo

A continuación se muestra un ejemplo de conversión de un archivo de Flash 4 que contenga una comparación de cualidad numérica.

Archivo de Flash 4:

```
x < y
```

Archivo de Flash 5 convertido:

```
Number(x) < Number(y)
```

El ejemplo siguiente muestra las devoluciones de `true` y `false` tanto para cadenas como para números:

```
3 < 10 or "Al" < "Jack" return true
10 < 3 or "Jack" < "Al" return false
```

<< (desplazamiento a la izquierda como bit)

Sintaxis

expresión1 << *expresión2*

Argumentos

expresión1 Un número, cadena o expresión que se va a desplazar a la izquierda.

expresión2 Un número, cadena o expresión que se convierte en un número entero entre 0 y 31.

Descripción

Operador (como bit); convierte *expresión1* y *expresión2* en números enteros de 32 bits y desplaza todos los bits de *expresión1* hacia la izquierda el número de espacios especificado por el número entero que resulta de la conversión de *expresión2*. Las posiciones de bit que se vacían como resultado de esta operación se rellenan con 0. Desplazar un valor a la izquierda 1 posición es el equivalente de multiplicarlo por 2.

Reproductor

Flash 5 o posterior.

Ejemplo
El ejemplo siguiente desplaza el número entero `1` diez bits a la izquierda.

```
x = 1 << 10
```

El resultado de esta operación es `x = 1024`. Esto es debido a que 1 decimal es igual a 1 binario, 1 binario desplazado a la izquierda por 10 es 10000000000 binario y 10000000000 binario es 1024 decimal.

El ejemplo siguiente desplaza el número entero `7` ocho bits a la izquierda.

```
x = 7 << 8
```

El resultado de esta operación es `x = 1792`. Esto es debido a que 7 decimal es igual a 111 binario, 111 binario desplazado a la izquierda por 8 es 11100000000 binario y 11100000000 binario es 1792 decimal.

Véase también
"`>>= (desplazamiento a la derecha como bit y asignación)`" a pagina 221

<< (desplazamiento a la izquierda como bit y asignación)

Sintaxis
expresión1 <<= *expresión2*

Argumentos
expresión1 Un número, cadena o expresión que se va a desplazar a la izquierda.

expresión2 Un número, cadena o expresión que se convierte en un número entero entre 0 y 31.

Descripción
Operador (de asignación compuesta); este operador realiza una operación de desplazamiento a la izquierda como bit y almacena el contenido como resultado de *expresión1*.

Reproductor
Flash 5 o posterior.

Ejemplo
Las dos expresiones siguientes son equivalentes.

```
A <<= B
A = (A << B)
```

Véase también
"`<< (desplazamiento a la izquierda como bit)`" a pagina 213
"`>>= (desplazamiento a la derecha como bit y asignación)`" a pagina 221

<= (menor o igual que)

Sintaxis

expresión1 <= *expresión2*

Argumentos

expresión1, expresión2 Números o cadenas.

Descripción

Operador (de comparación); compara dos expresiones y determina si *expresión1* es menor o igual que *expresión2* (`true`), o si *expresión1* es mayor que *expresión2* (`false`).

Reproductor

Flash 4; Flash 5 o posterior. En Flash 5 <= es un operador de comparación capaz de manejar varios tipos de datos. En Flash 4, <= es un operador numérico. Los archivos de Flash 4 que se incorporen a un entorno de creación de Flash 5 sufren un proceso de conversión para mantener la integridad de los tipos de datos. El primer ejemplo a continuación muestra el proceso de conversión.

Ejemplo

A continuación se muestra un ejemplo de conversión de un archivo de Flash 4 que contenga una comparación de cualidad numérica.

Archivo de Flash 4:

```
x <= y
```

Archivo de Flash 5 convertido:

```
Number(x) <= Number(y)
```

El ejemplo siguiente muestra los resultados de `true` y `false` tanto para cadenas como para números:

```
5 <= 10 or "Al" <= "Jack" returns true
10<= 5 or "Jack" <= "Al" returns false
```

<> (no igualdad)

Sintaxis

expresión1 <> *expresión2*

Argumentos

expresión1, expresión2 Números, cadenas, Booleanos, variables, objetos, matrices o funciones.

Descripción
Operador (igualdad); comprueba el opuesto exacto del operador ==. Si *expresión1* es igual a *expresión2*, el resultado es `false`. Al igual que con el operador ==, la definición de *igual* depende de los tipos de datos que se comparan.

- Los números, cadenas y los valores Booleanos se comparan por valor.
- Las variables, objetos, matrices y funciones se comparan por referencia.

Este operador se ha desestimado en Flash 5 y se recomienda a los usuarios que utilicen el nuevo operador `!=`.

Reproductor
Flash 2 o posterior.

Véase también
"!= (no igualdad)" a pagina 193

= (asignación)

Sintaxis
expresión1 = expresión2

Argumentos
expresión1 Una variable, elemento de una matriz o la propiedad de un objeto.

expresión2 Un valor de cualquier tipo.

Descripción
Operador (de asignación); asigna el tipo de *expresión2* (el argumento de la derecha) a la variable, elemento de matriz o propiedad en *expresión1*.

Reproductor
Flash 4; Flash 5 o posterior. En Flash 5 = es un operador de asignación y el operador == se utiliza para evaluar la igualdad. En Flash 4, = es un operador de igualdad numérico. Los archivos de Flash 4 que se incorporen a un entorno de creación de Flash 5 sufren un proceso de conversión para mantener la integridad de los tipos de datos. El primer ejemplo a continuación muestra el proceso de conversión.

Ejemplo
A continuación se muestra un ejemplo de conversión de un archivo de Flash 4 que contenga una comparación de cualidad numérica.

Archivo de Flash 4:

```
x = y
```

Archivo de Flash 5 convertido:

```
Number(x) == Number(y)
```

El ejemplo siguiente utiliza el operador de asignación para asignar el tipo de datos número a la variable `x`.

```
x = 5
```

El ejemplo siguiente utiliza el operador de asignación para asignar el tipo de datos cadena a la variable `x`.

```
x = "hello"
```

-= (asignación de negación)

Sintaxis
expresión1 -= expresión2

Argumentos
expresión1, expresión2 Números enteros, números con coma flotante o cadenas.

Descripción
Operador (de asignación compuesta); asigna a *expresión1* el valor de *expresión1 - expresión2*.

Reproductor
Flash 4 o posterior.

Ejemplo
A continuación se muestra la utilización del operador `-=` con variables y números:

```
x -= y es lo mismo que x = x - y
Si x = 5 y y = 10entonces
x -= 10 devuelve -5
```

== (igualdad)

Sintaxis
expresión1 == *expresión2*

Argumentos
expresión1, expresión2 Números, cadenas, Booleanos, variables, objetos, matrices o funciones.

Descripción
Operador (de igualdad); comprueba la igualdad de dos expresiones. El resultado es `true` si las expresiones son iguales.

La definición de *igual* depende del tipo de datos del argumento:

- Los números, cadenas y valores Booleanos se comparan por valor y se consideran igual si tienen el mismo valor. Por ejemplo, dos cadenas son iguales si tienen el mismo número de caracteres.
- Las variables, objetos, matrices y funciones se comparan por referencia. Dos variables son iguales si se refieren al mismo objeto, matriz o función. Dos matrices independientes nunca se consideran iguales, incluso aunque tengan el mismo número de elementos.

Reproductor
Flash 5 o posterior.

Ejemplo
El ejemplo siguiente utiliza el operador == con una sentencia `if`:

```
a = "David" , b = "David";
if (a == b)
trace("David is David");
```

> (mayor que)

Sintaxis
expresión1 > *expresión2*

Argumentos
expresión1,*expresión2* Números enteros, números con coma flotante o cadenas.

Descripción
Operador (de comparación); compara dos expresiones y determina si *expresión1* es mayor que *expresión2* (`true`), o si *expresión1* es menor o igual que *expresión2* (`false`).

Reproductor
Flash 4; Flash 5 o posterior. En Flash 5 > es un operador de comparación capaz de manejar varios tipos de datos. En Flash 4, > es un operador numérico. Los archivos de Flash 4 que se incorporen a un entorno de creación de Flash 5 sufren un proceso de conversión para mantener la integridad de los tipos de datos. El ejemplo a continuación muestra el proceso de conversión.

Ejemplo
A continuación se muestra un ejemplo de conversión de un archivo de Flash 4 que contenga una comparación de cualidad numérica.

Archivo de Flash 4:

```
x > y
```

Archivo de Flash 5 convertido:

```
Number(x) > Number(y)
```

>= (mayor o igual que)

Sintaxis
expresión1 `>=` *expresión2*

Argumentos
expresión1, expresión2 Números enteros o números con coma flotante.

Descripción
Operador (de comparación); compara dos expresiones y determina si *expresión1* es mayor o igual que *expresión2* (`true`), o si *expresión1* es menor que *expresión2* (`false`).

Reproductor
Flash 4; Flash 5 o posterior. En Flash 5 >= es un operador de comparación capaz de manejar varios tipos de datos. En Flash 4, >= es un operador numérico. Los archivos de Flash 4 que se incorporen a un entorno de creación de Flash 5 sufren un proceso de conversión para mantener la integridad de los tipos de datos. El ejemplo a continuación muestra el proceso de conversión.

Ejemplo
A continuación se muestra un ejemplo de conversión de un archivo de Flash 4 que contenga una comparación de cualidad numérica.

Archivo de Flash 4:

```
x >= y
```

Archivo de Flash 5 convertido:

```
Number(x) >= Number(y)
```

>> (desplazamiento a la derecha como bit)

Sintaxis

expresión1 >> *expresión2*

Argumentos

expresión1 Un número, cadena o expresión que se va a desplazar a la derecha.

expresión2 Un número, cadena o expresión que se convierte en un número entero entre 0 y 31.

Descripción

Operador (como bit); convierte *expresión1* y *expresión2* en números enteros de 32 bits y desplaza todos los bits de *expresión1* hacia la derecha el número de espacios especificado por el número entero que resulta de la conversión de *expresión2*. Los bits que se desplazan a la derecha se descartan. Para preservar el signo de la *expresión* original, los bits a la izquierda se rellenan con 0 si el bit más significativo (el bit más a la izquierda) de *expresión1* es 0 y se rellena con 1 si el bit más significativo es 1. Desplazar un valor a la derecha en 1 posición es equivalente a dividir por 2 y descartar el resto.

Reproductor

Flash 5 o posterior.

Ejemplo

El ejemplo siguiente convierte 65535 en un número entero de 32 bits y lo desplaza ocho bits a la derecha.

```
x = 65535 >> 8
```

El resultado de la operación anterior es lo que se muestra a continuación:

```
x = 255
```

Esto es debido a que 65535 decimal es igual a 1111111111111111 binario (*dieciséis unos*), 1111111111111111 binario desplazado a la derecha por ocho bits es 11111111 binario y11111111 binario es 255 decimal. El bit más significativo es 0 debido a que los números enteros son de 32 bits, así que el bit de relleno es 0.

El ejemplo siguiente convierte -1 en un número entero de 32 bits y lo desplaza un bit a la derecha.

```
x = -1 >> 1
```

El resultado de la operación anterior es lo que se muestra a continuación:

```
x = -1
```

Esto es debido a que -1 decimal es igual a 11111111111111111111111111111111 binario (treinta y dos unos), desplazar a la derecha un bit hace que se descarte el bit menos significativo (el bit más a la derecha) y que el bit más significativo se rellene con 1. El resultado es 11111111111111111111111111111111 (*treinta y dos unos*) binario, que representa el número entero de 32 bits -1.

Véase también

"">>= (desplazamiento a la derecha como bit y asignación)" a pagina 221

>>= (desplazamiento a la derecha como bit y asignación)

Sintaxis

expresión1 =>> expresión2

Argumentos

expresión1 Un número, cadena o expresión que se va a desplazar a la izquierda.

expresión2 Un número, cadena o expresión que se convierte en un número entero entre 0 y 31.

Descripción

Operador (de asignación compuesta); este operador realiza una operación de desplazamiento a la derecha como bit y almacena el contenido como resultado de *expresión1*.

Reproductor

Flash 5 o posterior.

Ejemplo

Las dos expresiones siguientes son equivalentes.

```
A >>= B
A = (A >> B)
```

El siguiente código comentado utiliza el operador como bit >>=. También es un ejemplo de la utilización de todos los operadores como bit.

```
function convertToBinary(number)
{
var result = "";
for (var i=0; i<32; i++) {
// Extract least significant bit using bitwise AND
var lsb = number & 1;
// Add this bit to our result string
result = (lsb ? "1" : "0") + result;
// Shift number right by one bit, to see next bit
}number >>= 1;
return result;
}
convertToBinary(479)
//Returns the string
00000000000000000000000111011111
//The above string is the binary representation of the decimal
number 479.
```

Véase también

"<< (desplazamiento a la izquierda como bit)" a pagina 213

>>> (desplazamiento a la derecha como bit sin signo)

Sintaxis

expresión1 >>> *expresión2*

Argumentos

expresión1 Un número, cadena o expresión que se va a desplazar a la derecha.

expresión2 Un número, cadena o expresión que se convierte en un número entero entre 0 y 31.

Descripción

Operador (como bit); es lo mismo que el operador de desplazamiento a la derecha como bit(>>) excepto en que no preserva el signo de la *expresión* original debido a que los bits de la izquierda siempre se rellenan con 0.

Reproductor

Flash 5 o posterior.

Ejemplo

El ejemplo siguiente convierte -1 en un número entero de 32 bits y lo desplaza un bit a la derecha.

```
x = -1 >>> 1
```

El resultado de la operación anterior es lo que se muestra a continuación:

```
x = 2147483647
```

Esto es debido a que -1 decimal es 11111111111111111111111111111111 binario (*treinta y dos unos*) y cuando desplaza a la derecha (sin signo) en un bit, el bit menos significativo (el más a la derecha) se descarta y el más bit significativo (el más a la izquierda) se rellena con un 0. El resultado es:

01111111111111111111111111111111 binario,

que representa el número entero de 32 bits `2147483647`.

Véase también

"`>>= (desplazamiento a la derecha como bit y asignación)`" a pagina 221

>>>= (desplazamiento a la derecha como bit sin signo y asignación)

Sintaxis

expresión1 >>>= *expresión2*

Argumentos

expresión1 Un número, cadena o expresión que se va a desplazar a la izquierda.

expresión2 Un número, cadena o expresión que se convierte en un número entero entre 0 y 31.

Descripción
Operador (de asignación compuesta); este operador realiza una operación de desplazamiento a la derecha como bit y almacena el contenido como resultado en *expresión1*.

Reproductor
Flash 5 o posterior.

Ejemplo
Las dos expresiones siguientes son equivalentes.

```
A >>>= B
A = (A >>> B)
```

Véase también
"`>>> (desplazamiento a la derecha como bit sin signo)`" a pagina 222
"`>>= (desplazamiento a la derecha como bit y asignación)`" a pagina 221

add

Sintaxis
cadena1 `add` *cadena2*

Argumentos
cadena1, 2 Cualquier cadena.

Descripción
Operador; concatena dos o más cadenas. El operador `add` sustituye al operador `&` de Flash 4; los archivos de Flash 4 que utilizan el operador `&` se convierten automáticamente para utilizar el operador `add` para concatenación de cadenas cuando se incorporan a un entorno de creación de Flash 5. Sin embargo, el operador `add` se desestima en Flash 5 y se recomienda la utilización del operador `+` cuando se crea contenido para Flash Player 5. Utilice el operador `add` para concatenar cadenas si está creando contenido para Flash 4 o para versiones anteriores del Reproductor.

Reproductor
Flash 4 o posterior.

Véase también
"`+ (suma)`" a pagina 210

_alpha

Sintaxis

```
nombre_instancia._alpha
nombre_instancia._alpha = valor;
```

Argumentos

nombre_instancia El nombre de una instancia de clip de película.

valor Un número de 0 a 100 que especifica la transparencia alfa.

Descripción

Propiedad; establece o recupera la transparencia alfa (*valor*) del clip de película. Los valores válidos son de 0 (completamente transparente) a 100 (completamente opaco). Los objetos de un clip de película con `_alpha` establecido en 0 son activos, aunque sean invisibles. Por ejemplo, en un botón de un clip de película con la propiedad `_alpha` establecida en 0 aún puede hacerse clic.

Reproductor

Flash 4 o posterior.

Ejemplo

Las sentencias siguientes establecen la propiedad `_alpha` de un clip de película llamado `star` en el 30% cuando se hace clic sobre el botón.

```
on(release) {
   setProperty(star._alpha = 30);
}
```

o

```
on(release) {
   star._alpha = 30;
}
```

and

Sintaxis

condición1 `and` *condición2*

Argumentos

condición1, condición2 Condiciones o expresiones que evalúan como `true` o `false`.

Descripción

Operador; realiza una operación AND lógica en Flash Player 4. Si ambas expresiones evalúan como `true`, entonces toda la expresión es `true`.

Reproductor

Flash 4 o posterior. Este operador se ha desestimado en Flash 5 y se recomienda a los usuarios que utilicen el nuevo operador `&&`.

Véase también

"&& (operador AND de cortocircuito)" a pagina 196

Array (objeto)

El objeto Array le permite acceder a matrices y manipularlas. Una matriz es un objeto cuyas propiedades se identifican con un número que representa su posición en la matriz. A veces se hace referencia al número como el índice. Todas las matrices, que tienen base cero, lo que quiere decir que el primer elemento de la matriz es [0], el segundo elemento es [1] y así sucesivamente. En el ejemplo siguiente, `myArray` contiene los meses del año, identificados por número.

```
myArray[0] = "January"
myArray[1] = "February"
myArray[2] = "March"
myArray[3] = "April"
```

Para crear un objeto Array, utilice el constructor `new Array`. Para acceder a los elementos de una matriz utilice el operador de acceso a matriz [ ].

Resumen de los métodos de un objeto Array

Método	Descripción
`concat`	Concatena argumentos y los devuelve como una nueva matriz.
`join`	Une todos los elementos de una matriz en una cadena.
`pop`	Elimina el último elemento de una matriz y devuelve su valor.
`push`	Agrega uno o más elementos al final de una matriz y devuelve la nueva longitud de la matriz.
`reverse`	Invierte la dirección de una matriz.
`shift`	Elimina el primer elemento de una matriz y devuelve su valor.
`slice`	Extrae una sección de una matriz y lo devuelve como una nueva matriz.
`sort`	Ordena una matriz en contexto.
`splice`	Agrega y/o elimina elementos de una matriz.
`toString`	Devuelve un valor de cadena que representa los elementos de un objeto Array.
`unshift`	Agrega uno o más elementos al principio de una matriz y devuelve la nueva longitud de la matriz.

Resumen de las propiedades de un objeto Array

Propiedad	Descripción
`length`	Devuelve la longitud de la matriz.

Constructor del objeto Array

Sintaxis

```
new Array();
new Array(longitud);
new Array(elemento0, elemento1, elemento2,...elementoN);
```

Argumentos

`longitud` Un número entero que especifica el número de elementos de la matriz. En el caso de elementos no contiguos, la longitud especifica el número de índice del último elemento de la matriz más 1. Para obtener más información, véase la propiedad `Array.length`.

`elemento0...elementoN` Una lista de dos o más valores arbitrarios. Los valores pueden ser números, nombres u otros elementos especificados en una matriz. El primer elemento de una matriz siempre tiene el índice o posición 0.

Descripción

Constructor; le permite acceder y manipular los elementos de una matriz. Las matrices tienen base cero y sus elementos están indexados por su número ordinal.

Si no especifica ningún argumento, se crea una matriz de longitud cero.

Reproductor

Flash 5 o posterior.

Ejemplo

El ejemplo siguiente crea un nuevo objeto Array con una longitud inicial de 0.

```
myArray = new Array();
```

El ejemplo siguiente crea un nuevo objeto Array `A-Team`, con una longitud inicial de 4.

```
A-Team = new Array("Jody", "Mary", "Marcelle", "Judy");
```

Los elementos iniciales de la matriz `A-Team` son los que se muestran a continuación:

```
myArray[0] = "Jody"
myArray[1] = "Mary"
myArray[2] = "Marcelle"
myArray[3] = "Judy"
```

Véase también

"Array.length" a pagina 228

Array.concat

Sintaxis

myArray.concat(*valor0, valor1,...valorN*);

Argumentos

valor0,...valorN Números, elementos o cadenas que se van a concatenar en una nueva matriz.

Descripción

Método; concatena los elementos especificados en los argumentos, si hay alguno y crea y devuelve una nueva matriz. Si los argumentos especifican una matriz, los elementos de esa matriz se concatenan, en lugar de la propia matriz.

Reproductor

Flash 5 o posterior.

Ejemplo

El código siguiente concatena dos matrices:

```
alpha = new Array("a","b","c");
numeric = new Array(1,2,3);
alphaNumeric=alpha.concat(numeric); // creates array
["a","b","c",1,2,3]
```

El código siguiente concatena tres matrices:

```
num1=[1,3,5];
num2=[2,4,6];
num3=[7,8,9];
nums=num1.concat(num2,num3) // creates array [1,3,5,2,4,6,7,8,9]
```

Array.join

Sintaxis

myArray.join();

myArray.join(*separador*);

Argumentos

separador Un carácter o cadena que separa los elementos de la matriz en la cadena devuelta. Si omite este argumento, se utiliza una coma como separador predeterminado.

Descripción

Método; convierte los elementos de una matriz en cadenas, los concatena, inserta el separador especificado entre los elementos y devuelve la cadena resultante.

Reproductor
Flash 5 o posterior.

Ejemplo
El ejemplo siguiente crea una matriz, con tres elementos. Entonces une la matriz tres veces: utilizando el separador predeterminado, después una coma y un espacio y después un signo más.

```
a = new Array("Earth","Moon","Sun")
// assigns "Earth,Moon,Sun" to myVar1
myVar1=a.join();
// assigns "Earth, Moon, Sun" to myVar2
myVar2=a.join(", ");
// assigns "Earth + Moon + Sun" to myVar3
myVar3=a.join(" + ");
```

Array.length

Sintaxis
myArray.length;

Argumentos
Ninguno.

Descripción
Propiedad; contiene la longitud de la matriz. Esta propiedad se actualiza automáticamente cuando se agregan nuevos elementos a la matriz. Durante la asignación `myArray[index] = value`; si `index` es un número y `index+1` es mayor que la propiedad `length`, la propiedad `length` se actualiza a `index + 1`.

Reproductor
Flash 5 o posterior.

Ejemplo
El código siguiente explica como se actualiza la propiedad length.

```
//initial length is 0
myArray = new Array();
//myArray.length is updated to 1
myArray[0] = 'a'
//myArray.length is updated to 2
myArray[1] = 'b'
//myArray.length is updated to 10
myArray[9] = 'c'
```

Array.pop

Sintaxis
myArray.pop();

Argumentos
Ninguno.

Descripción
Método; elimina el último elemento de una matriz y devuelve el valor de ese elemento.

Reproductor
Flash 5 o posterior.

Ejemplo
El código siguiente crea la matriz `myPets` que contiene cuatro elementos, después elimina su último elemento.

```
myPets = ["cat", "dog", "bird", "fish"];
popped = myPets.pop();
```

Array.push

Sintaxis
myArray.push(*valor,...*);

Argumentos
valor Uno o más valores que se anexan a la matriz.

Descripción
Método; agrega uno o más elementos al final de una matriz y devuelve la nueva longitud de la matriz.

Reproductor
Flash 5 o posterior.

Ejemplo
El código siguiente crea la matriz `myPets` que contiene dos elementos, después le agrega dos elementos. Después de que se ejecute el código, `pushed` contiene 4.

```
myPets = ["cat", "dog"];
pushed = myPets.push("bird", "fish")
```

Array.reverse

Sintaxis
myArray.reverse();

Argumentos
Ninguno.

Descripción
Método; invierte la matriz en contexto.

Reproductor
Flash 5 o posterior.

Ejemplo
A continuación se muestra un ejemplo de la utilización del método `Array.reverse`.

```
var numbers = [1, 2, 3, 4, 5, 6];
trace(numbers.join())
 numbers.reverse()
 trace(numbers.join())
```

Salida:

```
1,2,3,4,5,6
6,5,4,3,2,1
```

Array.shift

Sintaxis
myArray.shift();

Argumentos
Ninguno.

Descripción
Método; elimina el primer elemento de una matriz y devuelve ese elemento.

Reproductor
Flash 5 o posterior.

Ejemplo
El código siguiente crea la matriz `myPets` y después elimina el primer elemento de la matriz:

```
myPets = ["cat", "dog", "bird", "fish"];
shifted = myPets.shift();
```

El valor devuelto es `cat`.

Véase también
"Array.pop" a pagina 229

"Array.unshift" a pagina 234

Array.slice

Sintaxis

```
myArray.slice(inicio, fin);
```

Argumentos

inicio Un número que especifica el índice del punto de inicio del sector. Si *inicio* es un número negativo, el punto de inicio comienza al final de la matriz, donde -1 es el último elemento.

fin Un número que especifica el índice del punto final del sector. Si omite este argumento, el sector incluirá todos los elementos desde el inicio al final de la matriz. Si *fin* es un número negativo, el punto final se especifica desde el final de la matriz, donde -1 es el último elemento.

Descripción

Método; extrae un sector o una subcadena de la matriz y la devuelve como una nueva matriz sin modificar la matriz original. La matriz devuelta incluye el elemento *inicio* y todos los elementos hasta el elemento *fin*, pero sin incluirlo.

Reproductor

Flash 5 o posterior.

Array.sort

Sintaxis

```
myArray.sort();
myArray.sort(ordenfunc);
```

Argumentos

ordenfunc Una función de comparación opcional utilizada para determinar el orden de clasificación. Dados los argumento A y B, la función de ordenamiento especificada debería realizar el ordenamiento como se muestra a continuación:

- -1 si A aparece antes que B en la secuencia de ordenamiento
- 0 si A = B
- 1 si A aparece después que B en la secuencia de ordenamiento

Descripción

Método; ordena la matriz en contexto, sin hacer una copia. Si omite el argumento *ordenfunc*, Flash ordena los elementos en contexto utilizando el operador de comparación <.

Reproductor

Flash 5 o posterior.

Ejemplo

El ejemplo siguiente utiliza `Array.sort` sin especificar el argumento *`ordenfunc`*.

```
var fruits = ["oranges", "apples", "strawberries",
                "pineapples", "cherries"];
 trace(fruits.join())
 fruits.sort()
 trace(fruits.join())
```

Salida:

```
 oranges,apples,strawberries,pineapples,cherries
 apples,cherries,oranges,pineapples,strawberries
```

El ejemplo siguiente utiliza `array.sort` con una función de ordenamiento especificada.

```
var passwords = [
   "gary:foo",
   "mike:bar",
   "john:snafu",
   "steve:yuck",
   "daniel:1234"
 ];
 function order (a, b) {
   // Entries to be sorted are in form
   // name:password
   // Sort using only the name part of the
   // entry as a key.
   var name1 = a.split(':')[0];
   var name2 = b.split(':')[0];
   if (name1 < name2) {
     return -1;
   } else if (name1 > name2) {
     return 1;
   } else {
     return 0;
   }
 }
 for (var i=0; i< password.length; i++) {
   trace (passwords.join());
 }
 passwords.sort(order);
 trace ("Sorted:")
 for (var i=0; i< password.length; i++) {
   trace (passwords.join());
 }
```

Salida:

```
 daniel:1234
 gary:foo
 john:snafu
 mike:bar
 steve:yuck
```

Array.splice

Sintaxis

```
myArray.splice(inicio, Cuenta_elim, valor0,valor1...valorN);
```

Argumentos

inicio El índice del elemento de la matriz donde comienza la inserción y/o el borrado.

Cuenta_elim El número de elementos que se van a borrar. Este número incluye el elemento especificado en el argumento *inicio*. Si no se especifica valor para *Cuenta_elim*, el método borra todos los valores desde el elemento *inicio* hasta el último elemento de la matriz.

valor Cero o más valores que se van a insertar en la matriz en el punto de inserción especificado en el argumento *inicio*. Este argumento es opcional.

Descripción

Método; agrega y/o elimina elementos de una matriz. Este método modifica la propia matriz sin hacer una copia.

Reproductor

Flash 5 o posterior.

Array.toString

Sintaxis

```
myArray.toString();
```

Argumentos

Ninguno.

Descripción

Método; devuelve un valor de cadena que representa los elementos del objeto Array especificado. Todos los elementos de la matriz, comenzando con el índice 0 y finalizando con el índice *myArray*.length-1, se convierte en una cadena concatenada separada por comas.

Reproductor

Flash 5 o posterior.

Ejemplo

El ejemplo siguiente crea myArray y la convierte en una cadena.

```
myArray = new Array();
myArray[0] = 1;
myArray[1] = 2;
myArray[2] = 3;
myArray[3] = 4;
myArray[4] = 5;

trace(myArray.toString())
```

Salida:

```
1,2,3,4,5
```

Array.unshift

Sintaxis
`myArray.unshift(valor1,valor2,...valorN);`

Argumentos
`valor1,...valorN` Uno o más números, elementos o variables que se van a insertar al comienzo de la matriz.

Descripción
Método; agrega uno o más elementos al comienzo de una matriz y devuelve la nueva longitud de la matriz.

Reproductor
Flash 5 o posterior.

Boolean (función)

Sintaxis
`Boolean(expresión);`

Argumentos
`expresión` La variable, número o cadena que se va a convertir en un Booleano.

Descripción
Función; convierte el argumento especificado en un Booleano y devuelve el valor Booleano.

Reproductor
Flash 5 o posterior.

Boolean (objeto)

El objeto Boolean es un objeto envolvente sencillo con la misma funcionalidad que el objeto Boolean de JavaScript estándar. Utilice el objeto Boolean para recuperar el tipo de datos primitivo o la representación de cadena del objeto Boolean.

Resumen de los métodos de un objeto Boolean

Método	Descripción
`toString`	Devuelve la representación de la cadena `(true)` o `(false)` del objeto Boolean.
`valueOf`	Devuelve el tipo de valor primitivo del objeto Boolean especificado.

Constructor del objeto Boolean.

Sintaxis

```
new Boolean();
new Boolean(x);
```

Argumentos

x Un número, cadena, Booleano, objeto, clip de película u otra expresión. Este argumento es opcional.

Descripción

Constructor; crea una instancia del objeto Boolean. Si omite el argumento *x*, el objeto Boolean se inicializa con un valor de `false`. Si especifica *x*, el método evalúa el argumento y devuelve el resultado como un valor Booleano según las siguientes reglas de reparto.

- Si *x* es un número, la función devuelve `true` si *x* no es igual a 0, o `false` si *x* es cualquier otro número.
- Si *x* es un Booleano, la función devuelve *x*.
- Si *x* es un objeto o clip de película, la función devuelve `true` si *x* no es igual a `null`; en caso contrario, la función devuelve `false`.
- Si *x* es una cadena, la función devuelve `true` si `Number(x)` no es igual a 0; en caso contrario, la función devuelve `false`.

Nota: Para mantener la compatibilidad con Flash 4, el manejo de cadenas por el objeto Boolean no es estándar de ECMA-262.

Reproductor

Flash 5 o posterior.

Boolean.toString

Sintaxis

```
Boolean.toString();
```

Argumentos

Ninguno.

Descripción

Método; devuelve la representación de la cadena `(true)` o `(false)` del objeto Boolean.

Reproductor

Flash 5 o posterior.

Boolean.valueOf

Sintaxis
`Boolean.valueOf();`

Argumentos
Ninguno.

Descripción
Método; devuelve el tipo de valor primitivo del objeto Boolean especificado y convierte el objeto envolvente Booleano a su tipo de valor primitivo.

Reproductor
Flash 5 o posterior.

break

Sintaxis
`break;`

Argumentos
Ninguno.

Descripción
Acción; aparece dentro de un bucle (`for`, `for...in`, `do...while` o `while`). La acción `break` da instrucciones a Flash para que se salte el resto del cuerpo del bucle, detenga la acción de bucle y ejecute la sentencia que sigue a la sentencia del bucle. Utilice la acción `break` para romper una serie de bucles anidados.

Reproductor
Flash 4 o posterior.

Ejemplo
El ejemplo siguiente utiliza la acción `break` para salir de un bucle que si no es infinito.

```
i = 0;
while (true) {
  if (i >= 100) {
    break;
  }
  i++;
}
```

call

Sintaxis
```
call(frame);
```

Argumentos
frame El nombre o el número del fotograma que se va a llamar en el contexto del script.

Descripción
Acción; intercambia el contexto del script actual al script anexado al fotograma que se llama. Las variables locales no existirán una vez que el script acaba de ejecutarse.

Reproductor
Flash 4 o posterior. Esta acción se ha desestimado en Flash 5 y se le recomienda que utilice la acción `function`.

Véase también
"function" a pagina 272

chr

Sintaxis
```
chr(número);
```

Argumentos
número El número de código ASCII que se convierte en carácter.

Descripción
Función de cadena; convierte los números de código ASCII en caracteres.

Reproductor
Flash 4 o posterior. Esta función se ha desestimado en Flash 5; se recomienda la utilización del método `String.fromCharCode`.

Ejemplo
El ejemplo siguiente convierte el número en la letra "A".

```
chr(65) = "A"
```

Véase también
"String.fromCharCode" a pagina 374

Color (objeto)

El objeto Color le permite establecer y recuperar el valor de color RGB y la transformación de color de un clip de película. El objeto Color se admite en Flash 5 y en versiones posteriores de Flash Player.

Debe utilizar el constructor `new Color()` para crear una instancia del objeto Color antes de llamar a los métodos del objeto Color.

Resumen de los métodos del objeto Color

Método	Descripción
`getRGB`	Devuelve el valor numérico RGB establecido por la última llamada `setRGB`.
`getTransform`	Devuelve la información de transformación establecida por la última llamada `setTransform`.
`setRGB`	Establece la representación hexadecimal del valor RGB para un objeto Color.
`setTransform`	Establece la transformación de color para un objeto Color.

Constructor del objeto Color.

Sintaxis

```
new Color(destino);
```

Argumentos

`destino` El nombre del clip de película al que se le aplica el nuevo color.

Descripción

Constructor; crea un objeto Color para el clip de película especificado por el argumento *`destino`*.

Reproductor

Flash 5 o posterior.

Ejemplo

El ejemplo siguiente crea un nuevo objeto Color llamado `myColor` para la película `myMovie`.

```
myColor = new Color(myMovie);
```

Color.getRGB

Sintaxis
myColor.getRGB();

Argumentos
Ninguno.

Descripción
Método; devuelve los valores numéricos establecidos por la última llamada `setRGB`.

Reproductor
Flash 5 o posterior.

Ejemplo
El código siguiente recupera el valor RGB como cadena hexadecimal.

```
value = (getRGB()).toString(16);
```

Véase también
"Color.setRGB" a pagina 240

Color.getTransform

Sintaxis
myColor.getTransform();

Argumentos
Ninguno.

Descripción
Método; devuelve el valor de transformación establecido por la última llamada `setTransform`.

Reproductor
Flash 5 o posterior.

Véase también
"Color.setTransform" a pagina 240

Color.setRGB

Sintaxis
myColor.setRGB(*0xRRGGBB*);

Argumentos
0xRRGGBB El hexadecimal o color RGB que se va a establecer. *RR*, *GG* y *BB* cada uno consiste en dos dígitos hexadecimales que especifican el desplazamiento de cada componente de color.

Descripción
Método; especifica un color RGB para el objeto Color. Llamar a este método cancela cualquier configuración anterior del método `setTransform`.

Reproductor
Flash 5 o posterior.

Ejemplo
El ejemplo siguiente establece el valor de color RGB para el clip de película `myMovie`.

```
myColor = new Color(myMovie);
myColor.setRGB(0x993366);
```

Véase también
"Color.setTransform" a pagina 240

Color.setTransform

Sintaxis
myColor.setTransform(*colorTransformObject*);

Argumentos
colorTransformObject Un objeto creado utilizando el constructor del objeto Object genérico, especificando los valores de transformación del color para los parámetros. El objeto transformación de color debe tener los parámetros `ra`, `rb`, `ga`, `gb`, `ba`, `bb`, `aa`, `ab`, que se explican más adelante.

Descripción
Método; establece la información de transformación de color para un objeto Color. El argumento *colorTransformObject* es un objeto que cree utilizando el objeto Object genérico con parámetros que especifican los valores de porcentaje y de desplazamiento para los componentes rojo, verde, azul y alfa (transparencia) de un color, introducidos en un formato *0xRRGGBBAA*.

Los parámetros para un objeto de transformación de color se definen como se muestra a continuación:

- *ra* es el porcentaje del componente rojo (de -100 a 100).
- *rb* es el desplazamiento del componente rojo (de -255 a 255).
- *ga* es el porcentaje del componente verde (de -100 a 100).
- *gb* es el desplazamiento del componente verde (de -255 a 255).
- *ba* es el porcentaje del componente azul (de -100 a 100).
- *bb* es el desplazamiento del componente azul (de -255 a 255).
- *aa* es el porcentaje de alfa (de -100 a 100).
- *ab* es el desplazamiento de alfa (de -255 a 255).

Cree un objeto de transformación de color como se muestra a continuación:

```
myColorTransform = new Object();
myColorTransform.ra = 50;
myColorTransform.rb = 244;
myColorTransform.ga = 40;
myColorTransform.gb = 112;
myColorTransform.ba = 12;
myColorTransform.bb = 90;
myColorTransform.aa = 40;
myColorTransform.ab = 70;
```

También podría utilizar la sintaxis siguiente:

```
myColorTransform = { ra: '50', rb: '244', ga: '40', gb: '112',
ba: '12', bb: '90', aa: '40', ab: '70'}
```

Reproductor
Flash 5 o posterior.

Ejemplo
El ejemplo siguiente muestra el proceso de creación de un nuevo objeto Color para una película de destino, la creación de un objeto de transformación de color con los parámetros definidos anteriormente y pasar el objeto de transformación de color a un objeto Color utilizando el método `setTransform`.

```
//Create a color object called myColor for the target myMovie

myColor = new Color(myMovie);

//Create a color transform object called myColorTransfrom using
the generic object Object

myColorTransform = new Object;

// Set the values for myColorTransform

myColorTransform = { ra: '50', rb: '244', ga: '40', gb: '112',
ba: '12', bb: '90', aa: '40', ab: '70'}

//Associate the color transform object with the Color object
created for myMovie

myColor.setTransform(myColorTransform);
```

continue

Sintaxis
`continue;`

Argumentos
Ninguno.

Descripción
Acción; aparece dentro de varios tipos de sentencias de bucle.

En un bucle `while` `continue` hace que Flash se salte el resto del cuerpo del bucle y salte a la parte superior del bucle, donde se comprueba la condición.

En un bucle `do...while` `continue` hace que Flash se salte el resto del cuerpo del bucle y salte a la parte inferior del bucle, donde se comprueba la condición.

En un bucle `for` `continue` hace que Flash se salte el resto del cuerpo del bucle y vaya a la evaluación de la postexpresión `for` del bucle.

En un bucle `for...in` `continue` hace que Flash se salte el resto del cuerpo del bucle y retroceda a la parte superior del bucle, donde se procesa el siguiente valor de la enumeración.

Reproductor
Flash 4 o posterior.

Véase también
"do...while" a pagina 262
"for" a pagina 268
"for...in" a pagina 270
"while" a pagina 387

_currentframe

Sintaxis
`nombre_instancia``._currentframe`

Argumentos
`nombre_instancia` El nombre de una instancia de clip de película.

Descripción
Propiedad (de sólo lectura); devuelve el número del fotograma donde se encuentra actualmente la cabeza lectora en la Línea de tiempo.

Reproductor
Flash 4 o posterior.

Ejemplo
El ejemplo siguiente utiliza `_currentframe` para dirigir a una película para que avance cinco fotogramas a partir del fotograma que contiene la acción.

```
gotoAndStop(_currentframe + 5);
```

Date (objeto)

El objeto Date le permite recuperar valores de fecha y hora relativos al horario universal (Hora de Greenwich, que ahora se llama Hora universal coordinada) o relativos al sistema operativo en el que se está ejecutando Flash Player. Para llamar a los métodos del objeto Date, primero debe crear una instancia del objeto Date utilizando el constructor.

El objeto Date requiere Flash Player 5.

Los métodos del objeto Date no son estáticos, sino que se aplican a la instancia individual del objeto Date especificado cuando se llama al método.

Resumen de métodos del objeto Date

Método	Descripción
`getDate`	Devuelve el día del mes del objeto Date especificado según la hora local.
`getDay`	Devuelve el día del mes del objeto Date especificado según la hora local.
`getFullYear`	Devuelve el año en cuatro dígitos del objeto Date especificado según la hora local.
`getHours`	Devuelve la hora del objeto Date especificado según la hora local.
`getMilliseconds`	Devuelve los milisegundos del objeto Date especificado según la hora local.
`getMinutes`	Devuelve los minutos del objeto Date especificado según la hora local.
`getMonth`	Devuelve el mes del objeto Date especificado según la hora local.
`getSeconds`	Devuelve los segundos del objeto Date especificado según la hora local.
`getTime`	Devuelve el número de milisegundos desde la media noche del 1 de enero de 1970, hora universal, para el objeto Date especificado.
`getTimezoneOffset`	Devuelve la diferencia, en minutos, entre la hora local del sistema y la hora universal.
`getUTCDate`	Devuelve el día (fecha) del mes del objeto Date especificado según la hora universal.
`getUTCDay`	Devuelve el día de la semana del objeto Date especificado según la hora universal.

Método	Descripción
`getUTCFullYear`	Devuelve el año en cuatro dígitos del objeto Date especificado según la hora universal.
`getUTCHours`	Devuelve la hora del objeto Date especificado según la hora universal.
`getUTCMilliseconds`	Devuelve los milisegundos del objeto Date especificado según la hora universal.
`getUTCMinutes`	Devuelve el minuto del objeto Date especificado según la hora universal.
`getUTCMonth`	Devuelve el mes del objeto Date especificado según la hora universal.
`getUTCSeconds`	Devuelve los segundos del objeto Date especificado según la hora universal.
`getYear`	Devuelve el año del objeto Date especificado según la hora local.
`setDate`	Devuelve el día del mes de un objeto Date especificado según la hora local.
`setFullYear`	Establece el año completo para un objeto Date según la hora local.
`setHours`	Establece las horas para un objeto Date según la hora local.
`setMilliseconds`	Establece los milisegundos para un objeto Date según la hora local.
`setMinutes`	Establece los minutos para un objeto Date según la hora local.
`setMonth`	Establece el mes para un objeto Date según la hora local.
`setSeconds`	Establece los segundos para un objeto Date según la hora local.
`setTime`	Establece la fecha para el objeto Date especificado en milisegundos.
`setUTCDate`	Establece la fecha del objeto Date especificado según la hora universal.
`setUTCFullYear`	Establece el año del objeto Date especificado según la hora universal.
`setUTCHours`	Establece la hora del objeto Date especificado según la hora universal.
`setUTCMilliseconds`	Establece los milisegundos del objeto Date especificado según la hora universal.

Método	Descripción
`setUTCMinutes`	Establece el minuto del objeto Date especificado según la hora universal.
`setUTCMonth`	Establece el mes representado por objeto Date especificado según la hora universal.
`setUTCSeconds`	Establece los segundos del objeto Date especificado según la hora universal.
`setYear`	Establece el año del objeto Date especificado según la hora local.
`toString`	Devuelve un valor de cadena que representa la fecha y la hora almacenada en el objeto Date especificado.
`Date.UTC`	Devuelve el número de milisegundos desde la media noche del 1 de enero de 1970, hora universal, y la hora especificada.

Constructor del objeto Date.

Sintaxis

```
new Date();
new Date(año [, mes [, fecha [, hora [, minuto [, segundo [, milisegundo ]]]]]] );
```

Argumentos

año Un valor de 0 a 99 indica de 1900 a 1999, en caso contrario deben especificarse los 4 dígitos del año.

mes Un número entero desde 0 (enero) hasta 11 (diciembre). Este argumento es opcional.

fecha Un número entero de 1 a 31. Este argumento es opcional.

hora Un número entero desde 0 (media noche) hasta 23 (11 p.m.).

minuto Un número entero de 0 a 59. Este argumento es opcional.

segundo Un número entero de 0 a 59. Este argumento es opcional.

milisegundo Un número entero de 0 a 999. Este argumento es opcional.

Descripción

Objeto; construye un nuevo objeto Date manteniendo la fecha y la hora actuales.

Reproductor
Flash 5 o posterior.

Ejemplo
El ejemplo siguiente recupera la fecha y la hora actuales.

```
now = new Date();
```

El ejemplo siguiente crea un nuevo objeto Date para el cumpleaños de Gary, el 7 de agosto de 1974.

```
gary_birthday = new Date (74, 7, 7);
```

El ejemplo siguiente crea un nuevo objeto Date, concatena los valores devueltos de los métodos del objeto Date `getMonth`, `getDate` y `getFullYear`, y los muestra en el campo de texto especificado por la variable `dateTextField`.

```
myDate = new Date();
dateTextField = (mydate.getMonth() + "/" + myDate.getDate() + "/"
+ mydate.getFullYear());
```

Date.getDate

Sintaxis
myDate`.getDate();`

Argumentos
Ninguno.

Descripción
Método; devuelve el día del mes (un número entero de 1 a 31) del objeto Date especificado según la hora local.

Reproductor
Flash 5 o posterior.

Date.getDay

Sintaxis
myDate`.getDay();`

Argumentos
Ninguno.

Descripción
Método; devuelve el día del mes (0 para el domingo, 1 para el lunes, etc.) del objeto Date especificado según la hora local. La hora local la determina el sistema operativo en el que se esté ejecutando Flash Player.

Reproductor
Flash 5 o posterior.

Date.getFullYear

Sintaxis
myDate.getFullYear();

Argumentos
Ninguno.

Descripción
Método; devuelve el año completo (un número de cuatro dígitos, por ejemplo, 2000) del objeto Date especificado según la hora local. La hora local la determina el sistema operativo en el que se esté ejecutando Flash Player.

Reproductor
Flash 5 o posterior.

Ejemplo
El ejemplo siguiente utiliza el constructor para crear un nuevo objeto Date y envía el valor devuelto por el método `getFullYear` a la ventana de Salida.

```
myDate = new Date();
trace(myDate.getFullYear());
```

Date.getHours

Sintaxis
myDate.getHours();

Argumentos
Ninguno.

Descripción
Método; devuelve la hora (un número entero de 0 a 23) del objeto Date especificado según la hora local. La hora local la determina el sistema operativo en el que se esté ejecutando Flash Player.

Reproductor
Flash 5 o posterior.

Date.getMilliseconds

Sintaxis
```
myDate.getMilliseconds();
```

Argumentos
Ninguno.

Descripción
Método; devuelve los milisegundos (un número entero de 0 a 999) del objeto Date especificado según la hora local. La hora local la determina el sistema operativo en el que se esté ejecutando Flash Player.

Reproductor
Flash 5 o posterior.

Date.getMinutes

Sintaxis
```
myDate.getMinutes();
```

Argumentos
Ninguno.

Descripción
Método; devuelve los minutos (un número entero de 0 a 59) del objeto Date especificado según la hora local. La hora local la determina el sistema operativo en el que se esté ejecutando Flash Player.

Reproductor
Flash 5 o posterior.

Date.getMonth

Sintaxis
```
myDate.getMonth();
```

Argumentos
Ninguno.

Descripción
Método; devuelve el mes (0 para enero, 1 para febrero, etc.) del objeto Date especificado, según la hora local. La hora local la determina el sistema operativo en el que se esté ejecutando Flash Player.

Reproductor
Flash 5 o posterior.

Date.getSeconds

Sintaxis
```
myDate.getSeconds();
```

Argumentos
Ninguno.

Descripción
Método; devuelve los segundos (un número entero de 0 a 59) del objeto Date especificado, según la hora local. La hora local la determina el sistema operativo en el que se esté ejecutando Flash Player.

Reproductor
Flash 5 o posterior.

Date.getTime

Sintaxis
```
myDate.getTime();
```

Argumentos
Ninguno.

Descripción
Método; devuelve el número de milisegundos (un número entero de 0 a 999) desde la media noche del 1 de enero de 1970, hora universal, para el objeto Date especificado. Utilice este método para representar un instante específico en la hora cuando se comparan dos o más horas definidas en diferentes zonas horarias.

Reproductor
Flash 5 o posterior.

Date.getTimezoneOffset

Sintaxis
```
mydate.getTimezoneOffset();
```

Argumentos
Ninguno.

Descripción
Método; devuelve la diferencia, en minutos, entre la hora local del sistema y la hora universal.

Reproductor
Flash 5 o posterior.

Ejemplo
El ejemplo siguiente devuelve la diferencia entre la hora local de ahorro de energía de San Francisco y la hora local. Se introduce como factor la hora de ahorro de energía en el resultado devuelto solamente si la fecha definida en el objeto Date es durante el horario de ahorro de energía.

```
new Date().getTimezoneOffset();
```

El resultado es el que se muestra a continuación:

```
420 (7 hours * 60 minutes/hour = 420 minutes)
```

Date.getUTCDate

Sintaxis
myDate.getUTCDate();

Argumentos
Ninguno.

Descripción
Método; devuelve el día (fecha) del mes del objeto Date especificado según la hora universal.

Reproductor
Flash 5 o posterior.

Date.getUTCDay

Sintaxis
myDate.getUTCDate();

Argumentos
Ninguno.

Descripción
Método; devuelve el día de la semana del objeto Date especificado según la hora universal.

Date.getUTCFullYear

Sintaxis
```
myDate.getUTCFullYear();
```

Argumentos
Ninguno.

Descripción
Método; devuelve el año en cuatro dígitos del objeto Date especificado según la hora universal.

Reproductor
Flash 5 o posterior.

Date.getUTCHours

Sintaxis
```
myDate.getUTCHours();
```

Argumentos
Ninguno.

Descripción
Método; devuelve las horas del objeto Date especificado según la hora universal.

Reproductor
Flash 5 o posterior.

Date.getUTCMilliseconds

Sintaxis
```
myDate.getUTCMilliseconds();
```

Argumentos
Ninguno.

Descripción
Método; devuelve los milisegundos del objeto Date especificado según la hora universal.

Reproductor
Flash 5 o posterior.

Date.getUTCMinutes

Sintaxis
```
myDate.getUTCMinutes();
```

Argumentos
Ninguno.

Descripción
Método; devuelve los minutos del objeto Date especificado según la hora universal.

Reproductor
Flash 5 o posterior.

Date.getUTCMonth

Sintaxis
```
myDate.getUTCMonth();
```

Argumentos
Ninguno.

Descripción
Método; devuelve el mes del objeto Date especificado según la hora universal.

Reproductor
Flash 5 o posterior.

Date.getUTCSeconds

Sintaxis
```
myDate.getUTCSeconds();
```

Argumentos
Ninguno.

Descripción
Método; devuelve los segundos del objeto Date especificado según la hora universal.

Reproductor
Flash 5 o posterior.

Date.getYear

Sintaxis
```
myDate.getYear();
```

Argumentos
Ninguno.

Descripción
Método; devuelve el año del objeto Date especificado según la hora local. La hora local la determina el sistema operativo en el que se esté ejecutando Flash Player. El año es el año completo menos 1900. Por ejemplo, el año 2000 se representa como 100.

Reproductor
Flash 5 o posterior.

Date.setDate

Sintaxis
```
myDate.setDate(fecha);
```

Argumentos
fecha Un número entero de 1 a 31.

Descripción
Método; devuelve el día del mes del objeto Date especificado según la hora local. La hora local la determina el sistema operativo en el que se esté ejecutando Flash Player.

Reproductor
Flash 5 o posterior.

Date.setFullYear

Sintaxis
```
myDate.setFullYear(año [, mes [, fecha]] );
```

Argumentos
año Un número de cuatro dígitos que especifica un año. Los números de dos dígitos no representan años; por ejemplo, 99 no es el año 1999, sino el año 99.

mes Un número entero desde 0 (enero) hasta 11 (diciembre). Este argumento es opcional.

fecha Un número de 1 a 31. Este argumento es opcional.

Descripción
Método; establece el año del objeto Date especificado según la hora local. Si se especifican los argumentos *mes* y *fecha*, también se establecen en la hora local. La hora local la determina el sistema operativo en el que se esté ejecutando Flash Player.

Los resultados de `getUTCDay` y `getDay` pueden cambiar como resultado de la llamada de este método.

Reproductor
Flash 5 o posterior.

Date.setHours

Sintaxis
```
myDate.setHours(hora);
```

Argumentos
hora Un número entero desde 0 (media noche) hasta 23 (11 p.m.).

Descripción
Método; establece las horas del objeto Date especificado según la hora local. La hora local la determina el sistema operativo en el que se esté ejecutando Flash Player.

Reproductor
Flash 5 o posterior.

Date.setMilliseconds

Sintaxis
```
myDate.setMilliseconds(milisegundo);
```

Argumentos
milisegundo Un número entero de 0 a 999.

Descripción
Método; establece los milisegundos del objeto Date especificado según la hora local. La hora local la determina el sistema operativo en el que se esté ejecutando Flash Player.

Reproductor
Flash 5 o posterior.

Date.setMinutes

Sintaxis
```
myDate.setMinutes(minuto);
```

Argumentos
minuto Un número entero de 0 a 59.

Descripción
Método; establece los minutos del objeto Date especificado según la hora local. La hora local la determina el sistema operativo en el que se esté ejecutando Flash Player.

Reproductor
Flash 5 o posterior.

Date.setMonth

Sintaxis
```
myDate.setMonth(mes [, fecha ]);
```

Argumentos
mes Un número entero desde 0 (enero) hasta 11 (diciembre).

fecha Un número entero de 1 a 31. Este argumento es opcional.

Descripción
Método; establece el mes del objeto Date especificado en la hora local. La hora local la determina el sistema operativo en el que se esté ejecutando Flash Player.

Reproductor
Flash 5 o posterior.

Date.setSeconds

Sintaxis
```
myDate.setSeconds(segundo);
```

Argumentos
segundo Un número entero de 0 a 59.

Descripción
Método; establece los segundos del objeto Date especificado en la hora local. La hora local la determina el sistema operativo en el que se esté ejecutando Flash Player.

Reproductor
Flash 5 o posterior.

Date.setTime

Sintaxis
myDate.setTime(*milisegundo*);

Argumentos
milisegundo Un número entero de 0 a 999.

Descripción
Método; establece la fecha para el objeto Date especificado en milisegundos.

Reproductor
Flash 5 o posterior.

Date.setUTCDate

Sintaxis
myDate.setUTCDate(*fecha*);

Argumentos
fecha Un número entero de 1 a 31.

Descripción
Método; establece la fecha del objeto Date especificado en la hora universal. Llamar a este método no modifica los otros campos del objeto Date especificado, pero los métodos `getUTCDay` y `getDay` puede dar un nuevo valor si el día de la semana cambia como resultado de llamar a este método.

Reproductor
Flash 5 o posterior.

Date.setUTCFullYear

Sintaxis
myDate.setUTCFullYear(*año* [, *mes* [, *fecha*]])

Argumentos
año El año especificado como año de cuatro dígitos, por ejemplo, 2000.

mes Un número entero desde 0 (enero) hasta 11 (diciembre). Este argumento es opcional.

fecha Un número entero de 1 a 31. Este argumento es opcional.

Descripción
Método; establece el año del objeto Date especificado (*mydate*) en la hora universal.

De modo opcional, este método también puede establecer el mes y la fecha representados por el objeto Date especificado. No se modifican otros campos del objeto Date. Llamar a `setUTCFullyear` puede hacer que `getUTCDay` y `getDay` den como resultado un nuevo valor si el día de la semana cambia como resultado de esta operación.

Reproductor
Flash 5 o posterior.

Date.setUTCHours

Sintaxis
```
myDate.setUTCHours(hora [, minuto [, segundo [, milisegundo]]]));
```

Argumentos
hora Un número entero desde 0 (media noche) hasta 23 (11 p.m.).

minuto Un número entero de 0 a 59. Este argumento es opcional.

segundo Un número entero de 0 a 59. Este argumento es opcional.

milisegundo Un número entero de 0 a 999. Este argumento es opcional.

Descripción
Método; establece la hora del objeto Date especificado en la hora universal.

Reproductor
Flash 5 o posterior.

Date.setUTCMilliseconds

Sintaxis
```
myDate.setUTCMilliseconds(milisegundo);
```

Argumentos
milisegundo Un número entero de 0 a 999.

Descripción
Método; establece los milisegundos del objeto Date especificado en la hora universal.

Reproductor
Flash 5 o posterior.

Date.setUTCMinutes

Sintaxis
```
myDate.setUTCMinutes(minuto [, segundo [, milisegundo]]));
```

Argumentos
minuto Un número entero de 0 a 59.

segundo Un número entero de 0 a 59. Este argumento es opcional.

milisegundo Un número entero de 0 a 999. Este argumento es opcional.

Descripción
Método; establece el minuto del objeto Date especificado en la hora universal.

Reproductor
Flash 5 o posterior.

Date.setUTCMonth

Sintaxis
```
myDate.setUTCMonth(mes [, fecha]);
```

Argumentos
mes Un número entero desde 0 (enero) hasta 11 (diciembre).

fecha Un número entero de 1 a 31. Este argumento es opcional.

Descripción
Método; establece el mes y opcionalmente el día (fecha) del objeto Date especificado en la hora universal. Llamar a este método no modifica los otros campos del objeto Date especificado, pero los métodos `getUTCDay` y `getDay` pueden dar un nuevo valor si el día de la semana cambia como resultado de llamar al argumento *fecha* cuando se llama a `setUTCMonth`.

Reproductor
Flash 5 o posterior.

Date.setUTCSeconds

Sintaxis
```
myDate.setUTCSeconds(segundo [, milisegundo]));
```

Argumentos
segundo Un número entero de 0 a 59.

milisegundo Un número entero de 0 a 999. Este argumento es opcional.

Descripción
Método; establece los segundos del objeto Date especificado en la hora universal.

Reproductor
Flash 5 o posterior.

Date.setYear

Sintaxis
myDate.setYear(*año*);

Argumentos
año Un número de cuatro dígitos, por ejemplo, 2000.

Descripción
Método; establece el año del objeto Date especificado en la hora local. La hora local la determina el sistema operativo en el que se esté ejecutando Flash Player.

Reproductor
Flash 5 o posterior.

Date.toString

Sintaxis
myDate.toString();

Argumentos
Ninguno.

Descripción
Método; devuelve una valor de cadena para el objeto Date especificado en un formato que se puede leer.

Reproductor
Flash 5 o posterior.

Ejemplo
El ejemplo siguiente devuelve la información del objeto de fecha dateOfBirth como una cadena.

```
var dateOfBirth = newDate(74, 7, 7, 18, 15);
trace (dateOfBirth.toString());
```

Salida (para la hora estándar del Pacífico):

```
Wed Aug 7 18:15:00 GMT-0700 1974
```

Date.UTC

Sintaxis

```
Date.UTC(año, mes [, fecha [, hora [, minuto [, segundo [,
milisegundo ]]]]]);
```

Argumentos

año Un número de cuatro dígitos, por ejemplo, 2000.

mes Un número entero desde 0 (enero) hasta 11 (diciembre).

fecha Un número entero de 1 a 31. Este argumento es opcional.

hora Un número entero desde 0 (media noche) hasta 23 (11 p.m.).

minuto Un número entero de 0 a 59. Este argumento es opcional.

segundo Un número entero de 0 a 59. Este argumento es opcional.

milisegundo Un número entero de 0 a 999. Este argumento es opcional.

Descripción

Método; devuelve el número de milisegundos desde la media noche del 1 de enero de 1970, hora universal, y la hora especificada en los argumentos. Este es un método estático que se invoca por medio del constructor del objeto Date, no por medio de un objeto Date específico. Este método le permite crear un objeto Date que toma la hora universal, mientras que el constructor Date toma la hora local.

Reproductor

Flash 5 o posterior.

Ejemplo

El ejemplo siguiente crea un nuevo objeto Date `gary_birthday` definido en hora universal. Esta es la variación de hora universal del ejemplo utilizado para el método constructor `new Date()`.

```
gary_birthday = new Date(Date.UTC(1974, 7, 8));
```

delete

Sintaxis

```
delete (referencia);
```

Argumentos

referencia El nombre de la variable u objeto que se va a eliminar.

Descripción

Operador; destruye el objeto o variable especificada como *referencia* y devuelve `true` si el objeto se ha eliminado con éxito; en caso contrario devuelve `false`. Este operador es útil para liberar la memoria que utilizan los scripts. Aunque, `delete` es un operador se usa corrientemente como una sentencia:

```
delete x;
```

El operador delete puede fallar y devolver false si la *referencia* no existe o no puede eliminarse. Los objetos y propiedades predefinidas y las variables declaradas con `var`, no pueden eliminarse.

Reproductor

Flash 5 o posterior.

Ejemplo

El ejemplo siguiente crea un objeto, lo utiliza y después lo elimina cuando ya no es necesario.

```
account = new Object();
  account.name = 'Jon';
  account.balance = 10000;
  ...
  delete account;
```

El ejemplo siguiente elimina una propiedad de un objeto.

```
// create the new object "account"
account = new Object();
// assign property name to the account
  account.name = 'Jon';
// delete the property
delete account.name;
```

El ejemplo siguiente es otro ejemplo de la eliminación de una propiedad de un objeto.

```
// create an Array object with length 0
array = new Array();
// Array.length is now 1
   array[0] = "abc";
// add another element to the array,Array.length is now 2
   array[1] = "def";
// add another element to array,Array.length is now 3
   array[2] = "ghi";
// array[2] is deleted, but Array.length is not changed,
   delete array[2];
```

El ejemplo siguiente muestra el comportamiento de `delete` sobre referencias de objeto.

```
// create a new object, and assign the variable ref1 to refer
to the object
ref1 = new Object();
ref1.name = "Jody";
// copy the reference variable into a new variable, and delete
ref1
ref2 = ref1;
delete ref1;
```

Si `ref1` no se ha copiado en `ref2`, el objeto se habría eliminado cuando se eliminó `ref1`, debido a que no habría referencias a él. Si se fuera a eliminar `ref2`, ya no habría ninguna referencia al objeto y se destruiría y se haría disponible la memoria que estaba utilizando.

Véase también

"var" a pagina 386

do...while

Sintaxis

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

Argumentos

condición Las condiciones que se van a evaluar.

sentencia La sentencia que se va a ejecutar mientras que la *condición* evalúe como `true`.

Descripción

Acción; ejecuta las sentencias y después evalúa la condición de un bucle, mientras que la condición sea true.

Reproductor

Flash 4 o posterior.

Véase también

"break" a pagina 236
"continue" a pagina 242

_droptarget

Sintaxis

draggableInstanceName._droptarget

Argumentos

draggableInstanceName El nombre de una instancia de clip de película que era el destino de una acción startDrag.

Descripción

Propiedad (de sólo lectura); devuelve la ruta absoluta en notación de sintaxis de barras de la instancia de clip de película en el que *draggableInstanceName* se soltó. La propiedad _droptarget siempre devuelve una ruta que comienza con /. Para comparar la propiedad _droptarget de una instancia con una referencia, utilice eval para convertir el valor devuelto de sintaxis de barras en una referencia.

Reproductor

Flash 4 o posterior.

Ejemplo

El ejemplo siguiente evalúa la propiedad _droptarget de la instancia de clip de película garbage y utiliza eval para convertirlo de sintaxis de barras a referencia de sintaxis de punto. Entonces la referencia garbage se compara con la referencia a la instancia del clip de película trash. Si las dos referencias son equivalentes, la visibilidad de garbage está establecida en false. Si no son equivalentes, la instancia garbage se restablece a su posición original.

```
if (eval(garbage._droptarget) == _root.trash) {
   garbage._visible = false;
} else {
   garbage._x = x_pos;
   garbage._y = y_pos;
}
```

Las variables x_pos y y_pos se establecen en el fotograma 1 de la película con el script siguiente:

```
x_pos = garbage._x;
y_pos = garbage._y;
```

Véase también

"startDrag" a pagina 367

duplicateMovieClip

Sintaxis

```
duplicateMovieClip(destino, nuevo_nombre, profundidad);
```

Argumentos

destino La ruta de destino de la película que se va a duplicar.

nuevo_nombre Un identificador único para el clip de película duplicado.

profundidad El nivel de profundidad del clip de película. El nivel de profundidad es el orden de apilamiento que determina como aparecen cuando se superponen los clips de película y otros objetos. Al primer clip de película que cree o la primera instancia que arrastre al Escenario, se le asigna una profundidad de nivel 0. Debe asignar a cada clip de película sucesivo o duplicado un nivel de profundidad diferente para evitar que sustituya a películas en los niveles ocupados o al clip de película original.

Descripción

Acción; crea una instancia de un clip de película mientras se reproduce la película. Los clips de película duplicados siempre comienzan en el fotograma 1, sin tener en cuenta en que fotograma estaba el clip de película original. Las variables del clip de película principal no se copian en el clip de película duplicado. Si se elimina el clip de película principal también se elimina el clip de película duplicado. Utilice la acción o el método `removeMovieClip` para eliminar una instancia de clip de película creada con `duplicateMovieClip`.

Reproductor

Flash 4 o posterior.

Ejemplo

Esta sentencia duplica la instancia de clip de película `flower` diez veces. Se utiliza la variable `i` para crear un nuevo nombre de instancia y una profundidad.

```
on(release) {
   amount = 10;
   while(amount>0) {
        duplicateMovieClip (_root.flower, "mc" + i, i);
        setProperty("mc" + i, _x, random(275));
        setProperty("mc" + i, _y, random(275));
        setProperty("mc" + i, _alpha, random(275));
        setProperty("mc" + i, _xscale, random(50));
        setProperty("mc" + i, _yscale, random(50));
        i = i + 1;
        amount = amount-1;
    }
}
```

Véase también

"removeMovieClip" a pagina 350
"MovieClip.removeMovieClip" a pagina 325

else

Sintaxis
`else {sentencia(s)};`

Argumentos
sentencia(s) Una serie alternativa de sentencias que se ejecutan si la condición especificada en la sentencia `if` es `false`.

Descripción
Acción; especifica las acciones, cláusulas, argumentos y otros condicionales que se ejecutan si la sentencia inicial `if` devuelve `false`.

Reproductor
Flash 4 o posterior.

Véase también
"if" a pagina 279

eq (equal–string specific)

Sintaxis
expresión1 `eq` *expresión2*

Argumentos
expresión1, expresión2 Números, cadenas o variables.

Descripción
Operador de comparación; compara dos expresiones para ver su igualdad y devuelve `true` si *expresión1* es igual a *expresión2*; en caso contrario, devuelve `false`.

Reproductor
Flash 1 o posterior. Este operador se ha desestimado en Flash 5; se recomienda la utilización del nuevo operador == (de igualdad).

Véase también
"== (igualdad)" a pagina 218

escape

Sintaxis
`escape(expresión);`

Argumentos
expresión La expresión que se va a convertir en una cadena y se va a codificar en el formato URL codificado.

Descripción
Función; convierte el argumento en una cadena y utiliza en el formato URL codificado, en el que todos los caracteres alfanuméricos se convierten en secuencias hexadecimales % de escape.

Reproductor
Flash 5 o posterior.

Ejemplo
```
escape("Hello{[World]}");
```
El resultado del código anterior es lo que se muestra a continuación:

```
Hello%7B%5BWorld%5D%7D
```

Véase también
"unescape" a pagina 384

eval

Sintaxis
`eval(expresión);`

Argumentos
expresión Una cadena que contiene el nombre de una variable, propiedad, objeto o clip de película para recuperar.

Descripción
Función; accede a variables, propiedades, objetos o clip de película, por nombre. Si la *expresión* es una variable o una propiedad, el valor de la variable o de la propiedad no se devuelve. Si la *expresión* es un objeto o un clip de película, se devuelve una referencia al objeto o al clip de película. Si el elemento denominado en la *expresión* no puede encontrarse, se devuelve sin definir.

En Flash 4, la función `eval` se utilizaba para simular una matriz, en Flash 5 se le recomienda que utilice el objeto Array para crear matrices.

Nota: La acción `eval` de ActionScript no es la misma que la función `eval` de JavaScript y no puede utilizarse para evaluar sentencias.

Reproductor
Flash 5 o posterior para obtener la funcionalidad completa. Puede utilizar `eval` cuando exporte a Flash Player 4, pero debe utilizar notación de barras y solamente puede acceder a variables, no a propiedades u objetos.

Ejemplo
El ejemplo siguiente utiliza `eval` para determinar el valor de la variable `x` y lo establece en el valor de `y`.

```
x = 3;
y = eval("x");
```

El ejemplo siguiente utiliza eval para hacer referencia al objeto de clip de película asociado con una instancia de clip de película en el Escenario, `Ball`.

```
eval("_root.Ball");
```

Véase también
"Array (objeto)" a pagina 225

evaluate

Sintaxis
sentencia;

Argumentos
Ninguno.

Descripción
Acción; crea una línea nueva vacía e inserta un signo ; para introducir sentencias de script únicas utilizando el campo Expresión en el panel Acciones. La sentencia `evaluate` también permite a los usuarios que están escribiendo scripts en el panel Acciones de Flash 5 en el Modo Normal que llamen a funciones.

Reproductor
Flash 5 o posterior.

_focusrect

Sintaxis

```
_focusrect = Boolean;
```

Argumentos

Boolean `true` or `false`.

Descripción

Propiedad (global); especifica si aparece un rectángulo amarillo alrededor del botón que está resaltado actualmente o no. El valor predeterminado `true` (no cero) muestra un triángulo amarillo alrededor del botón resaltado actualmente o campo de texto cuando el usuario presiona la tecla de tabulación para desplazarse. Especifique `false` para visualizar solamente el estado "sobre" del botón (si se ha definido alguno) según se desplazan los usuarios.

Reproductor

Flash 4 o posterior.

for

Sintaxis

```
for(inic; condición; siguiente); {
sentencia;
}
```

Argumentos

inic Una expresión a evaluar antes de que comience la secuencia de bucle, normalmente una expresión de asignación. Una sentencia `var` también está permitida para este argumento.

condición Una condición que evalúa como `true` o `false`. La condición se evalúa antes de cada repetición del bucle, el bucle sale cuando la condición se evalúa como `false`.

siguiente Una expresión a evaluar después de cada repetición del bucle; normalmente se trata de una expresión de asignación que utiliza los operadores `++` (incremento) o `--` (disminución).

sentencia Una sentencia dentro del cuerpo del bucle que se ejecuta.

Descripción

Acción; una construcción de bucle que evalúa la expresión `init` (inicializar) una vez y después comienza una secuencia de bucle por medio de la cual, mientras que la *`condición`* evalúe como `true`, *`sentencia`* se ejecuta y se evalúa la siguiente expresión.

Algunas propiedades no pueden ser evaluadas por las acciones `for` o `for...in`. Por ejemplo, los métodos incorporados del objeto Array (`Array.sort` y `Array.reverse`) no se incluyen en la enumeración de un objeto Array y las propiedades de clip de película, como `_x` y `_y`, no se enumeran.

Reproductor

Flash 5 o posterior.

Ejemplo

El ejemplo siguiente utiliza `for` para agregar los elementos de una matriz.

```
for(i=0; i<10; i++) {
array [i] = (i + 5)*10;
}
```

Devuelve la matriz siguiente:

```
[50, 60, 70, 80, 90, 100, 110, 120, 130, 140]
```

A continuación se muestra un ejemplo de la utilización de `for` para realizar la misma acción repetidamente. En el código siguiente el bucle `for` agrega los números de 1 a 100.

```
var sum = 0;
  for (var i=1; i<=100; i++) {
    sum = sum + i;
  }
```

Véase también

“++ (incremento)” a pagina 191
“-- (disminución)” a pagina 191
“for...in” a pagina 270
“var” a pagina 386

for...in

Sintaxis

```
for(variableiterant in object){
sentencia; }
```

Argumentos

variableiterant El nombre de una variable que actúa como repetidor, haciendo referencia a cada propiedad de un objeto o elemento de una matriz.

objeto El nombre de un objeto que se va a repetir.

sentencia Una sentencia a ejecutar para cada repetición.

Descripción

Acción; realiza un bucle por la propiedades de un objeto o elementos de una matriz y ejecuta la *sentencia* para cada propiedad de un objeto.

Algunas propiedades no pueden ser enumeradas por las acciones `for` o `for...in`. Por ejemplo, los métodos incorporados del objeto Array (`Array.sort` y `Array.reverse`) no se incluyen en la enumeración de un objeto Array y las propiedades de clip de película, como `_x` y `_y` no se enumeran.

La construcción `for...in` se repite sobre las propiedades de los objetos en la cadena prototipo del objeto repetida. Si el prototipo secundario es principal, repetir las propiedades del secundario con `for...in`, también repetirá las propiedades del principal.

Reproductor

Flash 5 o posterior.

Ejemplo

A continuación se muestra un ejemplo de la utilización de for...in para repetir las propiedades de un objeto.

```
myObject = { name:'Tara', age:27, city:'San Francisco' };
for (name in myObject) {
  trace ("myObject." + name + " = " + myObject[name]);
}
```

La salida de este ejemplo es la que se muestra a continuación:

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

A continuación se muestra un ejemplo de la utilización del operador typeof con for...in para repetir un tipo concreto de secundario.

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

El ejemplo siguiente enumera los elementos secundarios de un clip de película y envía cada uno al fotograma 2 en sus respectivas Líneas de tiempo. El clip de película RadioButtonGroup es principal con varios secundarios, _RedRadioButton_, _GreenRadioButton_ y _BlueRadioButton.

```
for (var name in RadioButtonGroup) {
  RadioButtonGroup[name].gotoAndStop(2);
}
```

_framesloaded

Sintaxis

nombre_instancia._framesloaded

Argumentos

nombre_instancia El nombre de la instancia del clip de película que se va a evaluar.

Descripción

Propiedad (de sólo lectura); el número de fotogramas que se han cargado del flujo de una película. Este propiedad es útil para determinar si el contenido de un fotograma específico y todos los fotogramas anteriores a él se han cargado y están disponibles localmente en un navegador de usuario. Esta propiedad es útil para controlar el proceso de descarga de películas grandes. Por ejemplo, puede que desee mostrar un mensaje a los usuarios indicando que la película se está cargando hasta que se acabe de cargar un fotograma especificado de la película.

Reproductor

Flash 4 o posterior.

Ejemplo

A continuación se muestra un ejemplo de la utilización de la propiedad `_framesloaded` para coordinar el comienzo de la película con el número de fotogramas cargados.

```
if (_framesloaded >= _totalframes) {
gotoAndPlay ("Scene 1", "start");
} else {
setProperty ("_root.loader", _xscale, (_framesloaded/
_totalframes)*100);
}
```

fscommand

Sintaxis

```
fscommand(comando, argumentos);
```

Argumentos

comando Una cadena pasada a la aplicación anfitriona para cualquier utilización.

argumentos Una cadena pasada a la aplicación anfitriona para cualquier utilización.

Descripción

Acción; permite a la película de Flash que se comunique con el programa anfitrión de Flash Player. En un navegador Web, `fscommand` llama a la función `moviename_Dofscommand` de JavaScript en la página HTML que contiene la película de Flash, donde `moviename` es el nombre de Flash Player según se asignó mediante el atributo `NAME` de la etiqueta `EMBED` o la propiedad ID de la etiqueta `OBJECT`. Si Flash Player tiene asignado el nombre `theMovie`, la función de JavaScript llamada es `theMovie_Dofscommand`.

Reproductor

Flash 3 o posterior.

function

Sintaxis

```
function nombre_func ([argumento0, argumento1,...argumentoN]){
sentencia(s)
}

function ([argument0, argumento1,...argumentoN]){
sentencia(s)
}
```

Argumentos

nombre_func El nombre de la nueva función.

argumento Cero o más cadenas, números u objetos para pasar a la `función`.

sentencias Cero o más sentencias de ActionScript que ha definido para el cuerpo de la `función`.

Descripción

Acción; un conjunto de sentencias que define para realizar una determinada tarea. Puede *declarar* o definir una función en una ubicación y llamarla, o invocarla, desde diferentes scripts de una película. Cuando defina una función, también puede especificar argumentos para la función. Los argumentos son marcadores de lugar para los valores sobre los que operarán las funciones. Puede pasar diferentes argumentos a una función, también llamados parámetros, cada vez que la llame.

Utilice la acción `return` en la sentencia de una *`sentencia(s)`* función para hacer que la función devuelva, o genere, un valor.

Sintaxis 1: Declara una `función` con el especificado *`nombre_func`*, *`argumentos`* y *`sentencia(s)`*. Cuando se llama a una función, se invoca la declaración de función. Se permite la referencia hacia delante; dentro de la misma lista de Acciones, puede declararse una función después de que ha sido llamada. Una declaración de función sustituye a cualquier declaración anterior de la misma función. Puede utilizar esta sintaxis siempre que esté permitido un argumento.

Sintaxis 2: Crea una función anónima y la devuelve. Esta sintaxis se utiliza en expresiones y es muy útil para instalar métodos en objetos.

Reproductor

Flash 5 o posterior.

Ejemplo

(Sintaxis 2) El ejemplo siguiente define la función `sqr`, que acepta un argumento y devuelve el `square(x*x)` del argumento. Observe que la función se declara y utiliza en el mismo script, la declaración de función puede aparecer tras la utilización de la función.

```
y=sqr(3);
function sqr(x) {
return x*x;
}
```

(Sintaxis 2) El ejemplo siguiente define un objeto Círculo:

```
function Circle(radius) {
  this.radius = radius;
}
```

La sentencia siguiente define una función anónima que calcula el área de un círculo y la anexa al objeto `Circle` como un método:

```
Circle.prototype.area = function () {return Math.PI * this.radius
* this.radius}
```

ge (mayor o igual que, específico de cadena)

Sintaxis
expresión1 ge *expresión2*

Argumentos
expresión1, expresión2 Números, cadenas o variables.

Descripción
Operador (de comparación); compara *expresión1* con *expresión2* y devuelve `true` si *expresión1* es mayor o igual que *expresión2*; en caso contrario, devuelve `false`.

Reproductor
Flash 4 o posterior. Este operador se ha desestimado en Flash 5; se recomienda la utilización del nuevo operador >=.

Véase también
"`>= (mayor o igual que)`" a pagina 219

getProperty

Sintaxis
```
getProperty(nombre_instancia, propiedad);
```

Argumentos
nombre_instancia El nombre de instancia de un clip de película para el que se ha recuperado la propiedad.

propiedad Una propiedad de un clip de película, como una coordenada *x* o *y*.

Descripción
Función; devuelve el valor de la *propiedad* especificada para la instancia de clip de película.

Reproductor
Flash 4 o posterior.

Ejemplo
El ejemplo siguiente recupera la coordenada del eje horizontal (`_x`) para el clip de película `myMovie`.

```
getProperty(_root.myMovie_item._x);
```

getTimer

Sintaxis
```
getTimer();
```

Argumentos
Ninguno.

Descripción
Función; devuelve el número de milisegundos que han transcurrido desde que la película comenzó a reproducirse.

Reproductor
Flash 4 o posterior.

getURL

Sintaxis
```
getURL(url [, ventana [, variables]]);
```

Argumentos
url La URL de la cual obtener el documento. La URL debe estar en el mismo subdominio que la URL donde reside actualmente la película.

ventana Un argumento opcional que especifica la ventana o el fotograma HTML en el que debería cargarse el documento. Introduzca el nombre de una ventana específica o seleccione uno de los nombres de destino reservados siguientes:

- `_self` especifica el fotograma actual de la ventana activa.
- `_blank` especifica una nueva ventana.
- `_parent` especifica el elemento principal del fotograma actual.
- `_top` especifica el fotograma de nivel superior de la ventana actual.

variables Un argumento opcional que especifica un método para enviar variables. Si no hay variables, omita este argumento; en caso contrario, especifique si se cargan las variables utilizando un método `GET` o `POST`. `GET` anexa las variables al final de la URL y se utiliza para un número pequeño de variables. `POST` envía las variables en un encabezado HTTP aparte y se utiliza para cadenas largas de variables.

Descripción
Acción; carga un documento de una URL específica en una ventana o pasa variables a otra aplicación en una URL definida. Para probar esta acción, asegúrese de que el archivo que se va a cargar se encuentra en la ubicación especificada. Para utilizar una URL absoluta (por ejemplo, *http://www.myserver.com*), necesita una conexión de red.

Reproductor
Flash 2 o posterior. Las opciones GET y POST están disponibles solamente en Flash 4 y versiones posteriores del Reproductor.

Ejemplo
Este ejemplo carga una nueva URL en una ventana del navegador vacía. La acción `getURL` destina la variable `incomingAd` como el parámetro *`url`* para que pueda cambiar la URL cargada sin tener que editar la película de Flash. El valor de la variable `incomingAd` se pasa antes a Flash en la película utilizando una acción `loadVariables`.

```
on(release) {
    getURL(incomingAd, "_blank");
}
```

Véase también
“loadVariables” a pagina 296
“XML.send” a pagina 407
“XML.sendAndLoad” a pagina 407
“XMLSocket.send” a pagina 417

getVersion

Sintaxis
```
getVersion();
```

Argumentos
Ninguno.

Descripción
Función, devuelve una cadena que contiene la información de la versión de Flash Player y de la plataforma.

Esta función no funciona en el modo probar película y solamente devolverá información para la versión 5 y posteriores de Flash Player.

Ejemplo
A continuación se muestra un ejemplo de una cadena devuelta por la función `getVersion`:

```
WIN 5,0,17,0
```

Ésta indica que la plataforma es Windows y el número de la versión de Flash Player es versión mayor 5, versión menor 17 (5.0r1.7).

Reproductor
Flash 5 o posterior.

gotoAndPlay

Sintaxis

```
gotoAndPlay(escena, fotograma);
```

Argumentos

escena El nombre de escena a la que se envía la cabeza lectora.

fotograma El número de fotograma al que se envía la cabeza lectora.

Descripción

Acción; envía la cabeza lectora al fotograma especificado en una escena y reproduce a partir de ese fotograma. Si no se especifica escena, la cabeza lectora va el fotograma especificado en la escena actual.

Reproductor

Flash 2 o posterior.

Ejemplo

Cuando el usuario hace clic sobre un botón la acción `gotoAndPlay` se asigna, se envía la cabeza lectora al fotograma 16 y comienza a reproducirse.

```
on(release) {
    gotoAndPlay(16);
}
```

gotoAndStop

Sintaxis

```
gotoAndStop(escena, fotograma);
```

Argumentos

escena El nombre de escena a la que se envía la cabeza lectora.

fotograma El número de fotograma al que se envía la cabeza lectora.

Descripción

Acción; envía la cabeza lectora al fotograma especificado en una escena y lo detiene. Si no se especifica ninguna escena, la cabeza lectora se envía al fotograma en la escena actual.

Reproductor

Flash 2 o posterior.

Ejemplo

Cuando el usuario hace clic sobre un botón la acción `gotoAndStop` se asigna, se envía la cabeza lectora al fotograma 5 y deja de reproducirse.

```
on(release) {
    gotoAndStop(5);
}
```

gt (greater than –string specific)

Sintaxis
expresión1 gt *expresión2*

Argumentos
expresión1, expresión2 Números, cadenas o variables.

Descripción
Operador (de comparación); compara *expresión1* con *expresión2* y devuelve `true` si *expresión1* es mayor o igual que *expresión2*; en caso contrario, devuelve `false`.

Reproductor
Flash 4 o posterior. Este operador se ha desestimado en Flash 5; se recomienda la utilización del nuevo operador >.

Véase también
"> (mayor que)" a pagina 218

_height

Sintaxis
nombre_instancia._height
nombre_instancia._height = *valor*;

Argumentos
nombre_instancia Un nombre de instancia de un clip de película para el que se va a establecer o recuperar la propiedad _height.

valor Un número entero que especifica la altura de la película en píxeles.

Descripción
Propiedad; establece y recupera la altura del espacio ocupado por el contenido de una película. En versiones anteriores de Flash, _height y _width eran propiedades de sólo lectura, en Flash 5 estas propiedades se pueden establecer.

Reproductor
Flash 4 o posterior.

Ejemplo
El código de ejemplo siguiente establece la altura y la anchura de un clip de película cuando el usuario hace clic sobre un botón.

```
onClipEvent(mouseDown) {
   _width=200;
   _height=200;
}
```

_highquality

Sintaxis

```
_highquality = value;
```

Argumentos

valor El nivel de suavizado aplicado a la película. Especifique 2 (MEJOR) para aplicar alta calidad con el suavizado de mapa de bits siempre activado. Especifique 1 (alta calidad) para aplicar suavizado; esto suavizará los mapas de bits si la película no contiene animación. Especifique 0 (baja calidad) para evitar el suavizado.

Descripción

Propiedad (global); especifica el nivel de suavizado aplicado a la película actual.

Reproductor

Flash 4 o posterior.

Véase también

"_quality" a pagina 349
"toggleHighQuality" a pagina 381

if

Sintaxis

```
if(condición) {

sentencia;

{
```

Argumentos

condición Una expresión que evalúa como `true` o `false`. Por ejemplo, `if(name == "Erica")`, evalúa la variable `name` para ver si es "Erica".

sentencia Las instrucciones que se deben ejecutar si o cuando la condición evalúa como `true`.

Descripción

Acción; evalúa una condición para determinar la siguiente acción en una película. Si la condición es `true`, Flash ejecuta la sentencia que va a continuación. Utilice `if` parta crear lógica de ramas en sus scripts.

Reproductor

Flash 4 o posterior.

Véase también

"else" a pagina 265
"for" a pagina 268
"for...in" a pagina 270

ifFrameLoaded

Sintaxis

```
ifFrameLoaded(scene, fotograma) {
sentencia;}
```

```
ifFrameLoaded(fotograma) {
sentencia;}
```

Argumentos

escena La escena que se está consultando.

fotograma El número o etiqueta de fotograma que se va a cargar antes de que se ejecute la sentencia siguiente.

Descripción

Acción; comprueba si el contenido de un fotograma específico está disponible localmente. Utilice `ifFrameLoaded` para empezar a reproducir una animación sencilla mientras que se descarga el resto de la película en el sistema local. La diferencia entre utilizar `_framesloaded` y `ifFrameLoaded` es que `_framesloaded` le permite agregar sentencias `if`, o `else`, mientras que la acción `ifFrameLoaded` le permite especificar un número de fotogramas en una sentencia sencilla.

Reproductor

Flash 3 o posterior. La acción `ifFrameLoaded` se ha desestimado en Flash 5; se recomienda la utilización de la acción `_framesloaded`.

Véase también

"_framesloaded" a pagina 271

#include

Sintaxis

```
#include "filename.as";
```

Argumentos

filename.as El nombre de archivo a incluir; `.as` es la extensión de archivo recomendada.

Descripción

Acción; incluye el contenido del archivo especificado en el argumento cuando se prueba, se publica o se exporta una película. Se invoca a la acción `#include` cuando prueba, publica o exporta. Se comprueba la acción `#include` cuando se produce una comprobación de sintaxis.

Reproductor

N/A.

Infinity

Sintaxis
```
Infinity
```

Argumentos
Ninguno.

Descripción
Variable de nivel superior; una variable predefinida con el valor ECMA-262 para infinito.

Reproductor
Flash 5 o posterior.

int

Sintaxis
```
int(valor);
```

Argumentos
valor Un número que se va a redondear en un número entero.

Descripción
Función; convierte un número decimal en su valor de número entero más cercano.

Reproductor
Flash 4 o posterior. Esta función se ha desestimado en Flash 5; se recomienda la utilización del método `Math.floor`.

Véase también
"Math.floor" a pagina 303

isFinite

Sintaxis
```
isFinite(expresión);
```

Argumentos
expresión El Booleano, variable o expresión que se va a evaluar.

Descripción
Función de nivel superior; evalúa el argumento y devuelve `true` si es un número finito y `false` si es infinito o infinito negativo. La presencia de infinito o de infinito negativo indica una condición de error matemático como una división por 0.

Reproductor
Flash 5 o posterior.

Ejemplo
A continuación se muestran ejemplos de los valores devueltos por `isFinite`:

`isFinite(56)` devuelve `true`

`isFinite(Number.POSITIVE_INFINITY)` devuelve `false`

`isNaN(Number.POSITIVE_INFINITY)` devuelve `false`

isNaN

Sintaxis
`isNaN(`*`expresión`*`);`

Argumentos
`expresión` El Booleano, variable o expresión que se va a evaluar.

Descripción
Función de nivel superior; evalúa el argumento y devuelve `true` si el valor no es un número (`NaN`), indicando la presencia de errores matemáticos.

Reproductor
Flash 5 o posterior.

Ejemplo
A continuación se muestra el valor devuelto por `isNan`:

`isNan("Tree")` devuelve `true`

`isNan(56)` devuelve `false`

`isNaN(Number.POSITIVE_INFINITY)` devuelve `false`

Key (objeto)

El objeto Key es un objeto de nivel superior al que puede acceder sin utilizar un constructor. Utilice los métodos del objeto Key para crear una interfaz que pueda ser controlada por cualquier usuario con un teclado estándar. Las propiedades del objeto Key son constantes que representan las teclas que se utilizan con mayor frecuencia para controlar juegos. Consulte el Apéndice B, "Teclas del teclado y Valores de código de tecla" para ver una lista completa de los valores de código de tecla.

Ejemplo

```
onClipEvent (enterFrame) {
    if(Key.isDown(Key.RIGHT)) {
        setProperty ("", _x, _x+10);
    }
}
or
onClipEvent (enterFrame) {
    if(Key.isDown(39)) {
        setProperty("", _x, _x+10);
    }
}
```

Resumen de los métodos de un objeto Key

Método	Descripción
`getAscii;`	Devuelve el valor ASCII de la última tecla presionada.
`getCode;`	Devuelve el código de tecla virtual de la última tecla presionada.
`isDown;`	Devuelve `true` si la tecla especificada en el argumento se presiona.
`isToggled;`	Devuelve `true` si está activada la tecla Bloq Num o Bloq Mayús.

Resumen de las propiedades de un objeto Key

Todas las propiedades del objeto Key son constantes.

Propiedad	Descripción
BACKSPACE	Constante asociada con el valor del código de tecla para la tecla Retroceso (9).
CAPSLOCK	Constante asociada con el valor del código de tecla para la tecla Bloq Mayús (20).
CONTROL	Constante asociada con el valor del código de tecla para la tecla Control (17).
DELETEKEY	Constante asociada con el valor del código de tecla para la tecla Supr (46).
DOWN	Constante asociada con el valor del código de tecla para la tecla Flecha abajo (40).
END	Constante asociada con el valor del código de tecla para la tecla Fin (35).
ENTER	Constante asociada con el valor del código de tecla para la tecla Intro (13).
ESCAPE	Constante asociada con el valor del código de tecla para la tecla Escape (27).
HOME	Constante asociada con el valor del código de tecla para la tecla Inicio (36).
INSERT	Constante asociada con el valor del código de tecla para la tecla Insert (45).
LEFT	Constante asociada con el valor del código de tecla para la tecla Flecha izquierda (37).
PGDN	Constante asociada con el valor del código de tecla para la tecla Av Pág (34).
PGUP	Constante asociada con el valor del código de tecla para la tecla Re Pág (33).
RIGHT	Constante asociada con el valor del código de tecla para la tecla Flecha derecha (39).
SHIFT	Constante asociada con el valor del código de tecla para la tecla Mayús (16).
SPACE	Constante asociada con el valor del código de tecla para la Barra espaciadora (32).
TAB	Constante asociada con el valor del código de tecla para la tecla Tabulador (9).
UP	Constante asociada con el valor del código de tecla para la tecla Flecha arriba (38).

Key.BACKSPACE

Sintaxis
`Key.BACKSPACE`

Argumentos
Ninguno.

Descripción
Propiedad; constante asociada con el valor del código de tecla para la tecla Retroceso (9).

Reproductor
Flash 5 o posterior.

Key.CAPSLOCK

Sintaxis
`Key.CAPSLOCK`

Argumentos
Ninguno.

Descripción
Propiedad; constante asociada con el valor del código de tecla para la tecla Bloq Mayús (20).

Reproductor
Flash 5 o posterior.

Key.CONTROL

Sintaxis
`Key.CONTROL`

Argumentos
Ninguno.

Descripción
Propiedad; constante asociada con el valor del código de tecla para la tecla Control (17).

Reproductor
Flash 5 o posterior.

Key.DELETEKEY

Sintaxis
```
Key.DELETEKEY
```

Argumentos
Ninguno.

Descripción
Propiedad; constante asociada con el valor del código de tecla para la tecla Supr (46).

Reproductor
Flash 5 o posterior.

Key.DOWN

Sintaxis
```
Key.DOWN
```

Argumentos
Ninguno.

Descripción
Propiedad; constante asociada con el valor del código de tecla para la tecla Flecha abajo (40).

Reproductor
Flash 5 o posterior.

Key.END

Sintaxis
```
Key.END
```

Argumentos
Ninguno.

Descripción
Propiedad; constante asociada con el valor del código de tecla para la tecla Fin (35).

Reproductor
Flash 5 o posterior.

Key.ENTER

Sintaxis
```
Key.ENTER
```

Argumentos
Ninguno.

Descripción
Propiedad; constante asociada con el valor del código de tecla para la tecla Intro (13).

Reproductor
Flash 5 o posterior.

Key.ESCAPE

Sintaxis
```
Key.ESCAPE
```

Argumentos
Ninguno.

Descripción
Propiedad; constante asociada con el valor del código de tecla para la tecla Escape (27).

Reproductor
Flash 5 o posterior.

Key.getAscii

Sintaxis
```
Key.getAscii();
```

Argumentos
Ninguno.

Descripción
Método; devuelve el código ASCII de la última tecla presionada o soltada.

Reproductor
Flash 5 o posterior.

Key.getCode

Sintaxis
```
Key.getCode();
```

Argumentos
Ninguno.

Descripción
Método; devuelve el código de tecla virtual de la última tecla presionada. Utilice la información del Apéndice B, "Teclas del teclado y Valores de código de tecla" para hacer coincidir el valor de código de tecla devuelto con la tecla virtual de un teclado estándar.

Reproductor
Flash 5 o posterior.

Key.HOME

Sintaxis
```
Key.HOME
```

Argumentos
Ninguno.

Descripción
Propiedad; constante asociada con el valor del código de tecla para la tecla Inicio (36).

Reproductor
Flash 5 o posterior.

Key.INSERT

Sintaxis
```
Key.INSERT
```

Argumentos
Ninguno.

Descripción
Propiedad; constante asociada con el valor del código de tecla para la tecla Insert (45).

Reproductor
Flash 5 o posterior.

Key.isDown

Sintaxis
`Key.isDown(`*`código_tecla`*`);`

Argumentos
`código_tecla` El valor de código de tecla asignado a una tecla específica o a una propiedad de objeto Key asociada con una tecla específica. En el Apéndice B, "Teclas del teclado y Valores de código de tecla", se muestra una lista de los códigos clave asociados con las teclas de un teclado estándar.

Descripción
Método; devuelve `true` si la tecla especificada en *`código_tecla`* se presiona. En Macintosh, los valores de código de tecla para las teclas Bloq Mayús y Bloq Num son idénticos.

Reproductor
Flash 5 o posterior.

Key.isToggled

Sintaxis
`Key.isToggled(`*`código_tecla`*`)`

Argumentos
`código_tecla` El código de tecla de Bloq Mayús (20) o Bloq Num (144).

Descripción
Método; devuelve `true` si está activada la tecla Bloq Num o Bloq Mayús. En Macintosh, los valores de código de tecla para estas teclas son idénticos.

Reproductor
Flash 5 o posterior.

Key.LEFT

Sintaxis
`Key.LEFT`

Argumentos
Ninguno.

Descripción
Propiedad; constante asociada con el valor del código de tecla para la tecla Flecha izquierda (37).

Reproductor
Flash 5 o posterior.

Key.PGDN

Sintaxis
```
Key.PGDN
```

Argumentos
Ninguno.

Descripción
Propiedad; constante asociada con el valor del código de tecla para la tecla Av Pág (34).

Reproductor
Flash 5 o posterior.

Key.PGUP

Sintaxis
```
Key.PGUP
```

Argumentos
Ninguno.

Descripción
Propiedad; constante asociada con el valor del código de tecla para la tecla Re Pág (33).

Reproductor
Flash 5 o posterior.

Key.RIGHT

Sintaxis
```
Key.RIGHT
```

Argumentos
Ninguno.

Descripción
Propiedad; constante asociada con el valor del código de tecla para la tecla Flecha derecha (39).

Reproductor
Flash 5 o posterior.

Key.SHIFT

Sintaxis
```
Key.SHIFT
```

Argumentos
Ninguno.

Descripción
Propiedad; constante asociada con el valor del código de tecla para la tecla Mayús (16).

Reproductor
Flash 5 o posterior.

Key.SPACE

Sintaxis
```
Key.SPACE
```

Argumentos
Ninguno.

Descripción
Propiedad; constante asociada con el valor del código de tecla de la Barra espaciadora (32).

Reproductor
Flash 5 o posterior.

Key.TAB

Sintaxis
```
Key.TAB
```

Argumentos
Ninguno.

Descripción
Propiedad; constante asociada con el valor del código de tecla de la tecla Tabulador (9).

Reproductor
Flash 5 o posterior.

Key.UP

Sintaxis
`Key.UP`

Argumentos
Ninguno.

Descripción
Propiedad; constante asociada con el valor del código de tecla de la tecla Flecha arriba (38).

Reproductor
Flash 5 o posterior.

le (menor o igual que–específico de cadena)

Sintaxis
`expresión1` `le` *`expresión2`*

Argumentos
`expresión1,expresión2` Números, cadenas o variables.

Descripción
Operador (de comparación); compara *`expresión1`* con *`expresión2`* y devuelve `true` si *`expresión1`* es menor o igual que *`expresión2`*; en caso contrario, devuelve `false`.

Reproductor
Flash 4 o posterior. Este operador se ha desestimado en Flash 5; se recomienda la utilización del nuevo operador `<=` .

Véase también
"<= (menor o igual que)" a pagina 215

length

Sintaxis
`length(`*`expresión`*`);`

`length(`*`variable`*`);`

Argumentos
`expresión` Cualquier cadena.

`variable` El nombre de una variable.

Descripción
Función de cadena; devuelve la longitud de la cadena o nombre de variable especificado.

Reproductor
Flash 4 o posterior. Esta función, junto con todas las funciones de cadena se ha desestimado en Flash 5. Se le recomienda que utilice los métodos y la propiedad `length` del objeto String para realizar las mismas operaciones.

Ejemplo
El ejemplo siguiente devuelve el valor de la cadena `Hello`:

```
length("Hello")
```

El resultado es 5.

Véase también
“" " (delimitador de cadena)” a pagina 370
“String.length” a pagina 375

_level

Sintaxis
```
_levelN;
```

Argumentos
N Un número entero no negativo que especifica un nivel de profundidad. El valor predeterminado `_level` está establecido en 0, la película en la base de la jerarquía.

Descripción
Propiedad; una referencia a la Línea de tiempo de la película raíz de *levelN*. Debe cargar películas utilizando la acción `loadMovie`, antes de seleccionarles destino utilizando la propiedad `_level`.

En Flash Player, las películas llevan un número asignado de acuerdo con el orden en que se cargaron. La película cargada en primer lugar se carga en el nivel inferior, es decir, el nivel 0. La película del nivel 0 establece la velocidad de los fotogramas, el color de fondo y el tamaño de los fotogramas de todas las demás películas cargadas a continuación. Las películas se apilan en niveles con un número más alto por encima de la película en el nivel 0.Se hace referencia al nivel en el que reside un clip de película como nivel de profundidad o profundidad.

Reproductor
Flash 4 o posterior.

Ejemplo

El ejemplo siguiente detiene la Línea de tiempo de la película en el nivel 0.

```
_level0.stop();
```

El ejemplo siguiente envía la Línea de tiempo de la película en el nivel 4 al fotograma 5. La película del nivel 4 debe haberse cargado previamente con una acción `loadMovie`.

```
_level4.gotoAndStop(5);
```

Véase también

"loadMovie" a pagina 294
"MovieClip.swapDepths" a pagina 326

loadMovie

Sintaxis

```
loadMovie(url [,ubicación/destino, variables]]);
```

Argumentos

`url` Una URL absoluta o relativa para el archivo SWF que se va a cargar. Una ruta relativa debe ser relativa respecto al SWF. La URL debe estar en el mismo subdominio que la URL donde reside actualmente la película. Para utilizarlos en Flash Player o para realizar pruebas en el modo de prueba de película en el entorno de creación de Flash, todos los archivos SWF tienen que guardarse en la misma carpeta y los nombres de archivo no pueden incluir especificaciones de carpeta ni de unidad de disco.

`destino` Un argumento opcional que especifica un clip de película de destino que se sustituirá con la película cargada. La película cargada hereda las propiedades de posición, rotación y escala del clip de película de destino. Especificar *`destino`* es lo mismo que especificar el (nivel) *`ubicación`* de una película de destino, no debería especificar ambos.

`ubicación` Un argumento opcional que especifica el nivel en el que la película está cargada. La película cargada hereda las propiedades de posición, rotación y escala del clip de película de destino. Para cargar una nueva película además de las películas existentes, especifique un nivel que no esté ocupado por otra película. Para sustituir una película existente con la película cargada, especifique un nivel que esté actualmente ocupado por otra película. Para sustituir la película original y descargar todos los niveles, cargue la nueva película en el nivel 0. La película del nivel 0 establece la velocidad de los fotogramas, el color de fondo y el tamaño de los fotogramas de todas las demás películas cargadas.

variables Un argumento opcional que especifica un método para enviar variables asociadas con la película que se va a cargar. El argumento debe ser la cadena "GET" o "POST". Si no hay variables, omita este argumento; en caso contrario, especifique si se cargan las variables utilizando un método `GET` o `POST`. `GET` anexa las variables al final de la URL y se utiliza para un número pequeño de variables. `POST` envía las variables en un encabezado HTTP aparte y se utiliza para cadenas largas de variables.

Descripción

Acción; reproduce películas adicionales sin cerrar Flash Player. Por lo general, Flash Player muestra una sola película de Flash Player (archivo SWF) y después se cierra. La acción `loadMovie` permite ver varias películas al mismo tiempo o cambiar entre películas sin cargar otro documento HTML.

Puede cargar películas en los niveles que ya tienen archivos SWF cargados. Al hacerlo, la nueva película sustituye al archivo SWF existente. Si carga una nueva película en el nivel 0, todos los niveles se descargarán y el nivel 0 se sustituirá por el nuevo archivo. Utilice la acción `loadVariables` para mantener la película activa y actualizar las variables con nuevos valores.

Utilice la acción `unloadMovie` para eliminar películas cargadas con la acción `loadMovie`.

Reproductor

Flash 3 o posterior.

Ejemplo

Esta sentencia `loadMovie` está anexada a un botón de navegación con la etiqueta Productos. Hay un clip de película invisible en el Escenario con el nombre de instancia `dropZone`. La acción loadMovie utiliza este clip de película como parámetro de destino para cargar los productos del archivo SWF, en la posición correcta del Escenario.

```
on(release) {
    loadMovie("products.swf",_root.dropZone);
}
```

Véase también

"unloadMovie" a pagina 384
"_level" a pagina 293

loadVariables

Sintaxis

```
loadVariables (url ,ubicación [, variables]);
```

Argumentos

url Una URL absoluta o relativa donde están ubicadas las variables. El anfitrión de la URL debe estar en el mismo subdominio que la película cuando se accede utilizando un navegador Web.

ubicación Un nivel o destino para recibir las variables. En Flash Player, los archivos de película llevan un número asignado de acuerdo con el orden en que se cargaron. La primera película se carga en el nivel inferior (nivel 0). Dentro de la acción `loadMovie`, debe especificar un número de nivel para cada película sucesiva. Este argumento es opcional.

variables Un argumento opcional que especifica un método para enviar variables. Si no hay variables, omita este argumento; en caso contrario, especifique si se cargan las variables utilizando un método `GET` o `POST`. `GET` anexa las variables al final de la URL y se utiliza para un número pequeño de variables. `POST` envía las variables en un encabezado HTTP aparte y se utiliza para cadenas largas de variables.

Descripción

Acción; lee los datos de un archivo externo, como un archivo de texto o texto generado por un script CGI, Páginas de Active Server (ASP) o Página de Personal Home (PHP) y establece los valores para las variables de una película o clip de película. Esta acción también puede utilizarse para actualizar variables en la película activa con nuevos valores.

El testo de la URL especificada debe estar en formato MIME estándar *application/x-www-urlformencoded* (un formato estándar utilizado para scripts CGI). La película y las variables que se van a cargar deben residir en el mismo subdominio. Se puede especificar cualquier número de variables. Por ejemplo, la siguiente frase define varias variables:

```
company=Macromedia&address=600+Townsend&city=
San+Francisco&zip=94103
```

Reproductor

Flash 4 o posterior.

Ejemplo

Este ejemplo carga información de un archivo de texto en los campos de texto de la Línea de tiempo principal (nivel 0). Los nombres de variables de los campos de texto deben coincidir con los nombres de variables del archivo data.txt.

```
on(release) {
    loadVariables("data.txt", 0);
}
```

Véase también

"getURL" a pagina 275
"MovieClip.loadMovie" a pagina 321
"MovieClip.loadVariables" a pagina 322

lt (menor que, específico de cadena)

Sintaxis

expresión1 `lt` *expresión2*

Argumentos

expresión1, expresión2 Números, cadenas o variables.

Descripción

Operador (de comparación); compara *expresión1* con *expresión2* y devuelve `true` si *expresión1* es menor que *expresión2*; en caso contrario, devuelve `false`.

Reproductor

Flash 4 o posterior. Este operador se ha desestimado en Flash 5; se recomienda la utilización del nuevo operador < (menor que).

Véase también

"< (menor que)" a pagina 212

Matemáticas (objeto)

El objeto Math es un objeto de nivel superior al que puede acceder sin utilizar un constructor.

Utilice los métodos y propiedades de este objeto para acceder y manipular constantes y funciones matemáticas. Todas las propiedades y métodos del objeto Math son estáticos y deben llamarse utilizando la sintaxis `Math.method(`*argumento*`)` o `Math.constant`. En ActionScript, las constantes se definen con la máxima precisión de números con coma flotante IEEE-754 de doble precisión.

El objeto Math está admitido por completo en Flash Player 5. En Flash Player 4, los métodos del objeto Math funcionan, pero se emulan usando aproximaciones y es posible que no sean tan precisos como las funciones matemáticas no emuladas que soporta Flash Player 5.

Algunos de los métodos del objeto Math toman el radián de un ángulo como argumento. Puede utilizar la ecuación siguiente para calcular valores de radianes o sencillamente pase la ecuación (introduciendo un valor para grados) para el argumento radián.

Para calcular un valor de radián, utilice esta fórmula:

```
radián = Math.PI/180 * grado
```

A continuación se muestra un ejemplo de pasar una ecuación como argumento para calcular el seno de un ángulo de 45 grados:

`Math.SIN(Math.PI/180 * 45)` es lo mismo que `Math.SIN(.7854)`

Resumen de los métodos del objeto Math

Método	Descripción
`abs`	Calcula un valor absoluto.
`acos`	Calcula un arco coseno.
`asin`	Calcula un arco seno.
`atan`	Calcula un arco tangente.
`atan2`	Calcula un ángulo desde el eje x hasta el punto.
`ceil`	Redondea un número al entero más cercano hacia arriba.
`cos`	Calcula un coseno.
`exp`	Calcula un valor exponencial.
`floor`	Redondea un número al entero más cercano hacia abajo.
`log`	Calcula un logaritmo natural.
`max`	Devuelve el mayor de dos números enteros.
`min`	Devuelve el menor de dos números enteros.
`pow`	Calcula *x* elevado a la potencia de *y*.
`aleatorio`	Devuelve un número seudoaleatorio ente 0,0 y 1,1.
`round`	Redondea al número entero más cercano.
`sin`	Calcula un seno.
`sqrt`	Calcula una raíz cuadrada.
`tan`	Calcula una tangente.

Resumen de las propiedades de un objeto Math

Todas las propiedades del objeto Math son constantes.

Propiedad	Descripción
E	La constante de Euler y la base de los logaritmos naturales (aproximadamente 2,718).
LN2	El logaritmo natural de 2 (aproximadamente 0, 693).
LOG2E	El logaritmo en base 2 de *e* (aproximadamente 1,442).
LN10	El logaritmo natural de 10 (aproximadamente 2, 302).
LOG10E	El logaritmo en base 10 de *e* (aproximadamente 0,434).
PI	La relación entre la circunferencia de un círculo y su diámetro (aproximadamente 3,14159).
SQRT1_2	El recíproco de la raíz cuadrada de 1/2 (aproximadamente 0, 707).
SQRT2	La raíz cuadrada de 2 (aproximadamente 1,414).

Math.abs

Sintaxis

```
Math.abs(x);
```

Argumentos

x Cualquier número.

Descripción

Método; calcula y devuelve un valor absoluto para el número especificado por el argumento *x*.

Reproductor

Flash 5 o posterior. En Flash Player 4, se emulan los métodos y propiedades del objeto Math usando aproximaciones y es posible que no sea tan preciso como las funciones matemáticas no emuladas que incluye Flash Player 5.

Math.acos

Sintaxis

```
Math.acos(x);
```

Argumentos

x Un número entero de -1,0 a 1,0.

Descripción
Método; calcula y devuelve el arco coseno del número especificado en el argumento *x*, en radianes.

Reproductor
Flash 5 o posterior. En Flash Player 4, se emulan los métodos y propiedades del objeto Math usando aproximaciones y es posible que no sea tan preciso como las funciones matemáticas no emuladas que incluye Flash Player 5.

Math.asin

Sintaxis
```
Math.asin(x);
```

Argumentos
x Un número entero de -1,0 a 1,0.

Descripción
Método; calcula y devuelve el arco coseno del número especificado en el argumento *x*, en radianes.

Reproductor
Flash 5 o posterior. En Flash Player 4, se emulan los métodos y propiedades del objeto Math usando aproximaciones y es posible que no sea tan preciso como las funciones matemáticas no emuladas que incluye Flash Player 5.

Math.atan

Sintaxis
```
Math.atan(x);
```

Argumentos
x Cualquier número.

Descripción
Método; calcula y devuelve el arco tangente del número especificado en el argumento *x*. El valor devuelto está entre pi negativo dividido por 2 y pi positivo dividido por 2.

Reproductor
Flash 5 o posterior. En Flash Player 4, se emulan los métodos y propiedades del objeto Math usando aproximaciones y es posible que no sea tan preciso como las funciones matemáticas no emuladas que incluye Flash Player 5.

Math.atan2

Sintaxis

```
Math.atan2(y, x);
```

Argumentos

x Un número que especifica la coordenada *x* del punto.

y Un número que especifica la coordenada *y* del punto.

Descripción

Método; calcula y devuelve el arco tangente de *y*/*x* en radianes. El valor devuelto representa el ángulo contrario de un triángulo recto, donde *x* es la longitud del lado adyacente y *y* es la longitud del lado contrario.

Reproductor

Flash 5 o posterior. En Flash Player 4, se emulan los métodos y propiedades del objeto Math usando aproximaciones y es posible que no sea tan preciso como las funciones matemáticas no emuladas que incluye Flash Player 5.

Math.ceil

Sintaxis

```
Math.ceil(x);
```

Argumentos

x Un número o expresión.

Descripción

Método; devuelve el plano superior del número o expresión especificada. El plano superior de un número es el número entero más cercano que es mayor o igual que el número.

Reproductor

Flash 5 o posterior. En Flash Player 4, se emulan los métodos y propiedades del objeto Math usando aproximaciones y es posible que no sea tan preciso como las funciones matemáticas no emuladas que incluye Flash Player 5.

Math.cos

Sintaxis

```
Math.cos(x);
```

Argumentos

x Un ángulo medido en radianes.

Descripción
Método; devuelve el coseno (un valor de -1,0 a 1,09 del ángulo especificado en el argumento *x*. El ángulo *x* debe especificarse en radianes. Utilice la información que se explicó en la introducción del objeto Math para calcular un radián.

Reproductor
Flash 5 o posterior. En Flash Player 4, se emulan los métodos y propiedades del objeto Math usando aproximaciones y es posible que no sea tan preciso como las funciones matemáticas no emuladas que incluye Flash Player 5.

Math.E

Sintaxis
```
Math.E
```

Argumentos
Ninguno.

Descripción
Constante; una constante matemática para la base de los logaritmos naturales, expresada como *e*. El valor aproximado de *e* es 2,71828.

Reproductor
Flash 5 o posterior. En Flash Player 4, se emulan los métodos y propiedades del objeto Math usando aproximaciones y es posible que no sea tan preciso como las funciones matemáticas no emuladas que incluye Flash Player 5.

Math.exp

Sintaxis
```
Math.exp(x);
```

Argumentos
x El exponente; un número o expresión.

Descripción
Método; devuelve el valor de la base del logaritmo natural (*e*), a la potencia del exponente especificado en el argumento *x*. La constante `Math.E` puede proporcionar el valor de *e*.

Reproductor
Flash 5 o posterior. En Flash Player 4, se emulan los métodos y propiedades del objeto Math usando aproximaciones y es posible que no sea tan preciso como las funciones matemáticas no emuladas que incluye Flash Player 5.

Math.floor

Sintaxis
```
Math.floor(x);
```

Argumentos
x Un número o expresión.

Descripción
Método; devuelve el plano inferior del número o expresión especificada en el argumento *x*. El plano inferior es el número entero más cercano que es menor o igual que el número o expresión especificada.

Reproductor
Flash 5 o posterior. En Flash Player 4, se emulan los métodos y propiedades del objeto Math usando aproximaciones y es posible que no sea tan preciso como las funciones matemáticas no emuladas que incluye Flash Player 5.

Ejemplo
El ejemplo siguiente devuelve un valor de 12.

```
Math.floor(12.5);
```

Math.log

Sintaxis
```
Math.log(x);
```

Argumentos
x Un número o expresión con un valor mayor que 0.

Descripción
Método; devuelve el logaritmo natural del argumento *x*.

Reproductor
Flash 5 o posterior. En Flash Player 4, se emulan los métodos y propiedades del objeto Math usando aproximaciones y es posible que no sea tan preciso como las funciones matemáticas no emuladas que incluye Flash Player 5.

Math.LOG2E

Sintaxis
```
Math.LOG2E
```

Argumentos
Ninguno.

Descripción
Constante; una constante matemática para el logaritmo de base 2 de la constante *e* (`Math.E`), expresada como $\log_e 2$, con un valor aproximado de 1,442695040888963387.

Reproductor
Flash 5 o posterior. En Flash Player 4, se emulan los métodos y propiedades del objeto Math usando aproximaciones y es posible que no sea tan preciso como las funciones matemáticas no emuladas que incluye Flash Player 5.

Math.LOG10E

Sintaxis
```
Math.LOG10E
```

Argumentos
Ninguno.

Descripción
Constante; una constante matemática para el logaritmo en base 10 de la constante *e* (`Math.E`), expresada como $\log_{10} e$, con un valor aproximado de 0,43429448190325181667.

Reproductor
Flash 5 o posterior. En Flash Player 4, se emulan los métodos y propiedades del objeto Math usando aproximaciones y es posible que no sea tan preciso como las funciones matemáticas no emuladas que incluye Flash Player 5.

Math.LN2

Sintaxis
```
Math.LN2
```

Argumentos
Ninguno.

Descripción
Constante; una constante matemática para el logaritmo natural de 2, expresada como $\log_e 2$, con un valor aproximado de 0,69314718055994528623.

Reproductor
Flash 5 o posterior. En Flash Player 4, se emulan los métodos y propiedades del objeto Math usando aproximaciones y es posible que no sea tan preciso como las funciones matemáticas no emuladas que incluye Flash Player 5.

Math.LN10

Sintaxis
```
Math.LN10
```

Argumentos
Ninguno.

Descripción
Constante; una constante matemática para el logaritmo natural de base 10, expresada como $\log_e 10$, con un valor aproximado de 2,3025850929940459011.

Reproductor
Flash 5 o posterior. En Flash Player 4, se emulan los métodos y propiedades del objeto Math usando aproximaciones y es posible que no sea tan preciso como las funciones matemáticas no emuladas que incluye Flash Player 5.

Math.max

Sintaxis
```
Math.max(x , y);
```

Argumentos
x Un número o expresión.

y Un número o expresión.

Descripción
Método; evalúa *x* e *y* y devuelve el valor mayor.

Reproductor
Flash 5 o posterior. En Flash Player 4, se emulan los métodos y propiedades del objeto Math usando aproximaciones y es posible que no sea tan preciso como las funciones matemáticas no emuladas que incluye Flash Player 5.

Math.min

Sintaxis
```
Math.min(x , y);
```

Argumentos
x Un número o expresión.

y Un número o expresión.

Descripción
Método; evalúa *x* e *y* y devuelve el valor menor.

Reproductor
Flash 5 o posterior. En Flash Player 4, se emulan los métodos y propiedades del objeto Math usando aproximaciones y es posible que no sea tan preciso como las funciones matemáticas no emuladas que incluye Flash Player 5.

Math.PI

Sintaxis
```
Math.PI
```

Argumentos
Ninguno.

Descripción
Constante; una constante matemática para el radio de la circunferencia de un círculo con su diámetro, expresado como pi, con un valor de 3,14159265358979.

Reproductor
Flash 5 o posterior. En Flash Player 4, se emulan los métodos y propiedades del objeto Math usando aproximaciones y es posible que no sea tan preciso como las funciones matemáticas no emuladas que incluye Flash Player 5.

Math.pow

Sintaxis
```
Math.pow(x , y);
```

Argumentos
x Un número que se va a elevar a una potencia.

y Un número que especifica una potencia del argumento *x* a la que se eleva.

Descripción
Método; calcula y devuelve *x* a la potencia de *y*, x^y.

Reproductor
Flash 5 o posterior. En Flash Player 4, se emulan los métodos y propiedades del objeto Math usando aproximaciones y es posible que no sea tan preciso como las funciones matemáticas no emuladas que incluye Flash Player 5.

Math.random

Sintaxis
```
Math.random();
```

Argumentos
Ninguno.

Descripción
Método; devuelve un número seudoaleatorio ente 0,0 y 1,0.

Reproductor
Flash 5 o posterior. En Flash Player 4, se emulan los métodos y propiedades del objeto Math usando aproximaciones y es posible que no sea tan preciso como las funciones matemáticas no emuladas que incluye Flash Player 5.

Véase también
"random" a pagina 350

Math.round

Sintaxis
```
Math.round(x);
```

Argumentos
x Cualquier número.

Descripción
Método; redondea el valor del argumento *x* hacia arriba o hacia abajo al número entero más cercano y devuelve el valor.

Reproductor
Flash 5 o posterior. En Flash Player 4, se emulan los métodos y propiedades del objeto Math usando aproximaciones y es posible que no sea tan preciso como las funciones matemáticas no emuladas que incluye Flash Player 5.

Math.sin

Sintaxis
```
Math.sin(x);
```

Argumentos
x Un ángulo medido en radianes.

Descripción
Método; calcula y devuelve el seno del ángulo especificado, en radianes. Utilice la información que se explicó en la introducción del objeto Math para calcular un radián.

Reproductor
Flash 5 o posterior. En Flash Player 4, se emulan los métodos y propiedades del objeto Math usando aproximaciones y es posible que no sea tan preciso como las funciones matemáticas no emuladas que incluye Flash Player 5.

Véase también
"Matemáticas (objeto)" a pagina 297

Math.sqrt

Sintaxis
```
Math.sqrt(x);
```

Argumentos
x Cualquier número o expresión mayor o igual que 0.

Descripción
Método; calcula y devuelve la raíz cuadrada del número especificado.

Reproductor
Flash 5 o posterior. En Flash Player 4, se emulan los métodos y propiedades del objeto Math usando aproximaciones y es posible que no sea tan preciso como las funciones matemáticas no emuladas que incluye Flash Player 5.

Math.SQRT1_2

Sintaxis
```
Math.SQRT1_2
```

Argumentos
Ninguno.

Descripción
Constante; una constante matemática para el recíproco de la raíz cuadrada de un medio (1/2), con un valor aproximado de 0,707106781186.

Reproductor
Flash 5 o posterior. En Flash Player 4, se emulan los métodos y propiedades del objeto Math usando aproximaciones y es posible que no sea tan preciso como las funciones matemáticas no emuladas que incluye Flash Player 5.

Math.SQRT2

Sintaxis
```
Math.SQRT2
```

Argumentos
Ninguno.

Descripción
Constante; una constante matemática para la raíz cuadrada de 2, expresada como , con un valor aproximado de 1,414213562373.

Reproductor
Flash 5 o posterior. En Flash Player 4, se emulan los métodos y propiedades del objeto Math usando aproximaciones y es posible que no sea tan preciso como las funciones matemáticas no emuladas que incluye Flash Player 5.

Math.tan

Sintaxis
```
Math.tan(x);
```

Argumentos
x Un ángulo medido en radianes.

Descripción
Método; calcula y devuelve la tangente del ángulo especificado. Utilice la información que se explicó en la introducción del objeto Math para calcular un radián.

Reproductor
Flash 5 o posterior. En Flash Player 4, se emulan los métodos y propiedades del objeto Math usando aproximaciones y es posible que no sea tan preciso como las funciones matemáticas no emuladas que incluye Flash Player 5.

maxscroll

Sintaxis
```
nombre_variable.maxscroll = x
```

Argumentos
nombre_variable El nombre de la variable asociada con un campo de texto.

x El número de línea que es el valor máximo permitido para la propiedad `scroll`, basado en la altura del campo de texto. Este es un valor de sólo lectura establecido por Flash.

Descripción
Propiedad; una propiedad de sólo lectura que funciona con la propiedad `scroll` para controlar la forma de mostrar información en un campo de texto. Este propiedad puede recuperarse, pero no modificarse.

Reproductor
Flash 4 o posterior.

Véase también
"scroll" a pagina 353

mbchr

Sintaxis
```
mbchr(número);
```

Argumentos
número El número que se convierte en carácter multibyte.

Descripción
Función de cadena; convierte un número de código ASCII en un carácter multibyte.

Reproductor
Flash 4 o posterior. Esta función se ha desestimado en Flash 5; se recomienda la utilización del método `String.fromCharCode`.

Véase también
"String.fromCharCode" a pagina 374

mblength

Sintaxis
```
mblength(cadena);
```

Argumentos
cadena Una cadena.

Descripción
Función de cadena; devuelve la longitud de la cadena de caracteres multibyte.

Reproductor
Flash 4 o posterior. Esta función se ha desestimado en Flash 5; se recomienda la utilización del objeto String y sus métodos.

mbord

Sintaxis
`mbord(carácter);`

Argumentos
carácter El carácter que se convierte en un número multibyte.

Descripción
Función de cadena; convierte el carácter especificado en un número multibyte.

Reproductor
Flash 4 o posterior. Esta función se ha desestimado en Flash 5; se recomienda la utilización del método `String.charCodeAt`.

Véase también
"String.fromCharCode" a pagina 374

mbsubstring

Sintaxis
`mbsubstring(valor, índice, recuento);`

Argumentos
valor La cadena multibyte de la que extraer una nueva cadena multibyte.

índice El número del primer carácter que se va a extraer.

recuento El número de caracteres que se van a incluir en la cadena extra, sin incluir el carácter de índice.

Descripción
Función de cadena; extrae una nueva cadena de caracteres multibyte de una cadena de caracteres multibyte

Reproductor
Flash 4 o posterior. Esta función se ha desestimado en Flash 5; se recomienda la utilización del método `string.substr`.

Véase también
"String.substr" a pagina 376

Mouse (objeto)

Utilice los métodos del objeto Mouse para ocultar y mostrar el cursor en la película. El cursor es visible de modo predeterminado, pero puede ocultar el cursor e implantar un cursor personalizado que cree utilizando un clip de película.

Resumen de los métodos de Mouse

Método	Descripción
`hide`	Oculta el cursor en la película.
`show`	Muestra el cursor en la película.

Mouse.hide

Sintaxis

```
Mouse.hide();
```

Argumentos

Ninguno.

Descripción

Método; oculta el cursor en una película. El cursor es visible de modo predeterminado.

Reproductor

Flash 5 o posterior.

Ejemplo

El código siguiente, anexado a un clip de película en la Línea de tiempo principal, oculta el cursor estándar y establece las posiciones de *x* e *y* de la instancia del clip de película `customCursor` en las posiciones de ratón *x* e *y* en la Línea de tiempo principal:

```
onClipEvent(enterFrame){
   Mouse.hide();
   customCursorMC_x = _root._xmouse;
   customCursorMC_y = _root._ymouse;
}
```

Véase también

"_xmouse" a pagina 418
"_ymouse" a pagina 420
"Mouse.show" a pagina 313

Mouse.show

Sintaxis
```
Mouse.show();
```

Argumentos
Ninguno.

Descripción
Método; hace que el cursor sea visible en una película. El cursor es visible de modo predeterminado.

Reproductor
Flash 5 o posterior.

Véase también
"_xmouse" a pagina 418
"_ymouse" a pagina 420
"Mouse.show" a pagina 313

MovieClip (objeto)

Los métodos del objeto MovieClip proporcionan la misma funcionalidad que las acciones estándar que seleccionan destino para clips de película. También existen métodos adicionales que proporcionan funcionalidad que no está disponible utilizando las acciones estándar que aparecen en la lista de la categoría Acciones del panel Acciones. No necesita utilizar un método constructor para poder llamar a los métodos del objeto MovieClip, en lugar de eso haga referencia a las instancias de clip de película por nombre, utilizando la sintaxis siguiente:

```
anyMovieClip.play();
anyMovieClip.gotoAndPlay(3);
```

Resumen de los métodos del objeto MovieClip

Método	Descripción
`attachMovie`	Anexa una película de la biblioteca.
`duplicateMovieClip`	Duplica el clip de película especificado.
`getBounds`	Devuelve el máximo y el mínimo de las coordenadas *x* e *y* de una película en un espacio de coordenadas especificado.
`getBytesLoaded`	Devuelve el número de bytes cargados para el clip de película especificado.
`getBytesTotal`	Devuelve el tamaño del clip de película en bytes.

Método	Descripción
`getURL`	Recupera un documento de una URL.
`globalToLocal`	Convierte el objeto punto de las coordenadas del Escenario a las coordenadas locales del clip de película especificado.
`gotoAndPlay`	Envía la cabeza lectora a un fotograma específico en el clip de película y reproduce la película.
`gotoAndStop`	Envía la cabeza lectora a un fotograma específico en el clip de película y detiene la película.
`hitTest`	Devuelve `true` si la delimitación del clip de película especificado se cruza con la delimitación del clip de película de destino.
`loadMovie`	Carga la película especificada en el clip de película.
`loadVariables`	Carga variables de una URL u otra ubicación en el clip de película.
`localToGlobal`	Convierte un objeto punto de las coordenadas locales del clip de película a las coordenadas del Escenario global.
`nextFrame`	Envía la cabeza lectora al siguiente clip de película.
`play`	Reproduce el clip de película especificado.
`prevFrame`	Envía la cabeza lectora al fotograma anterior del clip de película.
`removeMovieclip`	Elimina el clip de película de la Línea de tiempo, si se ha creado con una acción o método MovieClip o con el método attachMovie.
`startDrag`	Especifica un clip de película como arrastrable y comienza a arrastrar el clip de película.
`stop`	Detiene la película que se está reproduciendo actualmente.
`stopDrag`	Detiene el arrastrado de cualquier película que se esté arrastrando.
`swapDepths`	Intercambia el nivel de profundidad de la película especificada con la película de un nivel determinado.
`unloadMovie`	Elimina una película cargada con `loadMovie`.

MovieClip.attachMovie

Sintaxis

```
anyMovieClip.attachMovie(id_Nombre, nuevo_nombre, profundidad);
```

Argumentos

id_Nombre El nombre de la película de la biblioteca que se va a anexar. Este es el nombre introducido en el campo Identificador en el cuadro de diálogo Propiedades de vínculos de símbolos.

nuevo_nombre Un nombre de instancia único para el clip de película que se está anexando.

profundidad Un número entero que especifica el nivel de profundidad en el que está la película.

Descripción

Método; crea un nuevo nombre de instancia de una película de la biblioteca y le anexa a la película especificada por *anyMovieClip*. Utilice la acción o método `removeMovieClip` o `unloadMovie` para eliminar una película anexada con `attachMovie`.

Reproductor

Flash 5 o posterior.

Véase también

"removeMovieClip" a pagina 350
"unloadMovie" a pagina 384
"MovieClip.removeMovieClip" a pagina 325
"MovieClip.unloadMovie" a pagina 327

MovieClip.duplicateMovieClip

Sintaxis

```
anyMovieClip.duplicateMovieClip(nuevo_nombre, profundidad);
```

Argumentos

nuevo_nombre Un identificador único para el clip de película duplicado.

profundidad Un número que especifica el nivel de profundidad donde se va a colocar la película especificada.

Descripción
Método; crea una instancia del clip de película especificado mientras se reproduce la película. Los clips de película duplicados siempre comienza en el fotograma 1, sin tener en cuenta en qué fotograma está el clip de película original cuando se llama al método `duplicateMovieClip`. Las variables del clip de película principal no se copian en el clip de película duplicado. Si se elimina el clip de película principal también se elimina el clip de película duplicado. Los clips de película agregados con `duplicateMovieClip` pueden borrarse con la acción o método `removeMovieClip`.

Reproductor
Flash 5 o posterior.

Véase también
"removeMovieClip" a pagina 350
"MovieClip.removeMovieClip" a pagina 325

MovieClip.getBounds

Sintaxis
anyMovieClip.getBounds(*espacio_coord_destino*);

Argumentos
espacio_coord_destino La ruta de destino de la Línea de tiempo cuyo espacio de coordenadas desea utilizar como punto de referencia.

Descripción
Método; devuelve los valores de coordenadas máximo y mínimo de *x* e *y* del Clip de película, para el espacio de coordenadas de destino especificado en el argumento. El objeto devuelto tendrá las propiedades `{xMin, xMax, yMin, yMax}`. Utilice los métodos `localToGlobal` y `globalToLocal` del objeto MovieClip para convertir las coordenadas locales del clip de película en las coordenadas del Escenario, o para convertir las coordenadas del Escenario en las coordenadas locales.

Reproductor
Flash 5 o posterior.

Ejemplo
El ejemplo siguiente utiliza `getBounds` para recuperar la delimitación de la instancia `myMovieClip` en el espacio de coordenadas de la película principal.

```
myMovieClip.getBounds(._root);
```

Véase también
"MovieClip.globalToLocal" a pagina 318
"MovieClip.localToGlobal" a pagina 323

MovieClip.getBytesLoaded

Sintaxis
anyMovieClip.getBytesLoaded();

Argumentos
Ninguno.

Descripción
Método; devuelve el número de bytes cargados (en flujo) para el objeto de clip de película especificado. Como los clips de película internos se cargan automáticamente, el resultado devuelto por este método y `MovieClip.getBytesTotal` será el mismo si el objeto de clip de película especificado hace referencia a un clip de película interno. Este método está pensado para su utilización en películas cargadas. Puede comparar el valor de `getBytesLoaded` con el valor de `getBytesTotal` para determinar el porcentaje que se ha cargado de una película externa.

Reproductor
Flash 5 o posterior.

MovieClip.getBytesTotal

Sintaxis
anyMovieClip.getBytesTotal();

Argumentos
Ninguno.

Descripción
Método; devuelve el tamaño, en bytes, del objeto de clip de película especificado. Para clips de película que son externos (la película raíz o un clip de película que se está cargando en un destino o en un nivel) el valor devuelto es el tamaño del archivo SWF.

Reproductor
Flash 5 o posterior.

MovieClip.getURL

Sintaxis

```
anyMovieClip.getURL(URL [,ventana, variables]]);
```

Argumentos

URL La URL de la cual obtener el documento.

ventana Un argumento opcional que especifica el nombre, el fotograma o la expresión que especifica la ventana o fotograma HTML en donde se carga el documento. También puede utilizar los siguientes nombres de destino reservados: `_self` especifica el fotograma actual en la ventana actual, `_blank` especifica una nueva ventana, `_parent` especifica el principal del fotograma actual, `_top` especifica el fotograma de nivel superior en la ventana actual.

variables Un argumento opcional que especifica un método para enviar variables asociadas con la película que se va a cargar. Si no hay variables, omita este argumento; en caso contrario, especifique si se cargan las variables utilizando un método `GET` o `POST`. `GET` anexa las variables al final de la URL y se utiliza para un número pequeño de variables. `POST` envía las variables en un encabezado HTTP aparte y se utiliza para cadenas largas de variables.

Descripción

Método; carga un documento de la URL especificada en la ventana especificada. El método `getURL` también puede utilizarse para pasar variables a otra aplicación definida en la URL utilizando un método `GET` o `POST`.

Reproductor

Flash 5 o posterior.

MovieClip.globalToLocal

Sintaxis

```
anyMovieClip.globalToLocal(point);
```

Argumentos

point El nombre del identificador de un objeto creado con el objeto Object genérico especificando las coordenadas *x* e *y* como propiedades.

Descripción

Método; convierte el objeto *point* de las coordenadas del Escenario (globales) a las coordenadas del clip de película (locales).

Reproductor

Flash 5 o posterior.

Ejemplo
El ejemplo siguiente convierte las coordenadas globales de *x* e *y* del objeto *point* a las coordenadas locales del clip de película.

```
onClipEvent(mouseMove) {
point = new object();
point.x = _root._xmouse;
point.y = _root._ymouse;
globalToLocal(point);
_root.out = _xmouse + " === " + _ymouse;
_root.out2 = point.x + " === " + point.y;
updateAfterEvent();
}
```

Véase también
"MovieClip.localToGlobal" a pagina 323
"MovieClip.getBounds" a pagina 316

MovieClip.gotoAndPlay

Sintaxis
anyMovieClip.gotoAndPlay(*fotograma*);

Argumentos
fotograma El número de fotograma al que se enviará la cabeza lectora.

Descripción
Método; comienza a reproducir la película en el fotograma especificado.

Reproductor
Flash 5 o posterior.

MovieClip.gotoAndStop

Sintaxis
anyMovieClip.gotoAndStop(*fotograma*);

Argumentos
fotograma El número de fotograma al que se enviará la cabeza lectora.

Descripción
Método; detiene la reproducción de la película en el fotograma especificado.

Reproductor
Flash 5 o posterior.

MovieClip.hitTest

Sintaxis

```
anyMovieClip.hitTest(x, y, indicador_forma);
anyMovieClip.hitTest(destino);
```

Argumentos

x La coordenada *x* de la zona activa en el Escenario.

y La coordenada *y* de la zona activa en el Escenario.

Las coordenadas *x* e *y* se definen en el espacio de coordenadas global.

destino La ruta de destino de la zona activa que puede cruzarse o solaparse con la instancia especificada por *anyMovieClip*. *destino* normalmente representa un botón o un campo de introducción de texto.

indicador_forma Un valor Booleano que especifica si se evalúa toda la forma de la instancia especificada (*true*), o solamente la delimitación (*false*). Este argumento sólo puede especificarse si la zona activa se identifica utilizando los argumentos de coordenadas *x* e *y*.

Descripción

Método; evalúa la instancia especificada por *anyMovieClip* para ver si se solapa o cruza con la zona activa identificada por los argumentos de coordenadas *destino* o *x* e *y*.

La Sintaxis 1 compara las coordenadas *x* e *y* con la forma o la delimitación de la instancia especificada, según la configuración de *indicador_forma*. Si *indicador_forma* está establecido en `true`, solamente el área ocupada actualmente por la instancia en el Escenario se evalúa y si *x* e *y* se solapan en algún punto, se devuelve un valor de `true`. Esto es muy útil para determinar si el clip de película se encuentra dentro de la zona activa o punto caliente especificado.

La Utilización 2 evalúa las delimitaciones de *destino* y de la instancia especificada y devuelve `true` si se solapan o cruzan en cualquier punto.

Reproductor

Flash 5 o posterior.

Ejemplo

El ejemplo siguiente utiliza `hitTest` con las propiedades `x_mouse` y `y_mouse` para determinar si el ratón se encuentra sobre la delimitación del destino.

```
if (hitTest( _root._xmouse, _root._ymouse, false));
```

El ejemplo siguiente utiliza `hitTest` para determinar si el clip de película `ball` se solapa o cruza con el clip de película `square`.

```
if(_root.ball, hittest(_root.square)){
trace("ball intersects square");
}
```

Véase también

"MovieClip.localToGlobal" a pagina 323
"MovieClip.globalToLocal" a pagina 318
"MovieClip.getBounds" a pagina 316

MovieClip.loadMovie

Sintaxis

anyMovieClip.loadMovie(*url* [,*variables*]);

Argumentos

url Una URL absoluta o relativa para el archivo SWF que se va a cargar.

Una ruta relativa debe ser relativa respecto al archivo SWF. La URL debe estar en el mismo subdominio que la URL donde reside actualmente la película. Para utilizarlos en Flash Player o para realizar pruebas en el modo de prueba de película en el entorno de creación de Flash, todos los archivos SWF tienen que guardarse en la misma carpeta y los nombres de archivo no pueden incluir especificaciones de carpeta ni de unidad de disco.

variables Un argumento opcional que especifica un método para enviar variables asociadas con la película que se va a cargar. El argumento debe ser la cadena "GET" o "POST". Si no hay variables, omita este argumento; en caso contrario, especifique si se cargan las variables utilizando un método `GET` o `POST`. `GET` anexa las variables al final de la URL y se utiliza para un número pequeño de variables. `POST` envía las variables en un encabezado HTTP aparte y se utiliza para cadenas largas de variables.

Descripción
Método; reproduce películas adicionales sin cerrar Flash Player. Por lo general, Flash Player muestra una sola película (archivo SWF) y después se cierra. El método `loadMovie` permite ver varias películas al mismo tiempo o cambiar entre películas sin cargar otro documento HTML.

Utilice la acción `unloadMovie` para eliminar películas cargadas con la acción `loadMovie`.

Utilice el método `loadVariables` para mantener la película activa y actualizar las variables con nuevos valores.

Reproductor
Flash 5 o posterior.

Véase también
"MovieClip.loadVariables" a pagina 322
"MovieClip.unloadMovie" a pagina 327

MovieClip.loadVariables

Sintaxis
anyMovieClip.loadVariables(*url, variables*);

Argumentos
url La URL absoluta o relativa para el archivo externo. El anfitrión de la URL debe estar en el mismo subdominio que el clip de película.

variables El método para recuperar las variables. `GET` anexa las variables al final de la URL y se utiliza para un número pequeño de variables. `POST` envía las variables en un encabezado HTTP distinto y se utiliza para cadenas largas de variables.

Descripción
Método; lee los datos de un archivo externo y establece los valores para variables en una película o en un clip de película. El archivo externo puede ser un archivo de texto generado por un script CGI, Páginas de Active Server (ASP) o PHP y puede contener cualquier número de variables.

Este método también puede utilizarse para actualizar variables en la película activa con nuevos valores.

Este método requiere que el texto de la URL esté en formato MIME estándar: *application/x-www-urlformencoded* (formato de script CGI).

Reproductor
Flash 5 o posterior.

Véase también
"MovieClip.loadMovie" a pagina 321

MovieClip.localToGlobal

Sintaxis

anyMovieClip.localToGlobal(*point*);

Argumentos

point El nombre del identificador de un objeto creado con el objeto Object, especificando las coordenadas *x* e *y* como propiedades.

Descripción

Método; convierte el objeto *point* de las coordenadas del clip de película (locales) a las coordenadas del Escenario (globales).

Reproductor

Flash 5 o posterior.

Ejemplo

El ejemplo siguiente convierte las coordenadas de *x* e *y* del objeto *point*, de las coordenadas del clip de película (locales) a las coordenadas del Escenario (globales). Las coordenadas locales *x* e *y* se especifican utilizando `xmouse` y `ymouse` para recuperar las coordenadas *x* e *y* de la posición del ratón.

```
onClipEvent(mouseMove) {
point = new object();
point.x = _xmouse;
point.y = _ymouse;
_root.out3 = point.x + " === " + point.y;
_root.out = _root._xmouse + " === " + _root._ymouse;
localToGlobal(point);
_root.out2 = point.x + " === " + point.y;
updateAfterEvent();
}
```

Véase también

"MovieClip.globalToLocal" a pagina 318

MovieClip.nextFrame

Sintaxis
anyMovieClip.nextFrame();

Argumentos
Ninguno.

Descripción
Método; envía la cabeza lectora al fotograma siguiente del clip de película.

Reproductor
Flash 5 o posterior.

MovieClip.play

Sintaxis
anyMovieClip.play();

Argumentos
Ninguno.

Descripción
Método; reproduce el clip de película.

Reproductor
Flash 5 o posterior.

MovieClip.prevFrame

Sintaxis
anyMovieClip.prevFrame();

Argumentos
Ninguno.

Descripción
Método; envía la cabeza lectora al fotograma anterior y la detiene.

Reproductor
Flash 5 o posterior.

MovieClip.removeMovieClip

Sintaxis
```
anyMovieClip.removeMovieClip();
```

Argumentos
Ninguno.

Descripción
Método; elimina una instancia de clip de película creada con la acción `duplicateMovieclip` o los métodos `duplicateMovieClip` o `attachMovie` del objeto MovieClip.

Reproductor
Flash 5 o posterior.

Véase también
"MovieClip.loadMovie" a pagina 321
"MovieClip.attachMovie" a pagina 315

MovieClip.startDrag

Sintaxis
```
anyMovieClip.startDrag([bloqueado, izquierda, derecha,
arriba, abajo]);
```

Argumentos
bloqueado Un valor Booleano que especifica si el clip de película arrastrable está bloqueado en el centro de la posición del ratón (*true*), o bloqueado en el punto en el que el usuario hizo clic por primera vez en el clip de película (*false*). Este argumento es opcional.

izquierda, arriba, derecha, abajo Los valores relativos a las coordenadas del principal del clip de película que especifica un rectángulo de limitación para el clip de película. Estos argumentos son opcionales.

Descripción
Método; permite al usuario arrastrar el clip de película especificado. Un clip de película puede arrastrarse hasta que se detiene explícitamente por la llamada del método `stopDrag` o hasta que otro clip de película se convierte en arrastrable. Sólo un clip de película puede arrastrarse al mismo tiempo.

Reproductor
Flash 5 o posterior.

Véase también
"MovieClip.stopDrag" a pagina 326
"_droptarget" a pagina 263

MovieClip.stop

Sintaxis
anyMovieClip.stop();

Argumentos
Ninguno.

Descripción
Método; detiene el clip de película que se está reproduciendo actualmente.

Reproductor
Flash 5 o posterior.

MovieClip.stopDrag

Sintaxis
anyMovieClip.stopDrag();

Argumentos
Ninguno.

Descripción
Método; finaliza una acción de arrastre implantada con el método `startDrag`. Una película permanece arrastrable hasta que se agrega un método`stopDrag` o hasta que otra película se convierte en arrastrable. Sólo puede arrastrarse un clip de película cada vez.

Reproductor
Flash 5 o posterior.

Véase también
"_droptarget" a pagina 263
"MovieClip.startDrag" a pagina 325

MovieClip.swapDepths

Sintaxis
anyMovieClip.swapDepths(*profundidad*);

anyMovieClip.swapDepths(*destino*);

Argumentos
destino La instancia del clip de película cuya profundidad se intercambia con la instancia especificada en *anyMovieClip*. Ambas instancias deben tener el mismo clip de película principal.

profundidad Un número que especifica el nivel de profundidad donde se va a colocar *anyMovieClip*.

Descripción
Método; intercambia el apilado, o *z*, orden (nivel de profundidad) de la instancia especificada con la película especificada por el argumento *destino* o con la película que actualmente ocupa el nivel *profundidad* especificado en el argumento. Ambas películas deben tener el mismo clip de película principal. Intercambiar el nivel de profundidad de los clips de película tiene el efecto de mover una película frente o detrás de la otra. Si la película se está interpolando cuando se llama a este método, se detiene la interpolación.

Reproductor
Flash 5 o posterior.

Véase también
"_level" a pagina 293

MovieClip.unloadMovie

Sintaxis
```
anyMovieClip.unloadMovie();
```

Argumentos
Ninguno.

Descripción
Método; elimina un clip de película cargado con los métodos `loadMovie` o `attachMovie` de MovieClip.

Reproductor
Flash 5 o posterior.

Véase también
"MovieClip.loadMovie" a pagina 321
"MovieClip.attachMovie" a pagina 315

_name

Sintaxis
```
nombre_instancia._name
nombre_instancia._name = valor;
```

Argumentos
nombre_instancia Un nombre de instancia de un clip de película para el que se va a establecer o recuperar la propiedad `_name`.

valor Una cadena que especifica un nuevo nombre de instancia.

Descripción
Propiedad; especifica el nombre de instancia de clip de película.

Reproductor
Flash 4 o posterior.

NaN

Sintaxis
`NaN`

Argumentos
Ninguno.

Descripción
Variable; una variable predefinida con valor IEEE-754 para NaN (No un Número, Not a Number).

Reproductor
Flash 5 o posterior.

ne (no igual, específico de cadena)

Sintaxis
expresión1 `ne` *expresión2*

Argumentos
expresión1, expresión2 Números, cadenas o variables.

Descripción
Operador (de comparación); compara *expresión1* con *expresión2* y devuelve `true` si *expresión1* no es igual que *expresión2*; en caso contrario, devuelve `false`.

Reproductor
Flash 4 o posterior. Este operador se ha desestimado en Flash 5; se recomienda la utilización del nuevo operador `!=` (no igual).

Véase también
"`!= (no igualdad)` a pagina 193

new

Sintaxis
`new` *constructor*`();`

Argumentos
constructor Una función seguida de cualquiera de los argumentos opcionales de los paréntesis. La función es habitualmente el nombre del tipo de objeto (por ejemplo, Matriz, Matemáticas, Número, Objeto) que se va a construir.

Descripción

Operador; crea un nuevo objeto, inicialmente anónimo, llama a la función identificada por el argumento *constructor*, pasa cualquiera de los argumentos de los paréntesis y pasa el objeto recién creado como un valor de la palabra clave `this`. La función constructor entonces puede utilizar `this` para instanciar el nuevo objeto.

La propiedad _prototype_ del objeto de la función constructor se copia en la propiedad _proto_ del nuevo objeto. Como resultado, el nuevo objeto soporta todos los métodos y propiedades especificadas en la función constructor del objeto prototipo.

Reproductor

Flash 5 o posterior.

Ejemplo

El ejemplo siguiente crea los objetos `book1` y `book2` utilizando el nuevo operador.

```
function Book(name, price)
{
  this.name = name;
  this.price = price;
}
book1 = new Book("Confederacy of Dunces", 19.95);
book2 = new Book("The Floating Opera", 10.95);
```

Véase también

"[] (operador de acceso a matriz)" a pagina 205
"{} (inicializador de objeto)" a pagina 207
La sección del método constructor dentro de una entrada de objeto.

newline

Sintaxis

```
newline;
```

Argumentos

Ninguno.

Descripción

Constante; inserta un carácter de retorno de carro ({) insertando una línea vacía en el código de ActionScript. Utilice `newline` para crear espacio para la información que se recupera por una función o una acción del código.

Reproductor

Flash 4 o posterior.

nextFrame

Sintaxis
```
nextFrame();
```

Argumentos
Ninguno.

Descripción
Acción; envía la cabeza lectora al fotograma siguiente y la detiene.

Reproductor
Flash 2 o posterior.

Ejemplo
Cuando el usuario hace clic sobre un botón la acción `nextFrame` se asigna, se envía la cabeza lectora al fotograma siguiente.

```
on (release) {
    nextFrame(5);
}
```

nextScene

Sintaxis
```
nextScene();
```

Argumentos
Ninguno.

Descripción
Acción; envía la cabeza lectora al fotograma 1 de la escena siguiente y la detiene.

Reproductor
Flash 2 o posterior.

Ejemplo
Esta acción está asignada a un botón que, cuando se presiona y se suelta, envía la cabeza lectora al fotograma 1 de la escena siguiente.

```
on(release) {
    nextScene();
}
```

not

Sintaxis
`not` *`expresión`*

Argumentos
`expresión` Cualquier variable o expresión que convierte a un valor Booleano.

Descripción
Operador; realiza una operación NOT lógica en Flash Player 4.

Reproductor
Flash 4 o posterior. Este operador se ha desestimado en Flash 5; se recomienda la utilización del nuevo operador `!` (NOT lógico).

Véase también
"`! (valor NOT lógico)`" a pagina 193

null

Sintaxis
`null`

Argumentos
Ninguno.

Descripción
Palabra clave; un valor especial que puede asignarse a variables o puede ser devuelto por una función si no se han proporcionado datos. Puede utilizar `null` para representar valores que faltan o que no tienen un tipo de datos definido.

Reproductor
Flash 5 o posterior.

Ejemplo
En un contexto numérico `null` evalúa como 0. Se pueden realizar comprobaciones de igualdad con `null`. En esta sentencia, un nodo de árbol binario no tiene nivel secundario a la izquierda, de modo que el campo de su nivel secundario a la izquierda podría establecerse como `null`.

```
if (tree.left == null) {
  tree.left = new TreeNode();
}
```

Number (función)

Sintaxis

```
Number(expresión);
```

Argumentos

expresión La cadena, Booleano u otra expresión que se va a convertir en un número.

Descripción

Función; convierte el argumento *x* en un número y devuelve un valor como se muestra a continuación:

Si *x* es un número, el valor devuelto es *x*.

Si *x* es un Booleano, el valor devuelto es 1 si *x* es `true`, 0 si *x* es `false`.

Si *x* es una cadena, la función intenta analizar *x* como un número decimal con un exponente final opcional, es decir, 1.57505e-3.

Si *x* no está definida, el valor devuelto es 0.

Esta función se utiliza para convertir los archivos de Flash 4 que contienen operadores desestimados que se importan al entorno de creación de Flash 5. Consulte el operador `&` para obtener más información.

Reproductor

Flash 4 o posterior.

Véase también

"Number (objeto)" a pagina 332

Number (objeto)

El objeto Number es un objeto envolvente sencillo para el tipo de datos número, lo que quiere decir que puede manipular valores numéricos primitivos utilizando los métodos y propiedades asociadas con el objeto Number. La funcionalidad proporcionada por este objeto es idéntica a la del objeto Number de JavaScript.

Debe utilizar el constructor `Number` cuando llama a los métodos del objeto Number, pero no necesita utilizar el constructor cuando llama a las propiedades del objeto Number. El ejemplo siguiente especifica la sintaxis para llamar a los métodos y propiedades del objeto Number.

A continuación se muestra un ejemplo de llamada al método `toString` del objeto Number:

```
myNumber = new Number(1234);
myNumber.toString();
```

Devuelve una cadena que contiene la representación binaria del número 1234.

A continuación se muestra un ejemplo de llamada a la propiedad `MIN_VALUE` (también denominada constante) del objeto Number:

```
smallest = Number.MIN_VALUE
```

Resumen de los métodos del objeto Number

Método	Descripción
`toString`	Devuelve la representación de cadena de un objeto Number.
`valueOf`	Devuelve el valor primitivo de un objeto Number.

Resumen de las propiedades de un objeto Number

Propiedad	Descripción
`MAX_VALUE`	Constante que representa el número mayor que se puede representar (IEEE-754 de doble precisión). Este número es aproximadamente 1.7976931348623158e+308.
`MIN_VALUE`	Constante que representa el número menor que se puede representar (IEEE-754 de doble precisión). Este número es aproximadamente 5e-324.
`NaN`	Constante que representa el valor de Not a Number (NaN).
`NEGATIVE_INFINITY`	Constante que representa el valor de infinito negativo.
`POSITIVE_INFINITY`	Constante que representa el valor de infinito positivo. Este valor es el mismo que la variable global Infinity.

Constructor del objeto Number.

Sintaxis

```
myNumber = new Number(valor);
```

Argumentos

`valor` El valor numérico del objeto Number que se está creando o un valor que se va a convertir en un número.

Descripción

Constructor; crea un nuevo objeto Number. Debe utilizar el constructor `Number` cuando utilice los métodos `toString` y `valueOf` del objeto Number. No se usa un constructor cuando se utilizan la propiedades del objeto Number. El constructor `new Number` se utiliza fundamentalmente como marcador de lugar. Una instancia del objeto Number no es lo mismo que la función `Number` que convierte un argumento en un valor primitivo.

Reproductor
Flash 5 o posterior.

Ejemplo
El código siguiente construye nuevos objetos Número.

```
n1 = new Number(3.4);
n2 = new Number(-10);
```

Véase también
"Number (función)" a pagina 332

Number.MAX_VALUE

Sintaxis
```
Number.MAX_VALUE
```

Argumentos
Ninguno.

Descripción
Propiedad; el número mayor que se puede representar (IEEE-754 de doble precisión). Este número es aproximadamente 1.79E+308.

Reproductor
Flash 5 o posterior.

Number.MIN_VALUE

Sintaxis
```
Number.MIN_VALUE
```

Argumentos
Ninguno.

Descripción
Propiedad; el número menor que se puede representar (IEEE-754 de doble precisión). Este número es aproximadamente 5e-324.

Reproductor
Flash 5 o posterior.

Number.NaN

Sintaxis
`Number.NaN`

Argumentos
Ninguno.

Descripción
Propiedad; el valor IEEE-754 que representa Not A Number (`NaN`).

Reproductor
Flash 5 o posterior.

Number.NEGATIVE_INFINITY

Sintaxis
`Number.NEGATIVE_INFINITY`

Argumentos
Ninguno.

Descripción
Propiedad; devuelve el valor IEEE-754 que representa infinito negativo. Este valor es el mismo que la variable global `Infinity`.

El infinito negativo es un valor numérico especial que se devuelve cuando una operación o función matemática devuelve un valor negativo mayor de lo que se puede representar.

Reproductor
Flash 5 o posterior.

Number.POSITIVE_INFINITY

Sintaxis
`Number.POSITIVE_INFINITY`

Argumentos
Ninguno.

Descripción
Propiedad; devuelve el valor IEEE-754 que representa infinito positivo. Este valor es el mismo que la variable global `Infinity`

El infinito positivo es un valor numérico especial que se devuelve cuando una operación o función matemática devuelve un valor mayor de lo que se puede representar.

Reproductor
Flash 5 o posterior.

Number.toString

Sintaxis
myNumber.toString(*raíz*);

Argumentos
raíz Especifica la base numérica (de 2 a 36) que se utiliza en la conversión de número a cadena. Si no especifica el argumento *raíz*, el valor predeterminado es 10.

Descripción
Método; devuelve la representación de la cadena del objeto Number especificado (*myNumber*).

Reproductor
Flash 5 o posterior.

Ejemplo
El ejemplo siguiente utiliza el método Number.toString, especificando 2 para el argumento *raíz*:

```
myNumber = new Number (1000);
(1000).toString(2);
```

Devuelve una cadena que contiene la representación binaria del número 1000.

Number.valueOf

Sintaxis
myNumber.valueOf();

Argumentos
Ninguno.

Descripción
Método; devuelve el tipo de valores primitivo del objeto Number especificado y convierte el objeto envolvente Número al tipo de valor primitivo.

Reproductor
Flash 5 o posterior.

Object (objeto)

El objeto Object genérico se encuentra en la raíz de la jerarquía de clases de ActionScript. La funcionalidad del objeto Object genérico es un pequeño subconjunto de la proporcionada por el objeto Object de JavaScript.

El objeto Object genérico requiere Flash Player 5.

Resumen de los métodos del objeto Object

Método	Descripción
`toString`	Convierte el objeto especificado en una cadena y la devuelve.
`valueOf`	Devuelve el valor primitivo de un objeto Object.

Constructor del objeto Object

Sintaxis

```
new Object();
```

```
new Object(valor);
```

Argumentos

`valor` Un número, Booleano o cadena que se va a convertir en un objeto. Este argumento es opcional. Si no especifica *`valor`*, el constructor crea un nuevo objeto sin propiedades definidas.

Descripción

Constructor; crea un nuevo objeto Object.

Reproductor

Flash 5 o posterior.

Véase también

"Sound.setTransform" a pagina 362
"Color.setTransform" a pagina 240

Object.toString

Sintaxis
```
myObject.toString();
```

Argumentos
Ninguno.

Descripción
Método; convierte el objeto especificado en una cadena y la devuelve.

Reproductor
Flash 5 o posterior.

Object.valueOf

Sintaxis
```
myObject.valueOf();
```

Argumentos
Ninguno.

Descripción
Método; devuelve el valor primitivo del objeto especificado. Si el objeto no tiene un valor primitivo, se devuelve el propio objeto.

Reproductor
Flash 5 o posterior.

onClipEvent

Sintaxis
```
onClipEvent(movieEvent);{
...
}
```

Argumentos
Una *movieEvent* es un evento desencadenante que ejecuta acciones que están asignadas a una instancia de clip de película. Cualquiera de los valores siguientes pueden especificarse para el argumento *movieEvent*.

- `load` La acción se inicia en cuanto se instancia el clip de película y aparece en la Línea de tiempo.
- `unload` La acción se inicia en el primer fotograma después de que se ha eliminado el clip de película de la Línea de tiempo. Estas acciones están asociadas con el evento de clip de película `Unload` se procesan antes que cualquiera de las acciones anexadas al fotograma afectado.

- `enterFrame` La acción se inicia según se reproduce cada fotograma, es similar a las acciones anexadas a un clip de película. Estas acciones están asociadas con el evento de clip de película `OnEnterFrame` se procesan después que cualquiera de las acciones anexadas a los fotogramas afectados.
- `mouseMove` La acción se inicia cada vez que se mueve el ratón. Utilice las propiedades `_xmouse` y `_ymouse` para determinar la posición actual del ratón.
- `mouseDown` La acción se inicia cada vez que se presiona el botón izquierdo del ratón.
- `mouseUp` La acción se inicia cada vez que se suelta el botón izquierdo del ratón.
- `keyDown` La acción se inicia cuando se presiona una tecla. Utilice el método `Key.getCode` para recuperar información sobre la última tecla presionada.
- `keyUp` La acción se inicia cuando se suelta una tecla. Utilice el método `Key.getCode` para recuperar información sobre la última tecla presionada.
- `data` La acción se inicia cuando se reciben datos en una acción `loadVariables` o `loadMovie`. Cuando se especifica con una acción `loadVariables`, el evento `data` se produce una sola vez, cuando se carga la última variable. Cuando se especifica con una acción `loadMovie`, el evento `data` se produce repetidamente, según se recupera cada sección de datos.

Descripción

Controlador; desencadena acciones para una instancia específica de un clip de película.

Reproductor

Flash 5 o posterior.

Ejemplo

La sentencia siguiente incluye el script de un archivo externo cuando se carga la instancia del clip de película y aparece por primera vez en la Línea de tiempo.

```
onClipEvent(load) {
    #include "myScript.as"
}
```

El ejemplo siguiente utiliza `onClipEvent` con el evento de película `keyDown`. El evento de película `keyDown` se utiliza habitualmente junto con uno o más métodos y propiedades asociadas con el objeto Key. En el script siguiente, `key.getCode` se utiliza para averiguar que tecla ha presionado el usuario, el valor devuelto se asocia con las propiedades `RIGHT` o `LEFT` del objeto Key y la película se dirige en consecuencia.

```
onClipEvent(keyDown) {

    if (Key.getCode() == Key.RIGHT) {

      } _parent.nextFrame();

    else if (Key.getCode() == Key.LEFT){

       _parent.prevFrame();

}
```

El ejemplo siguiente utiliza `onClipEvent` con el evento de película `mouseMove`. Las propiedades `xmouse` y `ymouse` realizan un seguimiento de la posición del ratón.

```
onClipEvent(mouseMove) {

stageX=_root.xmouse;

stageY=_root.ymouse;

}
```

Véase también

"on(mouseEvent)" a pagina 340
"Key (objeto)" a pagina 283
"_xmouse" a pagina 418
"_ymouse" a pagina 420

on(mouseEvent)

Sintaxis

```
on(mouseEvent) {
sentencia;
}
```

Argumentos

`sentencia` Las instrucciones que se van a ejecutar cuando tiene lugar mouseEvent.

Una acción *`mouseEvent`* puede tener uno de los siguientes argumentos:

- `press` Se ha presionado el botón del ratón mientras el puntero se encuentra sobre el botón.
- `release` Se ha soltado el botón del ratón mientras el puntero se encuentra sobre el botón.

- `releaseOutside` Se ha soltado el botón del ratón mientras el puntero se encuentra fuera del botón.
- `rollOver` El puntero del ratón se sitúa sobre el botón.
- `rollOut` El puntero se sitúa sobre el área del botón.
- `dragOver` Mientras el puntero se encuentra sobre el botón se ha presionado el botón del ratón, se ha situado fuera del botón y a vuelto a situarse sobre el botón.
- `dragOut` Mientras el puntero se encuentra sobre el botón se ha presionado el botón del ratón y después se sitúa fuera del área del botón.
- `keyPress (“tecla”)` La *`tecla`* especificada se ha presionado. La parte *`tecla`* del argumento se especifica utilizando cualquiera de los códigos de tecla que aparecen en la lista del Apéndice B, “Teclas del teclado y Valores de código de tecla” o cualquiera de las constantes de tecla que aparecen en la lista de “Resumen de las propiedades de un objeto Array” a pagina 226.

Descripción

Controlador; especifica el evento de ratón o las presiones de tecla que desencadenan una acción.

Reproductor

Flash 2 o posterior.

Ejemplo

En el script siguiente, la acción `startDrag` se ejecuta cuando se presiona el ratón y el script condicional se ejecuta cuando se suelta el ratón y se suelta el objeto.

```
on(press) {
    startDrag("rabbit");
}
on(release) {
    if(getproperty("", _droptarget) == target) {
        setProperty ("rabbit", _x, _root.rabbit_x);
        setProperty ("rabbit", _y, _root.rabbit_y);
    } else {
_root.rabbit_x = getProperty("rabbit", _x);
_root.rabbit_y = getProperty("rabbit", _y);
_root.target = "pasture";
    }
    trace(_root.rabbit_y);
    trace(_root.rabbit_x);
    stopDrag();
}
```

Véase también

“Key (objeto)” a pagina 283
“onClipEvent” a pagina 338

or

Sintaxis
condición1 or *condición2*

Argumentos
condición1, 2 Una expresión que evalúa como `true` o `false`.

Descripción
Operador; evalúa *condición1* y *condición2* y si cualquiera de las expresiones es `true`, entonces toda la expresión es `true`.

Reproductor
Flash 4 o posterior. Este operador se ha desestimado en Flash 5 y se recomienda a los usuarios que utilicen el nuevo operador `||` .

Véase también
"|| (operador OR)" a pagina 209

ord

Sintaxis
ord(*carácter*);

Argumentos
carácter El carácter que se convierte en un número de código ASCII.

Descripción
Función de cadena; convierte los caracteres en números de código ASCII.

Reproductor
Flash 4 o posterior. Esta función se ha desestimado en Flash 5, se recomienda que en su lugar utilice los métodos y propiedades del objeto `String`.

Véase también
"Cadena (objeto)" a pagina 371

_parent

Sintaxis
_parent.*propiedad* = *x*
_parent._parent.*propiedad* = *x*

Argumentos
propiedad La propiedad que se especifica para el clip de película actual y para el principal.

x El valor establecido para la propiedad. Este es un argumento opcional y puede que no necesite establecerlo en la propiedad.

Descripción

Propiedad; especifica o devuelve una referencia al clip de película que contiene el clip de película actual. El clip de película actual es el clip de película que contiene los scripts que se están ejecutando actualmente. Utilice `_parent` para especificar una ruta de acceso relativa.

Reproductor

Flash 4 o posterior.

Ejemplo

En el ejemplo siguiente el clip de película `desk` es un elemento secundario del clip de película `classroom`. Cuando el script siguiente se ejecuta dentro del clip de película `desk`, la cabeza lectora saltará al fotograma 10 de la Línea de tiempo del clip de película `classroom`.

```
_parent.gotoAndStop(10);
```

Véase también

"_root" a pagina 351
"targetPath" a pagina 379

parseFloat

Sintaxis

```
parseFloat(cadena);
```

Argumentos

`cadena` La cadena que se va a analizar para convertirla en un número con coma flotante.

Descripción

Función; convierte una cadena en un número con coma flotante. La función analiza y devuelve los números de la cadena, hasta que el analizador llega a un carácter que no es parte del número inicial. Si la cadena no comienza con un número que puede analizarse, `parseFloat` devuelve `NaN` o 0. El espacio en blanco que precede a un número entero válido se ignora, al igual que los caracteres no numéricos finales.

Reproductor

Flash 5 o posterior.

Ejemplo

A continuación se muestran ejemplos de la utilización de `parseFloat` para evaluar varios tipos de números:

`parseFloat("-2")` devuelve `-2`

`parseFloat("2.5")` devuelve `2.5`

`parseFloat("3.5e6")` devuelve `3.5e6`, o `3500000`

`parseFloat("foobar")` devuelve `NaN`

parseInt

Sintaxis

```
parseInt(expresión, raíz);
```

Argumentos

expresión La cadena, número con coma flotante u otra expresión que se va a analizar y convertir en un número entero.

raíz Un número entero que representa la raíz (base) del número que se va a analizar. Los valores legales van de 2 a 36. Este argumento es opcional.

Descripción

Función; convierte una cadena en un número entero. Si la cadena especificada en el argumento no puede convertirse en un número, la función devuelve `NaN` o 0. Los números enteros que comienzan por 0 o que especifican una raíz de 8 se interpretan como números octales. Los números enteros que comienzan por 0x se interpretan como números hexadecimales. El espacio en blanco que precede a los números enteros válidos se ignora, al igual que los caracteres no numéricos finales.

Reproductor

Flash 5 o posterior.

Ejemplo

A continuación se muestran ejemplos de la utilización de `parseInt` para evaluar varios tipos de números.

`parseInt("3.5")` devuelve `3.5`

`parseInt("bar")` devuelve `NaN`

`parseInt("4foo")` devuelve `4`

Ejemplo

Conversión hexadecimal:

`parseInt("0x3F8")` devuelve `1016`

`parseInt("3E8", 16)` devuelve `1000`

Conversión binaria:

`parseInt("1010", 2)` devuelve `10` (la representación decimal del binario 1010)

Análisis de números octales (en este caso el número octal se identifica por la raíz, 8):

`parseInt("777", 8)` devuelve `511` (la representación decimal del octal 777)

play

Sintaxis
```
play();
```

Argumentos
Ninguno.

Descripción
Acción; desplaza la cabeza lectora hacia delante en la Línea de tiempo.

Reproductor
Flash 2 o posterior.

Ejemplo
El código siguiente utiliza la sentencia `if` para comprobar el valor de un nombre que introduce el usuario. Si el usuario introduce `Steve`, se llama a la acción `play` y la cabeza lectora avanza en la Línea de tiempo. Si el usuario introduce cualquier cosa diferente de `Steve`, la película no se reproduce y aparece un campo de texto con el nombre de variable `alert`.

```
stop();
if (name = "Steve") {
    play();
} else {
    alert = "You are not Steve!";
}
```

prevFrame

Sintaxis
```
prevFrame();
```

Argumentos
Ninguno.

Descripción
Acción; envía la cabeza lectora al fotograma anterior y la detiene.

Reproductor
Flash 2 o posterior.

Ejemplo
Cuando el usuario hace clic sobre un botón al que se ha asignado la acción `prevFrame`, se envía la cabeza lectora al fotograma anterior.

```
on(release) {
    prevFrame(5);
}
```

Véase también
"MovieClip.prevFrame" a pagina 324

prevScene

Sintaxis

```
prevScene();
```

Argumentos

Ninguno.

Descripción

Acción; envía la cabeza lectora al fotograma 1 de la escena anterior y la detiene.

Reproductor

Flash 2 o posterior.

Véase también

"nextScene" a pagina 330

print

Sintaxis

```
print (destino, "bmovie");
print (destino, "bmax");
print (destino, "bframe");
```

Argumentos

destino El nombre de instancia del clip de película que se va a imprimir. De modo predeterminado, se imprimen todos los fotogramas de la película. Si desea imprimir solamente fotogramas específicos de la película, designe los fotogramas para imprimir anexando una etiqueta de fotograma `#P` a esos fotogramas en el entorno de creación.

bmovie Designa la delimitación de un fotograma específico de una película como el área de impresión para todos los fotogramas imprimibles de la película. Anexe una etiqueta `#b` (en el entorno de creación) para designar al fotograma cuya delimitación desea utilizar como área de impresión.

bmax Designa como área de impresión un compuesto de todas las delimitaciones, de todos los fotogramas imprimibles. Especifica el argumento *bmax* cuando los fotogramas imprimibles de su película varían de tamaño.

bframe Designa que se utilice la delimitación de cada fotograma imprimible como área de impresión para ese fotograma. Esto cambia el área de impresión para cada fotograma y cambia la escala de los objetos para ajustarse al área de impresión. Utilice *bframe* si tiene objetos de diferentes tamaños en cada fotograma y desea que cada objeto rellene la página impresa.

Descripción

Acción; imprime el clip de película *destino* según el modificador de impresora especificado en el argumento. Si desea imprimir solamente fotogramas específicos de la película de destino, anexe una etiqueta de fotograma `#P` a los fotogramas que desea imprimir. Aunque la acción `print` da como resultado impresiones de más alta calidad que la acción `printAsBitmap`, no puede utilizarse para imprimir películas que utilizan transparencias alfa o efectos especiales de color.

Si no especifica el argumento área de impresión, ésta se determina de modo predeterminado por el tamaño del Escenario de la película cargada. La película no hereda el tamaño del Escenario de la película principal. Puede controlar el área de impresión especificando los argumentos *bmovie*, *bmax*, o *bframe*.

Todos los elementos imprimibles de una película deben estar cargados por completo antes de que pueda comenzar la impresión.

La función de impresión de Flash Player soporta impresoras PostScript y no PostScript. Las impresoras no PostScript convierten vectores en mapas de bits.

Reproductor

Flash 5 o posterior.

Ejemplo

El ejemplo siguiente imprimirá todos los fotogramas imprimibles de `myMovie` con el área de impresión definida por la delimitación del fotograma con la etiqueta de fotograma `#b` anexada.

```
print("myMovie","bmovie");
```

El ejemplo siguiente imprimirá todos los fotogramas imprimibles de `myMovie` con el área de impresión definida por la delimitación de cada fotograma.

```
print("myMovie","bframe");
```

Véase también

"printAsBitmap" a pagina 348

printAsBitmap

Sintaxis

```
printAsBitmap(destino, "bmovie");
printAsBitmap(destino, "bmax");
printAsBitmap(destino, "bframe");
```

Argumentos

destino El nombre de instancia del clip de película que se va a imprimir. De modo predeterminado, se imprimen todos los fotogramas de la película. Si desea imprimir solamente fotogramas específicos de la película, designe los fotogramas para imprimir anexando una etiqueta de fotograma #P a esos fotogramas en el entorno de creación.

bmovie Designa la delimitación de un fotograma específico de una película como el área de impresión para todos los fotogramas imprimibles de la película. Anexe una etiqueta #b (en el entorno de creación) para designar al fotograma cuya delimitación desea utilizar como área de impresión.

bmax Designa como área de impresión un compuesto de todas las delimitaciones, de todos los fotogramas imprimibles. Especifica el argumento *bmax* cuando los fotogramas imprimibles de su película varían de tamaño.

bframe Designa que se utilice la delimitación de cada fotograma imprimible como área de impresión para ese fotograma. Esto cambia el área de impresión para cada fotograma y cambia la escala de los objetos para ajustarse al área de impresión. Utilice *bframe* si tiene objetos de diferentes tamaños en cada fotograma y desea que cada objeto rellene la página impresa.

Descripción

Acción; imprime el clip de película *destino* como un mapa de bits. Utilice `printAsBitmap` para imprimir películas que contienen fotogramas con objetos que utilizan transparencia alfa o efectos de color. La acción `printAsBitmap` imprime con la mayor resolución disponible de la impresora para mantener la definición y la calidad mayores posibles. Para calcular el tamaño de archivo imprimible de un fotograma designado para impresión como un mapa de bits, multiplique el ancho de píxel por el alto de píxel por la resolución de la impresora.

Si su película no contiene transparencias alfa o efectos de color se le recomienda que utilice la acción `print` para obtener resultados de mejor calidad.

De modo predeterminado, el área de impresión está determinada por el tamaño del Escenario de la película cargada. La película no hereda el tamaño del Escenario de la película principal. Puede controlar el área de impresión especificando los argumentos *bmovie*, *bmax*, o *bframe*.

Todos los elementos imprimibles de una película deben estar cargados por completo antes de que pueda comenzar la impresión.

La función de impresión de Flash Player soporta impresoras PostScript y no PostScript. Las impresoras no PostScript convierten vectores en mapas de bits.

Reproductor
Flash 5 o posterior.

Véase también
"print" a pagina 346

_quality

Sintaxis
```
_quality
_quality = x;
```

Argumentos
x Una cadena que especifica uno de los valores siguientes:

BAJA Calidad de representación baja. Los gráficos no se suavizan (antialiasing), los mapas de bits no se suavizan.

MEDIA Calidad de representación media. Los gráficos se suavizan (antialiasing) utilizando una cuadrícula de 2x2, pero los mapas de bits no se suavizan Apropiado para películas que no contienen texto.

ALTA Calidad de representación alta. Los gráficos se suavizan (antialiasing) utilizando una cuadrícula de 4x4 y los mapas de bits se suavizan si la película es estática. Esta es la configuración de la calidad de representación predeterminada utilizada por Flash.

MEJOR Calidad de representación muy alta. Los gráficos se suavizan (antialiasing) utilizando una cuadrícula de 4x4 y los mapas de bits se suavizan siempre (smoothing).

Descripción
Propiedad (global); establece o recupera la calidad de representación utilizada para una película.

Reproductor
Flash 5 o posterior.

Ejemplo
El ejemplo siguiente establece la representación de `oldQuality` en ALTA.

```
oldQualtiy = _quality
_quality = "HIGH";
```

Véase también
"_highquality" a pagina 279

random

Sintaxis
```
random();
```

Argumentos
valor El número entero mayor para el que `random` devolverá un valor.

Descripción
Función; devuelve un número entero aleatorio entre 0 y el número entero especificado en el argumento *valor*.

Reproductor
Flash 4. Esta función se ha desestimado en Flash 5; se recomienda la utilización del método `Math.random`.

Ejemplo
La utilización de `random` que se muestra a continuación devuelve un valor de 0, 1, 2, 3 o 4.

```
random(5);
```

Véase también
"Math.random" a pagina 307

removeMovieClip

Sintaxis
```
removeMovieClip(destino);
```

Argumentos
destino La ruta de destino de una instancia de clip de película creada con `duplicateMovieClip`, o el nombre de instancia de un clip de película creado con los métodos `attachMovie` o `duplicateMovie` del objeto MovieClip.

Descripción
Acción; elimina una instancia de clip de película creada con los métodos `attachMovie` o `duplicateMovieClip` del objeto MovieClip o con la acción `attachMovie`.

Reproductor
Flash 4 o posterior.

Véase también
"duplicateMovieClip" a pagina 264
"MovieClip.duplicateMovieClip" a pagina 315
"MovieClip.attachMovie" a pagina 315
"MovieClip.removeMovieClip" a pagina 325

return

Sintaxis

```
return[expresión];

return;
```

Argumentos

expresión Un tipo, cadena. número, matriz u objeto para evaluar y devolver como un valor de la función. Este argumento es opcional.

Descripción

Acción; especifica el valor devuelto por una función. Cuando se ejecuta devolver acción, la *expresión* se evalúa y se devuelve como un valor de la función. La acción devolver hace que la función deje de ejecutarse. Si la sentencia `return` se utiliza solo, o si Flash no encuentra una sentencia `return` durante la acción de bucle, devuelve `null`.

Reproductor

Flash 5 o posterior.

Ejemplo

A continuación se muestra un ejemplo de la utilización de `return`.

```
function sum(a, b, c){
return a + b + c;
}
```

Véase también

"function" a pagina 272

_root

Sintaxis

```
_root;
_root.movieClip;

_root.acción;
```

Argumentos

movieClip El nombre de instancia de un clip de película.

acción El valor establecido para la propiedad. Este es un argumento opcional y puede que no necesite establecerlo en la propiedad.

Descripción
Propiedad; especifica o devuelve una referencia a la Línea de tiempo de la película raíz. Si una película tiene varios niveles, la Línea de tiempo de la película raíz está en el nivel que contiene el script que se está ejecutando actualmente. Por ejemplo, si un script en el nivel 1 evalúa _root, se devuelve el nivel 1.

Especificar `_root` es lo mismo que utilizar la notación de barras "/ " para especificar una ruta relativa dentro del nivel actual.

Reproductor
Flash 4 o posterior.

Ejemplo
El ejemplo siguiente detiene la Línea de tiempo del nivel que contiene el script que se está ejecutando actualmente.

```
_rootl.stop();
```

El ejemplo siguiente envía la Línea de tiempo del nivel actual al fotograma 3.

```
_root.gotoAndStop(3);
```

Véase también
"_parent" a pagina 342
"targetPath" a pagina 379

_rotation

Sintaxis
nombre_instancia._rotation
nombre_instancia._rotation = *entero*

Argumentos
entero El número de grados que se va a girar el clip de película.

nombre_instancia El clip de película que se va a girar.

Descripción
Propiedad; especifica la rotación del clip de película en grados.

Reproductor
Flash 4 o posterior.

scroll

Sintaxis

nombre_variable.scroll = *x*

Argumentos

nombre_variable El nombre de la variable asociada con un campo de texto.

x El número de línea de la línea superior visible en el campo de texto. Puede especificar este valor o utilizar el valor predeterminado de 1. Flash Player actualiza este valor cuando el usuario se desplaza hacia arriba y hacia abajo en el campo de texto.

Descripción

Propiedad; controla la visualización de la información en un campo de texto asociado con una variable. La propiedad `scroll` define donde comienza el campo de texto a visualizar el contenido, después de que lo establezca, Flash Player lo actualiza cuando el usuario se desplaza por el campo de texto. La propiedad `scroll` es útil para dirigir a los usuarios a un párrafo específico en un pasaje largo o para crear campos de texto con desplazamiento. Este propiedad puede recuperarse y modificarse.

Reproductor

Flash 4 o posterior.

Véase también

"maxscroll" a pagina 309

Selection (objeto)

El objeto Selection le permite establecer y controlar el campo de texto de edición seleccionado. El campo de texto de edición seleccionado es el campo donde está situado actualmente el cursor del usuario. Los índices de espacio de selección tienen base cero (donde la primera posición es 0, la segunda 1, etc.).

No existe método constructor para el objeto Selection ya que sólo puede haber un campo seleccionado actualmente al mismo tiempo.

Resumen de los métodos de un objeto Selection

Método	Descripción
`getBeginIndex`	Devuelve el índice al principio del espacio de selección. Devuelve -1 si no hay índice o campo seleccionado actualmente.
`getCaretIndex`	Devuelve la posición de intercalación actual en el espacio de selección actualmente seleccionado. Devuelve -1 si no hay posición de intercalación o espacio de selección seleccionado.
getEndIndex	Devuelve el índice al final del espacio de selección. Devuelve -1 si no hay índice o campo seleccionado actualmente.
`getFocus`	Devuelve el nombre de la variable para el campo de texto de edición seleccionado actualmente. Devuelve nulo si no hay campo de texto de edición seleccionado.
`setFocus`	Selecciona el campo de texto de edición asociado con la variable especificada en el argumento.
`setSelection`	Establece los índices de inicio y fin del espacio de selección.

Selection.getBeginIndex

Sintaxis
```
Selection.getBeginIndex();
```

Argumentos
Ninguno.

Descripción
Método; devuelve el índice al principio del espacio de selección. Si no existe índice o no se ha seleccionado ningún campo actualmente, el método devuelve –1. Los índices del espacio de selección tienen base cero (donde la primera posición es 0, la segunda 1, etc.).

Reproductor
Flash 5 o posterior.

Selection.getCaretIndex

Sintaxis
```
Selection.getCaretIndex();
```

Argumentos
Ninguno.

Descripción
Método; devuelve el índice de la posición del cursor que parpadea. Si no se ve el cursor que parpadea, el método devuelve –1. Los índices del espacio de selección tienen base cero (donde la primera posición es 0, la segunda 1, etc.).

Reproductor
Flash 5 o posterior.

Selection.getEndIndex

Sintaxis
```
Selection.getEndIndex();
```

Argumentos
Ninguno.

Descripción
Método; devuelve el índice al final del espacio de selección actualmente seleccionado. Si no existe índice o no se ha seleccionado espacio de selección, el método devuelve –1. Los índices del espacio de selección tienen base cero (donde la primera posición es 0, la segunda 1, etc.).

Reproductor
Flash 5 o posterior.

Selection.getFocus

Sintaxis
```
Selection.getFocus();
```

Argumentos
Ninguno.

Descripción
Método; devuelve el nombre de la variable para el campo de texto de edición seleccionado actualmente. Si no se ha seleccionado actualmente campo de texto, el método devuelve `null`.

Reproductor
Flash 5 o posterior.

Ejemplo
El código siguiente devuelve el nombre de la variable.
```
_root.anyMovieClip.myTextField.
```

Selection.setFocus

Sintaxis
```
Selection.setFocus(variable);
```

Argumentos
variable Una cadena que especifica el nombre de una variable asociada con un campo de texto utilizando notación de puntos o de barras.

Descripción
Método; selecciona el campo de texto de edición asociado con la *variable* especificada.

Reproductor
Flash 5 o posterior.

Selection.setSelection

Sintaxis
```
Selection.setSelection(inicio, fin);
```

Argumentos
inicio El índice de inicio del espacio de selección.

fin El índice de fin del espacio de selección.

Descripción
Método; establece el espacio de selección del campo de texto seleccionado actualmente. El nuevo espacio de selección comenzará en el índice especificado en el argumento *inicio* y el final en el índice especificado en el argumento *fin*. Los índices de espacio de selección tienen base cero (donde la primera posición es 0, la segunda 1, etc.). Este método no tiene efecto si no existe campo de texto seleccionado actualmente.

Reproductor
Flash 5 o posterior.

set

Sintaxis
```
variable = expresión;
set(variable, expresión);
```

Argumentos
variable El nombre del contenedor que mantiene el valor del argumento *expresión*.

expresión El valor (o una frase que puede evaluarse en un valor) que se asigna a la variable.

Descripción

Acción; asigna un valor a una variable. Una variable es un contenedor que almacena información. El contenedor en sí es siempre el mismo, pero el contenido puede cambiar. La modificación del valor de una variable a medida que la película se reproduce permite registrar y guardar información sobre las acciones del usuario, registrar valores que se modifican conforme la película se reproduce o evaluar si determinada condición es `true` o `false`.

Las variables pueden contener números o cadenas de caracteres. Cada película y clip de película tiene su propio juego de variables y cada una de ellas con su propio valor independiente de variables en otras películas o clips de película.

ActionScript es un lenguaje en el que no se tiene en cuenta el formato escrito. Esto quiere decir que las variables no tienen que definirse explícitamente como que contienen un número o una cadena. Flash interpreta de forma adecuada si el tipo de datos es un número entero o una cadena.

Utilice la sentencia `set` junto con la acción `call` para pasar o devolver valores.

Reproductor

Flash 4 o posterior.

Ejemplo

Este ejemplo establece una variable llamada `orig_x_pos` que almacena la posición original del eje *x* del clip de película `ship` para poder restablecer el barco a su posición de inicio más adelante en la película.

```
on(release) {
    set(x_pos, getProperty ("ship", _x ));
}
```

Esto es equivalente a escribir lo que se muestra a continuación:

```
on(release) {
   orig_x_pos = getProperty ("ship", _x );
}
```

Véase también

"var" a pagina 386
"call" a pagina 237

setProperty

Sintaxis

`setProperty(`*`destino`*`, `*`propiedad`*`, `*`expresión`*`);`

Argumentos

`destino` La ruta al nombre de instancia del clip de película cuya propiedad se está estableciendo.

`propiedad` La propiedad que se va a establecer.

`expresión` El valor en el que está establecida la propiedad.

Descripción
Acción; cambia la propiedad de un clip de película mientras se reproduce la película.

Reproductor
Flash 4 o posterior.

Ejemplo
Esta sentencia establece la propiedad `_alpha` de un clip de película llamado `star` en el 30 por ciento cuando se hace clic sobre el botón.

```
on(release) {
    setProperty("star", _alpha = 30);
}
```

Véase también
"getProperty" a pagina 274

Sound (objeto)

El objeto Sound le permite establecer y controlar los sonidos de una instancia de clip de película particular, o para la Línea de tiempo global, si no especifica un *destino* cuando crea un nuevo objeto Sound. Debe utilizar el constructor `new Sound` para crear una instancia del objeto Sound antes de llamar a los métodos del objeto Sound.

El objeto Sound sólo se admite en Flash Player 5.

Resumen de los métodos de un objeto Sound

Método	Descripción
`attachSound`	Anexa el sonido especificado en el argumento.
`getPan`	Devuelve el valor de la llamada setPan anterior.
`getTransform`	Devuelve el valor de la llamada setTransform anterior.
`getVolume`	Devuelve el valor de la llamada setVolume anterior.
`setPan`	Establece el balance izquierda/derecha del sonido.
`setTransform`	Establece la transformación de un sonido.
`setVolume`	Establece el nivel de volumen de un sonido.
`start`	Comienza a reproducir un sonido desde el principio u opcionalmente desde un punto de desplazamiento establecido en el argumento.
`stop`	Detiene el sonido especificado o todos los sonidos que se están reproduciendo actualmente.

Constructor del objeto Sound.

Sintaxis

```
new Sound();
new Sound(destino);
```

Argumentos

destino La instancia del clip de película a la que se aplica el objeto Sound. Este argumento es opcional.

Descripción

Método; crea un nuevo objeto Sound para un clip de película especificado. Si no especifica un *destino*, el objeto Sound controla todos los sonidos de la Línea de tiempo global.

Reproductor

Flash 5 o posterior.

Ejemplo

```
GlobalSound = new Sound();
MovieSound = new Sound(mymovie);
```

Sound.attachSound

Sintaxis

```
mySound.attachSound("idName");
```

Argumentos

idName El nombre de la nueva instancia del sonido. Este es el mismo nombre introducido en el identificador en el cuadro de diálogo Propiedades de vínculos de símbolos. Este argumento debe aparecer encerrado entre `" "` (comillas).

Descripción

Método; anexa el sonido especificado en el argumento *idName* al objeto Sound especificado. El sonido debe estar en la biblioteca de la película actual y debe estar especificado para la exportación en el cuadro de diálogo Propiedades de vínculos de símbolos. Debe llamar a `Sound.start` para comenzar a reproducir el sonido.

Reproductor

Flash 5 o posterior.

Véase también

"Sound.start" a pagina 365

Sound.getPan

Sintaxis
mySound.getPan();

Argumentos
Ninguno.

Descripción
Método; devuelve el nivel de panorámica establecido en la última llamada `setPan` como un número entero de –100 a 100. La configuración de panorámica controla el balance izquierdo/derecho de los sonidos actuales y futuros de una película.

Este método es acumulativo con los métodos `setVolume` o `setTransform`.

Reproductor
Flash 5 o posterior.

Véase también
"Sound.setPan" a pagina 361

"Sound.setTransform" a pagina 362

Sound.getTransform

Sintaxis
mySound.getTransform();

Argumentos
Ninguno.

Descripción
Método; devuelve la información de transformación de sonido para el objeto Sound especificado con la última llamada `setTransform`.

Reproductor
Flash 5 o posterior.

Véase también
"Sound.setTransform" a pagina 362

Sound.getVolume

Sintaxis
mySound.getVolume();

Argumentos
Ninguno.

Descripción
Método; devuelve el nivel de volumen de sonido como un número entero de 0 a 100, donde 0 es apagado y 100 es a todo volumen. El valor predeterminado es 100.

Reproductor
Flash 5 o posterior.

Véase también
"Sound.setVolume" a pagina 365

Sound.setPan

Sintaxis
mySound.setPan(*pan*);

Argumentos
pan Un número entero que especifica el balance izquierda-derecha de un sonido. Los valores válidos deben estar comprendidos entre -100 y 100, donde -100 utiliza solamente el canal izquierdo, 100 utiliza solamente el canal derecho y 0 hace un balance de sonido uniforme entre los dos canales.

Descripción
Método; determina como se reproduce el sonido en los canales izquierdo y derecho (altavoces). Para los sonidos mono, *pan* afecta a que altavoz (izquierdo o derecho) reproduce el sonido.

Este método es acumulativo con los métodos setVolume y setTransform y llamar a este método elimina y actualiza la configuración anterior de setPan y setTransform.

Reproductor
Flash 5 o posterior.

Ejemplo

El ejemplo siguiente utiliza `setVolume` y `setPan` para controlar un objeto Sound con el destino "u2 " especificado.

```
onClipEvent(mouseDown) {

// create a sound object and

s = new Sound(this);

// attach a sound in the library

s.attachSound("u2");

//set volume at 50%

s.setVolume(50);

//turn off the sound in the right channel

s.setPan(-100);

//start 30 seconds into the sound and play it 5 times

s.start(30, 5);
```

Véase también

"Sound.setTransform" a pagina 362

"Sound.setVolume" a pagina 365

Sound.setTransform

Sintaxis

mySound.setTransform(*soundTransformObject*);

Argumentos

soundTransformObject Un objeto creado con el constructor del objeto Object genérico.

Descripción

Método; establece la información de transformación de sonido para un objeto Sound. Este método es acumulativo con los métodos `setVolume` y `setPan` y llamar a este método elimina y actualiza la configuración anterior de `setPan` o `setVolume`. Esta llamada es para usuarios expertos que desean añadir efectos de sonido interesantes.

Los sonidos necesitan una cantidad considerable de espacio en la unidad de disco y en la memoria. Debido a que los sonidos estéreo utilizan el doble de datos que los sonidos mono, normalmente es mejor utilizar sonidos mono de 22 Khz de 6 bits. Puede utilizar el método `setTransform` para reproducir sonidos mono y para agregar efectos interesantes a los sonidos.

El argumento `sound transformObject` es un objeto que cree utilizando el método constructor del objeto Object genérico con parámetros que especifican como se distribuye el sonido en los canales izquierdo y derecho (altavoces).

Los parámetros para un objeto de transformación de sonido se definen como se muestra a continuación:

`ll` Un valor de porcentaje que especifica qué cantidad de entrada izquierda se reproduce en el altavoz izquierdo (de -100 a 100).

`lr` Un valor de porcentaje que especifica qué cantidad de entrada derecha se reproduce en el altavoz izquierdo (de -100 a 100).

`rr` Un valor de porcentaje que especifica qué cantidad de entrada derecha se reproduce en el altavoz derecho (de -100 a 100).

`rl` Un valor de porcentaje que especifica qué cantidad de entrada izquierda se reproduce en el altavoz derecho (de -100 a 100).

El resultado neto de los parámetros se representa por la fórmula siguiente:

Salida izquierda = entrada izquierda * ll + entrada derecha * lr

Salida derecha = entrada derecha * rr + entrada izquierda * rl

Los valores de entrada izquierda y entrada derecha se determinan por el tipo del sonido (estéreo o mono) de su película.

Los sonidos estéreo dividen la entrada del sonido uniformemente entre los altavoces izquierdo y derecho y de forma predeterminada tienen la configuración de transformación siguiente:

```
ll = 100
lr = 0
rr = 100
rl = 0
```

Los sonidos mono reproducen toda la entrada de sonido en el altavoz izquierdo y de forma predeterminada tienen la configuración de transformación siguiente:

```
ll = 100
lr = 100
rr = 0
rl = 0
```

Reproductor
Flash 5 o posterior.

Ejemplo

El ejemplo siguiente crea un objeto de transformación de sonido que reproduce los canales izquierdo y derecho en el canal izquierdo.

```
mySoundTransformObject = new Object
mySoundTransformObject.ll = 100
mySoundTransformObject.lr = 100
mySoundTransformObject.rr = 0
mySoundTransformObject.rl = 0
```

El código anterior crea un objeto de transformación de sonido. Para poder aplicarlo a un objeto Sound, necesita pasar el objeto al objeto Sound utilizando `setTransform` como se muestra a continuación:

```
mySound.setTransform(mySoundTransformObject);
```

A continuación se muestran ejemplos de configuración que pueden establecerse utilizando `setTransform`, pero no pueden establecerse utilizando `setVolume` o `setPan`, aunque se combinen.

Este código reproduce los canales izquierdo y derecho por el canal izquierdo:

```
mySound.setTransform(soundTransformObjectLeft);
```

En el código anterior, *`soundTransformObjectLeft`* tiene los parámetros siguientes:

```
ll = 100
lr = 100
rr = 0
rl = 0
```

Ejemplo

Este código reproduce los sonidos estéreo como si fueran mono:

```
setTransform(soundTransformObjectMono);
```

En el código anterior, *`soundTransformObjectMono`* tiene los parámetros siguientes:

```
ll = 50
lr = 50
rr = 50
rl = 50
```

Ejemplo

Este código reproduce el canal izquierdo a media capacidad y agrega el resto del canal izquierdo al canal derecho:

```
setTransform(soundTransformObjectHalf);
```

En el código anterior, *`soundTransformObjectHalf`* tiene los parámetros siguientes:

```
ll = 50
lr = 0
rr = 100
rl = 50
```

Véase también

"Constructor del objeto Object" a pagina 337

Sound.setVolume

Sintaxis

```
mySound.setVolume(volumen);
```

Argumentos

volumen Un número de 0 a 100 que representa el nivel de volumen. 100 es a todo volumen y 0 es sin volumen. El valor predeterminado es 100.

Descripción

Método; establece el sonido para el objeto Sound.

Este método es acumulativo con los métodos `setPan` y `setTransform`.

Reproductor

Flash 5 o posterior.

Ejemplo

El ejemplo siguiente establece el volumen en el 50% y transfiere el sonido a lo largo del tiempo del altavoz izquierdo al altavoz derecho.

```
onClipEvent (load) {
   i = -100;
   s = new sound();
   s.setVolume(50);
}
onClipEvent (enterFrame) {
   S.setPan(i++);
}
```

Véase también

"Sound.setPan" a pagina 361
"Sound.setTransform" a pagina 362

Sound.start

Sintaxis

```
mySound.start();
```

```
mySound.start([segundo_desplaz, bucle]);
```

segundo_desplaz Un argumento opcional que le permite comenzar a reproducir el sonido en un punto específico. Por ejemplo, si tiene un sonido de 30 segundos y desea que el sonido comience a reproducirse a la mitad, especifique 15 para el argumento *segundo_desplaz*. El sonido no se retrasa 15 segundos, sino que comienza a reproducirse en la marca de 15 segundos.

bucle Un argumento opcional que le permite especificar el número de veces que el sonido debe realizar un bucle.

Descripción
Método; comienza a reproducir el último sonido anexado, desde el principio si no se especifican argumentos o comenzando en el punto del sonido especificado por el argumento *segundo_desplaz*.

Reproductor
Flash 5 o posterior.

Véase también
"Sound.setPan" a pagina 361
"Sound.stop" a pagina 366

Sound.stop

Sintaxis
```
mySound.stop();
mySound.stop(["nombre_Id"]);
```

Argumentos
nombre_Id Un argumento opcional que especifica que deje de reproducirse un sonido específico. El argumento *nombre_Id* debe ir encerrado entre comillas (" ").

Descripción
Método; detiene todos los sonidos que se están reproduciendo actualmente si no se especifican argumentos o solamente el sonido especificado en el argumento *nombre_Id*.

Reproductor
Flash 5 o posterior.

Véase también
"Sound.start" a pagina 365

_soundbuftime

Sintaxis
```
_soundbuftime = entero;
```

Argumentos
entero El número de segundos antes de que la película comience el flujo.

Descripción
Propiedad (global); establece el número de segundos de sonido de flujo que va a sufrir regulación previa. El valor predeterminado es 5 segundos.

Reproductor
Flash 4 o posterior.

startDrag

Sintaxis

```
startDrag(destino);

startDrag(destino,[bloqueado]);

startDrag(destino [,bloqueado [,izquierda , arriba , derecha,
abajo]]);
```

Argumentos

destino La ruta de destino del clip de película que se va a arrastrar.

bloqueado Un valor Booleano que especifica si el clip de película arrastrable está bloqueado en el centro de la posición del ratón (*true*), o bloqueado en el punto en el que el usuario hizo clic por primera vez en el clip de película (*false*). Este argumento es opcional.

izquierda, arriba, derecha, abajo Los valores relativos a las coordenadas del elemento principal del clip de película que especifica un rectángulo de limitación para el clip de película. Estos argumentos son opcionales.

Descripción

Acción; hace que el clip de película *destino* se pueda arrastrar mientras se reproduce la película. Sólo un clip de película puede arrastrarse al mismo tiempo. Una vez que se ha ejecutado una acción `startDrag`, el clip de película permanece arrastrable hasta que se detiene específicamente por una acción `stopDrag` o hasta que se llama a una acción `startDrag` de otro clip de película.

Ejemplo

Para crear un clip de película que los usuarios puedan colocar en cualquier ubicación, anexe las acciones `startDrag` y `stopDrag` a un botón dentro del clip de película, como se muestra a continuación:

```
on(press) {
    startDrag("",true);
}
on(release) {
    stopDrag();
}
```

Véase también

"stopDrag" a pagina 369

"_droptarget" a pagina 263

stop

Sintaxis
```
stop;
```

Argumentos
Ninguno.

Descripción
Acción; detiene el clip de película que se está reproduciendo actualmente. La utilización más corriente de esta acción es para controlar los clips de película con botones.

Reproductor
Flash 3 o posterior.

stopAllSounds

Sintaxis
```
stopAllSounds();
```

Argumentos
Ninguno.

Descripción
Acción; detiene todos los sonidos que se están reproduciendo actualmente sin detener la cabeza lectora. Los sonidos establecidos en flujo continuarán reproduciéndose mientras la cabeza lectora se mueve sobre los fotogramas en los que se encuentran.

Reproductor
Flash 3 o posterior.

Ejemplo
El código siguiente podría aplicarse a un botón que, cuando se hace clic sobre él, detiene todos los sonidos de la película.

```
on(release) {
    stopAllSounds();
}
```

Véase también
"Sound (objeto)" a pagina 358

stopDrag

Sintaxis
```
stopDrag();
```

Argumentos
Ninguno.

Descripción
Acción; detiene la operación de arrastre actual.

Reproductor
Flash 4 o posterior.

Ejemplo
Esta sentencia detiene la acción de arrastre de la instancia `mc` cuando el usuario suelta el botón del ratón.

```
on(press) {
   startDrag("mc");
}
on(release) {
 stopdrag();
}
```

Véase también
"startDrag" a pagina 367
"_droptarget" a pagina 263

String (función)

Sintaxis
```
String(expresión);
```

Argumentos
expresión El número, Booleano, variable u objeto que se va a convertir en una cadena.

Descripción
Función; devuelve una representación de cadena del argumento especificado como se muestra a continuación:

Si *x* es Booleano, la cadena devuelta es `true`, o `false`.

Si *x* es un número, la cadena devuelta es una representación decimal del número.

Si `x` es una cadena, la cadena devuelta es *x*.

Si *x* es un objeto, el valor devuelto es una representación de cadena del objeto generado por la llamada a la propiedad de cadena del objeto o por la llamada a `object.toString`, si no existe tal propiedad.

Si *x* es un clip de película, el valor devuelto es la ruta de destino del clip de película en notación de barras (/).

Si *x* no está definida, el valor devuelto es una cadena vacía.

Reproductor
Flash 3 o posterior.

Véase también
"Object.toString" a pagina 338
"Number.toString" a pagina 336
"Cadena (objeto)" a pagina 371
"" " (delimitador de cadena)" a pagina 370

" " (delimitador de cadena)

Sintaxis
"*text*"

Argumentos
texto Cualquier texto.

Descripción
Delimitador de cadena; cuando se utiliza antes y después de una cadena, las comillas indican que la cadena es literal (no una variable, valor numérico u otro elemento de ActionScript.

Reproductor
Flash 4 o posterior.

Ejemplo
Esta sentencia utiliza comillas para indicar que la cadena "Isla Príncipe Eduardo" es una cadena literal y no el valor de la variable:

```
province = "Prince Edward Island"
```

Véase también
"Cadena (objeto)" a pagina 371
"String (función)" a pagina 369

Cadena (objeto)

El objeto String es envolvente del tipo de datos primitivos de cadena, lo que le permite utilizar los métodos y propiedades del objeto String para manipular tipos de valor de cadena primitivos. Puede convertir el valor de cualquier objeto en una cadena utilizando la función `String()`.

Todos los métodos del objeto String, excepto para `concat`, `fromCharCode`, `slice`, y `substr` son genéricos. Esto quiere decir que los propios métodos pueden llamar a `this.toString` antes de realizar sus operaciones y puede utilizar estos métodos con otros objetos que no sean Cadena.

Puede llamar a cualquiera de los métodos del objeto String utilizando el método constructor `new String` o utilizando un valor de literal de cadena. Si especifica un literal de cadena, el intérprete de ActionScript lo convierte automáticamente en un objeto String temporal, llama al método y después descarta el objeto String temporal. Puede utilizar la propiedad `String.length` con un literal de cadena.

Es importante que no confunda un literal de cadena con una instancia del objeto String. En el ejemplo siguiente la primera línea de código crea el literal de cadena `s1`, y la segunda línea del código crea una instancia del objeto String `s2`.

```
s1 = "foo"
s2 = new String("foo")
```

Se le recomienda que utilice los literales de cadena a no ser que necesite utilizar específicamente un objeto String, ya que los objetos Cadena pueden tener un comportamiento contraintuitivo.

Resumen de métodos del objeto String

Método	Descripción
`charAt`	Devuelve un número que corresponde con la colocación del carácter en la cadena.
`charCodeAt`	Devuelve el valor del carácter en el índice dado como un número entero de 16 bits entre 0 y 65535.
`concat`	Combina el texto de dos cadenas y devuelve una nueva cadena.
`fromCharCode`	Devuelve una cadena hecha de los caracteres especificados en los argumentos.
`indexOf`	Busca la cadena y devuelve el índice del valor especificado en los argumentos. Si un valor aparece más de una vez, se devuelve el índice de la primera ocurrencia. Si no se encuentra valor, se devuelve -1.

Método	Descripción
`lastIndexOf`	Devuelve la última ocurrencia de la subcadena dentro de la cadena que aparece antes de la posición de inicio especificada en el argumento o -1 si no se encuentra.
`slice`	Extrae una sección de una cadena y devuelve una nueva cadena.
`split`	Divide una cadena en una matriz de cadenas dividiendo las cadenas en subcadenas.
`substr`	Devuelve un número especificado de caracteres en una cadena, comenzando en la ubicación especificada en el argumento.
`substring`	Devuelve los caracteres entre dos índices, especificados en los argumentos, en la cadena.
`toLowerCase`	Convierte la cadena a minúsculas y devuelve el resultado.
`toUpperCase`	Convierte la cadena a mayúsculas y devuelve el resultado.

Resumen de las propiedades de un objeto String

Propiedad	Descripción
`length`	Devuelve la longitud de la cadena.

Constructor del objeto String.

Sintaxis

```
new String(valor);
```

Argumentos

`valor` El valor inicial del nuevo objeto String.

Descripción

Constructor; crea un nuevo objeto String.

Reproductor

Flash 5 o posterior.

Véase también

“String (función)” a pagina 369
“" " (delimitador de cadena)” a pagina 370

String.charAt

Sintaxis
myString.charAt(*índice*);

Argumentos
índice El número del carácter en la cadena que se va a devolver.

Descripción
Método; devuelve el carácter especificado en el argumento *índice*. El índice del primer carácter en una cadena es 0. Si *índice* no es un número de 0 a `string.length` - 1, se devuelve una cadena vacía.

Reproductor
Flash 5 o posterior.

String.charCodeAt

Sintaxis
myString.charCodeAt(*índice*);

Argumentos
índice El número del carácter para el que se recupera el valor.

Descripción
Método; devuelve el valor del carácter especificado por *índice*. El valor devuelto es un número entero de 16 bits de 0 a 65535.

Este método es similar a `string.charAt` excepto en que el valor devuelto es para el carácter en una ubicación específica, en lugar de una subcadena que contiene el carácter.

Reproductor
Flash 5 o posterior.

String.concat

Sintaxis
myString.concat(*valor1,...valorN*);

Argumentos
valor1,...valorN Cero o más valores que se van a concatenar.

Descripción
Método; combina los valores especificados y devuelve una nueva cadena. Si es necesario, cada argumento *valor* se convierte en una cadena y se anexa, en orden, al final de la cadena.

Reproductor
Flash 5 o posterior.

String.fromCharCode

Sintaxis

```
myString.fromCharCode(c1,c2,...cN);
```

Argumentos

c1,c2,...cN Los caracteres que se van a convertir en una cadena.

Descripción

Método; devuelve una cadena hecha de los caracteres especificados en los argumentos.

Reproductor

Flash 5 o posterior.

String.indexOf

Sintaxis

```
myString.indexOf(valor);
```

```
myString.index of (valor, inicio);
```

Argumentos

valor Un número entero o cadena que especifica la subcadena que se va a buscar dentro de *myString*.

inicio Un número entero que especifica el punto de inicio de la subcadena. Este argumento es opcional.

Descripción

Método; busca en la cadena y devuelve la posición de la primera ocurrencia del *valor* especificado. Si no se encuentra el valor, el método devuelve -1.

Reproductor

Flash 5 o posterior.

String.lastIndexOf

Sintaxis

```
myString.lastIndexOf(subcadena);
```

```
myString.lastIndexOf(subcadena, inicio);
```

Argumentos

subcadena Un número entero o cadena que especifica la cadena que se va a buscar.

inicio Un número entero que especifica el punto de inicio dentro de la subcadena. Este argumento es opcional.

Descripción
Método; busca en la cadena y devuelve la posición de la última ocurrencia de `subcadena` que se encuentra dentro de la cadena de llamada.. Si no se encuentra *`subcadena`*, el método devuelve -1.

Reproductor
Flash 5 o posterior.

String.length

Sintaxis
```
string.length
```

Argumentos
Ninguno.

Descripción
Propiedad; devuelve el número de caracteres en el objeto String especificado. El índice del último carácter de cualquier cadena `x`, es x.length-1.

Reproductor
Flash 5 o posterior.

String.slice

Sintaxis
```
myString.slice(inicio, fin);
```

Argumentos
`inicio` Un número que especifica el índice del punto de inicio del sector. Si *`inicio`* es un número negativo, el punto inicial se especifica desde el final de la matriz, donde -1 es el último elemento.

`fin` Un número que especifica el índice del punto final del sector. Si no se especifica *`fin`*, el sector incluye todos los caracteres desde el inicio al final de la cadena. Si *`fin`* es un número negativo, el punto final se determina desde el final de la cadena, donde -1 es el último carácter.

Descripción
Método; extrae un sector o una subcadena del objeto String especificado, después la devuelve como una nueva cadena sin modificar el objeto String original. La cadena devuelta incluye el carácter *`inicio`* y todos los caracteres hasta (pero sin incluir) el carácter *`fin`*.

Reproductor
Flash 5 o posterior.

String.split

Sintaxis
```
myString.split(delimitador);
```

Argumentos
delimitador El carácter utilizado para delimitar la cadena.

Descripción
Método. divide un objeto String rompiendo la cadena cuando se produce el argumento *delimitador* especificado y devuelve las subcadenas en una matriz. Si no se especifica delimitador, la matriz devuelta contiene solamente un elemento: la propia cadena. Si el delimitador es una cadena vacía, cada carácter del objeto String se convierte en un elemento de la matriz.

Reproductor
Flash 5 o posterior.

String.substr

Sintaxis
```
myString.substr(inicio, longitud);
```

Argumentos
inicio Un número entero que indica la posición del primer carácter de la subcadena que se está creando. Si *inicio* es un número negativo, el punto inicial se determina desde el final de la matriz, donde -1 es el último carácter.

longitud El número de caracteres en la subcadena que se está creando. Si no se especifica *longitud*, la subcadena incluye todos los caracteres desde el inicio al final de la cadena.

Descripción
Método; devuelve los caracteres de una cadena desde el índice especificado en el argumento *inicio*, hasta el número de caracteres especificados en el argumento *longitud*.

Reproductor
Flash 5 o posterior.

String.substring

Sintaxis
```
myString.substring(desde, hasta);
```

Argumentos
desde Un número entero que indica la posición del primer carácter de la subcadena que se está creando. Los valores válidos para *desde* van de 0 a `string.length` - 1.

hasta Un número entero que es 1 + el índice del último carácter de la subcadena que se está creando. Los valores válidos para *hasta* van de 1 a `string.length`. Si no se especifica el argumento *hasta*, el final de la subcadena es el final de la cadena. Si *desde* es igual a *hasta*, el método devuelve una cadena vacía. Si *desde* es mayor que *hasta*, los argumentos se intercambian automáticamente antes de que se ejecute la función.

Descripción
Método; devuelve una cadena que consiste en los caracteres entre los puntos especificados por los argumentos *desde* y *hasta*.

Reproductor
Flash 5 o posterior.

String.toLowerCase

Sintaxis
```
myString.toLowerCase();
```

Argumentos
Ninguno.

Descripción
Método; devuelve una copia del objeto String, con todos los caracteres en mayúsculas convertidos en minúsculas.

Reproductor
Flash 5 o posterior.

String.toUpperCase

Sintaxis
```
myString.toUpperCase();
```

Argumentos
Ninguno.

Descripción
Método; devuelve una copia del objeto String, con todos los caracteres en minúsculas convertidos en mayúsculas.

Reproductor
Flash 5 o posterior.

substring

Sintaxis
```
substring(cadena, índice, recuento);
```

Argumentos
cadena La cadena de la que extraer la nueva cadena.

índice El número del primer carácter que se va a extraer.

recuento El número de caracteres que se van a incluir en la cadena extraída, sin incluir el carácter de índice.

Descripción
Función de cadena; extrae parte de una cadena.

Reproductor
Flash 4 o posterior. Esta función se ha desestimado en Flash 5.

Véase también
"String.substring" a pagina 377

_target

Sintaxis
```
nombre_instancia._target
```

Argumentos
nombre_instancia El nombre de una instancia de clip de película.

Descripción
Propiedad (de sólo lectura); devuelve la ruta de destino de la instancia de clip de película especificada en el argumento *nombre_instancia*.

Reproductor
Flash 4 o posterior.

targetPath

Sintaxis
`targetpath(movieClipObject);`

Argumentos
movieClipObject Referencia (por ejemplo, `_root` o `_parent`) al clip de película del que se está recuperando la ruta de destino.

Descripción
Función; devuelve una cadena que contiene la ruta de destino de *movieClipObject*. La ruta de destino se devuelve en notación de puntos. Para recuperar la ruta de destino en notación de barras, utilice la propiedad `_destino`.

Reproductor
Flash 5 o posterior.

Ejemplo
Los ejemplos siguientes son equivalentes. El primer ejemplo utiliza notación de puntos y el segundo ejemplo utiliza notación de barras.

```
targetPath (Board.Block[index*2+1]) {

play();

}
```

Es equivalente a:

```
tellTarget ("Board/Block:" + (index*2+1)) {

play();

}
```

Véase también
"eval" a pagina 266

tellTarget

Sintaxis
```
tellTarget(destino) {
sentencia;
}
```

Argumentos
destino Un cadena de ruta de destino que especifica la Línea de tiempo que se va a controlar.

sentencia Instrucciones aplicadas a la Línea de tiempo destino.

Descripción

Acción; aplica las instrucciones especificadas en el argumento *sentencias* a la Línea de tiempo especificada en el argumento *destino*. La acción `tellTarget` es útil para los controles de navegación. Asigna `tellTarget` a botones que detienen o inician los clips de película en otras partes del Escenario. También puede hacer que los clips de película vayan a un fotograma concreto de dicho clip. Por ejemplo, podría asignar `tellTarget` a botones que detienen o inician los clips de película en el Escenario o solicitar a los clips de película que salten a un fotograma concreto.

La acción `tellTarget` es muy similar a la acción `with`, excepto en que `with` toma un clip de película u otro objeto como *destino* y `tellTarget` requiere una ruta de destino a un clip de película y no puede controlar objetos.

Reproductor

Flash 3 o posterior. Esta acción se ha desestimado en Flash 5; se recomienda la utilización de la acción `with`.

Ejemplo

Esta sentencia `tellTarget` controla la instancia de clip de película `ball` en la Línea de tiempo principal. El fotograma 1 del clip de película está vacío y tiene una acción `stop` de modo que es invisible en el Escenario. Cuando se hace clic sobre el botón con la acción siguiente, `tellTarget` dice a la cabeza lectora del clip de película `ball` que vaya al fotograma 2 y reproduzca la animación que comienza ahí.

```
on(release) {
    tellTarget("ball") {
        gotoAndPlay(2);
    }
}
```

Véase también

"with" a pagina 389

this

Sintaxis

```
this
```

Argumentos

Ninguno.

Descripción

Palabra clave; hace referencia a una instancia de objeto o de clip de película. La palabra clave `this` tiene el mismo objetivo y función en ActionScript que en JavaScript, con algunas funciones adicionales. En ActionScript, cuando se ejecuta un script, `this` hace referencia a la instancia de clip de película que contiene el script. Cuando se utiliza con una invocación de método, `this` contiene una referencia al objeto que contiene el método ejecutado.

Reproductor
Flash 5 o posterior.

Ejemplo
En el ejemplo siguiente, la palabra clave `this` hace referencia al objeto Círculo.

```
function Circle(radius){
  this.radius = radius;
  this.area = math.PI * radius * radius;
}
```

En la sentencia siguiente asignada a un fotograma, la palabra clave `this` hace referencia al clip de película actual.

```
//sets the alpha property of the current movie clip to 20.

this._alpha = 20;
```

En la sentencia siguiente dentro de un controlador `onClipEvent`, la palabra clave `this` hace referencia al clip de película actual.

```
//when the movie clip loads, a startDrag operation is initiated
for the current movie clip.

onClipEvent (load) {
  startDrag (this, true);
}
```

Véase también
"new" a pagina 328

toggleHighQuality

Sintaxis
```
toggleHighQuality();
```

Argumentos
Ninguno.

Descripción
Acción; activa y desactiva el suavizado (antialiasing) en Flash Player. El suavizado (antialiasing) suaviza los bordes de los objetos y hace más lenta la reproducción de la película. La acción `toggleHighQuality` afecta a todas las películas de Flash Player.

Reproductor
Flash 2 o posterior.

Ejemplo
El código siguiente podría aplicarse a un botón que cuando se hace clic sobre él, activará o desactivará el suavizado (antialiasing).

```
on(release) {
    toggleHighQuality();
}
```

Véase también
"_quality" a pagina 349
"_highquality" a pagina 279

_totalframes

Sintaxis
nombre_instancia._totalframes

Argumentos
nombre_instancia El nombre del clip de película que se va a evaluar.

Descripción
Propiedad (de sólo lectura); evalúa el clip de película especificado en el argumento *nombre_instancia* y devuelve el número total de fotogramas de la película.

Reproductor
Flash 4 o posterior.

trace

Sintaxis
trace(*expresión*);

Argumentos
expresión Una sentencia que se va a evaluar. Cuando prueba una película, el resultado del argumento *expresión* aparece en la ventana Salida.

Descripción
Acción; evalúa la *expresión* y muestra el resultado en la ventana de Salida en modo de prueba de película.

Utilice `trace` para registrar notas de programación o para ver mensajes en la ventana Salida mientras prueba una película. Utilice el parámetro *expresión* para comprobar si existe una condición o para ver valores en la ventana Salida. La acción `trace` es similar a la función `alert` en JavaScript.

Reproductor
Flash 4 o posterior.

Ejemplo

Este ejemplo corresponde a un juego en el que una instancia de clip de película arrastrable llamada `rabbi` debe liberarse en un destino específico. Una sentencia condicional evalúa la propiedad `_droptarget` y ejecuta diferentes acciones dependiendo de donde se libera `rabbi`. La acción `trace` se utiliza al final del script para evaluar la ubicación del clip de película `rabbi` y muestra el resultado en la ventana Salida. Si `rabbi` no se comporta como se esperaba (por ejemplo, si encaja en el destino incorrecto), los valores enviados a la ventana Salida por la acción `trace` le ayudarán a determinar el problema del script.

```
on(press) {
rabbi.startDrag();
}
on(release) {
if(eval(_droptarget) != target) {
     rabbi._x = rabbi_x;
     rabbi._y = rabbi_y;
} else {
rabbi_x = rabbi._x;
rabbi_y = rabbi._y;
target = "_root.pasture";
}
trace("rabbi_y = " + rabbi_y);
trace("rabbi_x = " + rabbi_x);
stopDrag();
}
```

typeof

Sintaxis

`typeof(`*`expresión`*`);`

Argumentos

`expresión` Una cadena, clip de película, objeto o función.

Descripción

Operador; un operador unario situado antes de un solo argumento. Hace que Flash evalúe *`expresión;`* el resultado es una cadena que especifica si la expresión es una cadena, un clip de película, un objeto o una función.

Reproductor

Flash 5 o posterior.

unescape

Sintaxis
```
unescape(x);
```

Argumentos
x Una cadena con secuencia hexadecimal de escape.

Descripción
Función de nivel superior; evalúa el argumento *x* como una cadena, decodifica la cadena de un formato de URL codificado (convirtiendo todas las secuencias hexadecimales en caracteres ASCII) y devuelve la cadena.

Reproductor
Flash 5 o posterior.

Ejemplo
El ejemplo siguiente muestra el proceso de conversión de caracteres de escape en caracteres no de escape.

```
escape("Hello{[World]}");
```

El resultado con caracteres de escape es el que se muestra a continuación:

```
("Hello%7B%5BWorld%5D%7D');
```

Utilice `unescape` para volver al formato original:

```
unescape("Hello%7B%5BWorld%5D%7D")
```

El resultado es el que se muestra a continuación:

```
Hello{[World]}
```

unloadMovie

Sintaxis
```
unloadMovie(ubicación);
```

Argumentos
ubicación El nivel de profundidad o clip de película de destino del que descargar la película.

Descripción
Acción; elimina una película de Flash Player que se cargó anteriormente utilizando la acción `loadMovie` .

Reproductor
Flash 3 o posterior.

Ejemplo

El ejemplo siguiente descarga la película principal, dejando vacío el Escenario.

```
unloadMovie(_root);
```

El ejemplo siguiente descarga una película del nivel 15, cuando el usuario hace clic con el ratón.

```
on(press) {
  unloadMovie(_level15);
}
```

Véase también

"loadMovie" a pagina 294

updateAfterEvent

Sintaxis

```
updateAfterEvent(movie clip event);
```

Argumentos

movie clip event Puede especificar uno de los valores siguientes como un evento de clip de película:

- `mouseMove` La acción se inicia cada vez que se mueve el ratón. Utilice las propiedades `_xmouse` y `_ymouse` para determinar la posición actual del ratón.
- `mouseDown` La acción se inicia si se presiona el botón izquierdo del ratón.
- `mouseUp` La acción se inicia si se suelta el botón izquierdo del ratón.
- `keyDown` La acción se inicia cuando se presiona una tecla. Utilice el método `Key.getCode` para recuperar información sobre la última tecla presionada.
- `keyUp` La acción se inicia cuando se suelta una tecla. Utilice el método `key.getCode` para recuperar información sobre la última tecla presionada.

Descripción

Acción; actualiza la visualización (independientemente de los fotogramas por segundo establecidos para la película) después de que se haya completado el evento de clip especificado en los argumentos. Esta acción no aparece en la lista del panel Acciones de Flash. Utilizando `updateAfterEvent` con acciones de arrastre que especifican la propiedades `_x` y `_y` durante el movimiento del ratón, permite arrastrar los objetos suavemente sin el efecto de parpadeo de la pantalla.

Reproductor

Flash 5 o posterior.

Véase también

"onClipEvent" a pagina 338

_url

Sintaxis
nombre_instancia._url

Argumentos
nombre_instancia El clip de película de destino.

Descripción
Propiedad (de sólo lectura); recupera la URL del archivo SWF del que se descargó el clip de película.

Reproductor
Flash 4 o posterior.

var

Sintaxis
var *nombre_variable1* [= *valor1*] [...,*nombre_variableN* [=*valorN*]];

Argumentos
nombre_variable El nombre de la variable que se va a declarar.

valor El valor que se asigna a la variable.

Descripción
Acción; se utiliza para declarar variables locales. Si declara variables locales dentro de una función, las variables se definen para la función y expiran al final de la llamada a la función. Si las variables no se declaran dentro de un bloque, pero la lista de acciones se ejecutó con una acción call, las variables son locales y expiran al final de la lista actual. Si las variables no se declaran dentro de un bloque y la lista de acciones actual no se ejecutó con la acción llamada, las variables no son locales.

Reproductor
Flash 5 o posterior.

_visible

Sintaxis
nombre_instancia._visible
nombre_instancia._visible = *Boolean*;

Argumentos
Boolean Introduzca un valor *true* o *false* para especificar si el clip de película es visible.

Descripción
Propiedad; determina si la película especificada por el argumento *nombre_instancia* es visible o no. Los clips de película que no son visibles (propiedad establecida en `false`) están deshabilitados. Por ejemplo, en un botón de un clip de película con la propiedad `_visible` establecida en `false` no puede hacerse clic.

Reproductor
Flash 4 o posterior.

void

Sintaxis
```
void (expresión);
```

Argumentos
expresión Una expresión de cualquier valor.

Descripción
Operador; un operador unario que descarta el valor *expresión* y devuelve un valor sin definir. El operador `void` con frecuencia se utiliza para evaluar una URL para comprobar si tiene efectos secundarios sin tener que ver la expresión evaluada en la ventana del navegador. El operador `void` también se utiliza en comparaciones utilizando el operador `==` , para comprobar valores sin definir.

Reproductor
Flash 5 o posterior.

while

Sintaxis
```
while(condición) {
sentencia(s);
}
```

Argumentos
condición La sentencia que se vuelve a evaluar cada vez que se ejecuta la acción `while`. Si la sentencia evalúa como `true`, se ejecuta la expresión de la *sentencia(s)*.

sentencia(s) La expresión que se ejecutará si la condición evalúa como `true`.

Descripción

Acción; ejecuta una sentencia o una serie de sentencias repetidamente en un bucle mientras que el argumento de la condición es `true`. Al final de cada acción `while`, Flash reinicia el bucle volviendo a comprobar la condición. Si la condición es `false` o igual a 0, Flash se salta la primera sentencia después de la acción `while`.

Los bucles se utilizan con frecuencia para realizar una acción mientras que una variable de contador es menor que un valor especificado. Al final de cada bucle, el contador se incrementa, hasta que se alcanza el valor umbral, la *`condición`* ya no es `true` y el bucle finaliza.

Reproductor

Flash 4 o posterior.

Ejemplo

Este ejemplo duplica cinco clips de película en el Escenario, cada uno con una posición *x* e *y* generadas aleatoriamente, `xscale` y `yscale` y la propiedad `_alpha` para conseguir un efecto de dispersión. La variable `foo` se inicializa con el valor 0. El argumento *`condición`* está establecido de tal modo que el bucle `while` se ejecuta cinco veces o mientras que el valor de la variable `foo` es menor que 5. Dentro del bucle `while`, se duplica un clip de película y se utiliza `setProperty` para ajustar las diferentes propiedades del clip de película duplicado. La última sentencia del bucle incrementa `foo` de modo que cuando el valor llega a 5, el argumento *`condición`* evalúa como `false` y el bucle no se ejecutará.

```
on(release) {
   foo = 0;
   while(foo < 5) {
        duplicateMovieClip("/flower", "mc" + foo, foo);
        setProperty("mc" + foo, _x, random(275));
        setProperty("mc" + foo, _y, random(275));
        setProperty("mc" + foo, _alpha, random(275));
        setProperty("mc" + foo, _xscale, random(200));
        setProperty("mc" + foo, _yscale, random(200));
        foo = foo + 1;
    }
}
```

Véase también

"do...while" a pagina 262
"continue" a pagina 242

_width

Sintaxis

```
nombre_instancia._width
nombre_instancia._width =valor;
```

Argumentos

valor La anchura de la película en píxeles.

nombre_instancia Un nombre de instancia de un clip de película para el que se va a establecer o recuperar la propiedad _width.

Descripción

Propiedad; establece la anchura de la película. En versiones anteriores de Flash, _height y _widtheran propiedades de sólo lectura, en Flash 5 se pueden establecer y también recuperar.

Reproductor

Flash 4 como propiedad de sólo lectura. En Flash 5 o posterior, esta propiedad puede establecerse y también recuperarse.

Ejemplo

El código de ejemplo siguiente establece las propiedades de la altura y la anchura de un clip de película cuando el usuario hace clic con el ratón.

```
onclipEvent(mouseDown) {
   _width=200;
   _height=200;
}
```

Véase también

"_height" a pagina 278

with

x209E6 | IDS_ACTIONHELP_WITH, sentencia with

Sintaxis

```
with (objeto) {
sentencia(s);
}
```

Argumentos

objecto Una instancia de un objeto o clip de película de ActionScript.

sentencia(s) Una acción o grupo de acciones encerradas entre llaves.

Descripción

Acción; cambia temporalmente el ámbito (o ruta de destino) utilizado para evaluar expresiones y acciones en la *sentencia(s)*. Después de que se ejecute la acción `with`, se restaura la cadena del ámbito a su estado original.

El *objeto* se convierte en el contexto en el que se leen las propiedades, variables y funciones. Por ejemplo, si *objeto* es `myArray` y dos de las propiedades especificadas son `length` y `concat`, esas propiedades se leen automáticamente como `myArray.length` y `myArray.concat`. En otro ejemplo, si *objeto* es `state.california`, es como si cualquiera de las acciones o sentencias dentro de la acción `with` fueran llamadas desde dentro de la instancia `california`.

Para averiguar el valor de un identificador en *sentencia(s)*, ActionScript inicia al comienzo de la cadena de ámbito especificada por el *objeto* y busca el identificador en cada nivel de la cadena de ámbito, en un orden específico.

La cadena de ámbito utilizada por la acción `with` para resolver identificadores comienza con el primer elemento de la lista siguiente y continúa hasta el último, como se muestra a continuación:

- *objeto* hecho referencia por la acción `with` más interior.
- *objeto* hecho referencia por la acción `with` más exterior.
- Objeto de activación (un objeto temporal que se crea automáticamente cuando se llama a una función que contiene las variables locales llamadas en la función).
- Clip de película que contiene el script que se está ejecutando actualmente.
- Objeto Global (objetos predefinidos como Matemáticas, Cadena).

En Flash 5 la acción `with` sustituye a la acción `tellTarget` que se ha desestimado. Se le recomienda que utilice `with` en lugar de `tellTarget` debido a que es una extensión estándar de ActionScript del estándar ECMA-262. La diferencia principal entre las acciones `with` y `tellTarget` es que `with` toma una referencia de un clip de película u otro objeto como su argumento, mientras que `tellTarget` toma una cadena de ruta de destino que identifica a un clip de película y no puede utilizarse para destinar objetos.

Para establecer una variable dentro de una acción `with`, la variable debe haber sido declarada fuera de la acción `with` o debe introducir la ruta de acceso completa a la Línea de tiempo en la que desea que resida la variable. Si establece una variable en una acción `with` sin haberla declarado, la acción `with` buscará el valor según la cadena de ámbito. Si la variable todavía no existe, el nuevo valor se establecerá en la Línea de tiempo desde la que se llamó a la acción `with`.

Ejemplo

El ejemplo siguiente establece la propiedades *x* e *y* de la instancia `someOtherMovieClip` y después indica a `someOtherMovieClip` que vaya al fotograma 3 y se detenga:

```
with (someOtherMovieClip) {
  _x = 50;
  _y = 100;
  gotoAndStop(3);
}
```

El siguiente fragmento de código es como escribiría el código anterior, sin utilizar una acción `with`.

```
someOtherMovieClip._x = 50;
someOtherMovieClip._y = 100;
someOtherMovieClip.gotoAndStop(3);
```

Este código también podría haberse escrito utilizando la acción `tellTarget`.

```
tellTarget ("someOtherMovieClip") {
  _x = 50;
  _y = 100;
  gotoAndStop(3);
}
```

La acción `with` es útil para acceder a varios elementos de una lista de cadena de ámbito simultáneamente. En el ejemplo siguiente, el objeto incorporado `Math` se sitúa al frente de la cadena de ámbito. Establecer `Math` como objeto predeterminado, resuelve los identificadores `cos`, `sin` y `PI` en `Math.cos`, `Math.sin` y `Math.PI`, respectivamente. Los identificadores `a`, `x`, `y` y `r`, no son métodos o propiedades del objeto `Math`, pero como existen en el ámbito de activación de objeto de la función `polar`, se resuelven en las variables locales correspondientes.

```
function polar(r){
  var a, x, y
  with (Math) {
    a = PI * r * r
    x = r * cos(PI)
    y = r * sin(PI/2)
}
trace("area = " +a)
trace("x = " + x)
trace("y = " + y)
}
```

Puede utilizar acciones `with` anidadas para acceder a la información en varios ámbitos. En el ejemplo siguiente, la instancia `fresno` y la instancia `salinas` son secundarias respecto a la instancia `california`. La sentencia establece los valores `_alpha` de `fresno` y `salinas` sin cambiar el valor `_alpha` de `california`.

```
with (california){
   with (fresno){
      _alpha = 20;
   }
   with (salinas){
      _alpha = 40;
   }
}
```

Véase también

"tellTarget" a pagina 379

_x

Sintaxis

nombre_instancia._x
nombre_instancia._x = *entero*

Argumentos

entero La coordenada local *x* de la película.

nombre_instancia El nombre de una instancia de clip de película.

Descripción

Propiedad; establece la coordenada *x* de la película relativa a las coordenadas locales del clip de película principal. Si un clip de película está en la Línea de tiempo principal, su sistema de coordenadas hace referencia a la esquina superior izquierda del Escenario como (0, 0). Si el clip de película se encuentra dentro de otro clip de película que tiene transformaciones, el clip de película está en el sistema de coordenadas local del clip de película que lo contiene. Así, para un clip de película girado 90 grados en sentido contrario a las agujas del reloj, el clip de película secundario hereda un sistema de coordenadas girado 90 grados en sentido contrario a las agujas del reloj. Las coordenadas del clip de película hacen referencia a la posición del punto de registro.

Reproductor

Flash 3 o posterior.

Véase también

"_y" a pagina 419
"_xscale" a pagina 418

XML (objeto)

Utilice los métodos y propiedades del objeto XML para cargar, analizar, enviar, construir y manipular árboles de documentos XML.

Debe utilizar el constructor `new XML()` para crear una instancia del objeto XML antes de llamar a los métodos del objeto XML.

aEl lenguaje XML está admitido por Flash 5 o versiones posteriores de Flash Player.

Resumen de los métodos del objeto XML

Método	Descripción
`appendChild`	Anexa un nodo al final de la lista del objeto secundario especificado.
`cloneNode`	Duplica el nodo especificado y opcionalmente duplica repetidamente todos los nodos secundarios.
`createElement`	Crea un nuevo elemento XML.
`createTextNode`	Crea un nuevo nodo de texto XML.
`hasChildNodes`	Devuelve `true` si el nodo especificado tiene nodos secundarios, en caso contrario devuelve `false`.
`insertBefore`	Inserta un nodo delante de un nodo existente en la lista de nodos secundarios especificada.
`load`	Carga un documento (especificado por el objeto XML) de una URL.
`onLoad`	Una función de repetición de llamada para `load` y `sendAndLoad`.
`parseXML`	Analiza un documento XML en el árbol de objeto XML especificado.
`removeNode`	Elimina el nodo especificado de su principal.
`send`	Envía el objeto XML especificado a una URL.
`sendAndLoad`	Envía el objeto XML especificado a una URL y carga la respuesta del servidor en otro objeto XML.
`toString`	Convierte el nodo especificado y cualquiera de sus secundarios en texto XML.

Resumen de las propiedades de un objeto XML

Propiedad	Descripción
`docTypeDecl`	Establece y devuelve la información sobre la declaración DOCTYPE de un documento XML.
`firstChild`	Hace referencia al primer nodo secundario de la lista del nodo especificado.
`lastChild`	Hace referencia al último nodo secundario de la lista del nodo especificado.
`loaded`	Comprueba si se ha cargado el objeto XML especificado.
`nextSibling`	Hace referencia al siguiente valor colateral en la lista de nodos secundarios del nodo principal.
`nodeName`	Devuelve el nombre de etiqueta de un elemento XML.
`nodeType`	ReDevuelve el tipo del nodo especificado (elemento o nodo de texto XML).
`nodeValue`	Devuelve el texto del nodo especificado si el nodo es un nodo de texto.
`parentNode`	Hace referencia al nodo principal del nodo especificado.
`previousSibling`	Hace referencia al valor colateral anterior en la lista de nodos secundarios del nodo principal.
`status`	Devuelve el código de estado numérico que indica el éxito o el fracaso de una operación de análisis de un documento XML.
`xmlDecl`	Establece y devuelve la información sobre la declaración de documento de un documento XML.

Resumen de los métodos del objeto XML

Método	Descripción
`attributes`	Devuelve una matriz asociativa que contiene todos los atributos del nodo especificado.
`childNodes`	Devuelve una matriz que contiene la referencia a los nodos secundarios del nodo especificado.

Constructor del objeto XML.

Sintaxis

```
new XML();
```

```
nuevo XML(origen);
```

Argumentos

origen El documento XML que se va a analizar para crear un nuevo objeto XML.

Descripción

Constructor; crea un nuevo objeto XML. Debe utilizar el método constructor para crear una instancia del objeto XML antes de llamar a cualquiera de los métodos del objeto XML.

La primera sintaxis construye un nuevo objeto XML vacío.

La segunda sintaxis construye un nuevo objeto XML analizando el documento XML especificado en el argumento *origen* y rellena el objeto XML recién creado con el árbol del documento XML resultante.

Nota: Los métodos `createElement` y `createTextnode` son los métodos 'constructor' para crear los elementos y los nodos de texto en un árbol de documento XML.

Reproductor

Flash 5 o posterior.

Ejemplo

El ejemplo siguiente crea un nuevo objeto XML vacío.

```
myXML = new XML();
```

Véase también

"XML.createTextNode" a pagina 398
"XML.createElement" a pagina 398

XML.appendChild

Sintaxis

```
myXML.appendChild(childNode);
```

Argumentos

childNode El nodo secundario que se va a agregar a la lista de nodos secundarios del objeto XML.

Descripción
Método; anexa el nodo secundario especificado a la lista de nodos secundarios del objeto XML. El nodo anexado se ubica en la estructura del árbol una vez que se ha eliminado de su nodo principal existente, si existe.

Reproductor
Flash 5 o posterior.

Ejemplo
El ejemplo siguiente duplica el último nodo de `doc1` y lo anexa a `doc2`.

```
doc1 = new XML(src1);
doc2 = new XML();
node = doc1.lastChild.cloneNode(true);
doc2.appendChild(node);
```

XML.attributes

Sintaxis
myXML.attributes;

Argumentos
Ninguno.

Descripción
Colección (de lectura-escritura); devuelve una matriz asociativa que contiene todos los atributos del objeto XML especificado.

Reproductor
Flash 5 o posterior.

Ejemplo
El ejemplo siguiente muestra los nombres de los atributos XML en la ventana Salida.

```
str = "<mytag name=\"Val\"> intem </mytage>";
doc = new XML(str);
y = doc.fristChild.attributes.name;
  trace (y);
doc.firstChild.attributes.order = "first";
z = doc.firstChild.attributes.order
  trace(z);
```

Lo que se muestra a continuación es lo que aparece en la ventana Salida.

```
Val
First
```

XML.childNodes

Sintaxis
myXML.childNodes;

Argumentos
Ninguno.

Descripción
Colección (de sólo lectura); devuelve una matriz de los nodos secundarios del objeto XML especificado. Cada elemento de la matriz es una referencia a un objeto XML que representa un nodo secundario. Esta es una propiedad de sólo lectura y no puede utilizarse para manipular nodos secundarios. Utilice los métodos appendChild, insertBefore y removeNode para manipular nodos secundarios.

Esta colección no está definida para nodos de texto (nodeType == 3).

Reproductor
Flash 5 o posterior.

XML.cloneNode

Sintaxis
myXML.cloneNode(*deep*);

Argumentos
deep Valor Booleano que especifica si los nodos secundarios del objeto XML secundario se duplican repetidamente.

Descripción
Método; construye y devuelve un nuevo nodo XML del mismo tipo, nombre, valor y atributos que el objeto XML especificado. Si *deep* está establecido como true, todos los nodos secundarios se duplican repetidamente, lo que da como resultado una copia exacta del árbol de documentos del objeto.

Reproductor
Flash 5 o posterior.

XML.createElement

Sintaxis
myXML.createElement(*name*);

Argumentos
name El nombre de etiqueta del elemento XML que se está creando.

Descripción
Método; crea un nuevo elemento XML con el nombre especificado en el argumento. Inicialmente, el nuevo elemento no tiene nodo principal ni secundarios. El método devuelve una referencia al objeto XML recién creado que representa el elemento. Este método y `createTextNode` son los métodos constructor para crear nodos para un objeto XML.

Reproductor
Flash 5 o posterior.

XML.createTextNode

Sintaxis
myXML.createTextNode(*texto*);

Argumentos
texto El texto utilizado para crear el nuevo nodo de texto.

Descripción
Método; crea un nuevo nodo de texto XML con el texto especificado. Inicialmente, el nuevo elemento no tiene nodo principal y los nodos de texto no pueden tener nodos secundarios. Este método devuelve una referencia al objeto XML que representa el nuevo nodo de texto. Este método y `createElement` son los métodos constructor para crear nodos para un objeto XML.

Reproductor
Flash 5 o posterior.

XML.docTypeDecl

Sintaxis

myXML.XMLdocTypeDecl;

Argumentos

Ninguno.

Descripción

Propiedad; establece y devuelve la información sobre la declaración DOCTYPE de un documento XML. Una vez que se ha analizado el texto XML en un objeto XML, la propiedad `XML.docTypeDecl` del objeto XML se establece en el texto de la declaración DOCTYPE del documento XML. Por ejemplo, `<!DOCTYPE greeting SYSTEM "hello.dtd">`. Esta propiedad se establece utilizando una representación de cadena de la declaración DOCTYPE, no un nodo de objeto XML.

El analizador XML de ActionScript no es una analizador de validación. El analizador lee la declaración DOCTYPE y se almacena en la propiedad `docTypeDecl`, pero no se realiza validación DTD.

Si no se encuentra ninguna declaración DOCTYPE durante la operación de análisis, `XML.docTypeDecl` se establece como sin definir. `XML.toString` realiza la salida de contenido de `XML.docTypeDecl` inmediatamente después de la declaración XML almacenada en `XML.xmlDecl` y antes que cualquier otro texto del objeto XML. Si `XML.docTypeDecl` es sin definir, no se realiza salida de declaración DOCTYPE.

Reproductor

Flash 5 o posterior.

Ejemplo

El ejemplo siguiente utiliza `XML.docTypeDecl` para establecer la declaración DOCTYPE de un objeto XML.

```
myXML.docTypeDecl = "<!DOCTYPE greeting SYSTEM \"hello.dtd\">";
```

Véase también

"XML.toString" a pagina 409
"XML.xmlDecl" a pagina 409

XML.firstChild

Sintaxis
myXML.firstChild;

Argumentos
Ninguno.

Descripción
Propiedad (de sólo lectura); evalúa el objeto XML especificado y hace referencia al primer nodo secundario en la lista de nodos secundarios del nodo principal. Esta propiedad es `null` si el nodo no tiene nodos secundarios. Esta propiedad está sin definir si el nodo es un nodo de texto. Esta es una propiedad de sólo lectura y no puede utilizarse para manipular nodos secundarios; utilice los métodos `appendChild`, `insertBefore` y `removeNode` para manipular nodos secundarios.

Reproductor
Flash 5 o posterior.

Véase también
"XML.appendChild" a pagina 395

"XML.insertBefore" a pagina 401

"XML.removeNode" a pagina 407

XML.haschildNodes

Sintaxis
myXML.hasChildNodes();

Argumentos
Ninguno.

Descripción
Método; evalúa el objeto XML especificado y devuelve `true` si hay nodos secundarios; en caso contrario devuelve `false`.

Reproductor
Flash 5 o posterior.

Ejemplo
El ejemplo siguiente utiliza la información del objeto XML en una función definida por el usuario.

```
if (rootNode.hasChildNodes()) {
  myfunc (rootNode.firstChild);
}
```

XML.insertBefore

Sintaxis
myXML.insertBefore(*childNode, beforeNode*);

Argumentos
*childNod*e El nodo que se va a insertar.

beforeNode El nodo anterior al punto de inserción para *childNode*.

Descripción
Método; inserta un nuevo nodo secundario en la lista de nodos secundarios del objeto XML, antes de `beforeNode`.

Reproductor
Flash 5 o posterior.

XML.lastChild

Sintaxis
myXML.lastChild;

Argumentos
Ninguno.

Descripción
Propiedad (de sólo lectura); evalúa el objeto XML y hace referencia al último nodo secundario en la lista de nodos secundarios del nodo principal. Este método devuelve `null` si el nodo no tiene nodos secundarios. Esta es una propiedad de sólo lectura y no puede utilizarse para manipular nodos secundarios; utilice los métodos `appendChild`, `insertBefore` y `removeNode` para manipular nodos secundarios.

Reproductor
Flash 5 o posterior.

Véase también
"XML.appendChild" a pagina 395
"XML.insertBefore" a pagina 401
"XML.removeNode" a pagina 407

XML.load

Sintaxis

```
myXML.load(url);
```

Argumentos

url La URL donde está ubicado el documento XML que se va a cargar. La URL debe estar en el mismo subdominio que la URL donde reside actualmente la película.

Descripción

Método; carga un documento XML de la URL especificada y sustituye el contenido del objeto XML especificado con los datos XML descargados. El proceso de carga es asíncrono; no finaliza inmediatamente después de que se ejecute el método `load`. Cuando se ejecuta `load`, la propiedad de objeto XML `loaded` se establece en `false`. Cuando finalizan de descargarse los datos XML, la propiedad `loaded` se establece en `true` y se invoca al método `onLoad`. Los datos XML no se analizan hasta que no se han descargado por completo. Si el objeto XML contenía anteriormente cualquiera de los árboles XML, se descartan.

Puede especificar su propia función de repetición de llamada en lugar del método `onLoad`.

Reproductor

Flash 5 o posterior.

Ejemplo

A continuación se muestra un ejemplo sencillo de la utilización de `XML.load`.

```
doc = new XML();
doc.load ("theFile.xml");
```

Véase también

"XML.onLoad" a pagina 405
"XML.loaded" a pagina 402

XML.loaded

Sintaxis

```
myXML.loaded;
```

Argumentos

Ninguno.

Descripción

Propiedad (de sólo lectura); determina si el proceso de carga del documento iniciado por la llamada `XML.load` se ha completado. Si el proceso se completa con éxito, el método devuelve `true`; en caso contrario, devuelve `false`.

Reproductor
Flash 5 o posterior.

Ejemplo
El ejemplo siguiente utiliza `XML.loaded` en un script sencillo.

```
if (doc.loaded) {
  gotoAndPlay(4)
}
```

XML.nextSibling

Sintaxis
myXML`.nextSibling;`

Argumentos
Ninguno.

Descripción
Propiedad (de sólo lectura); evalúa el objeto XML y hace referencia al siguiente valor colateral en la lista de nodos secundarios del nodo principal. Este método devuelve `null` si el nodo no tiene un nodo de valor colateral siguiente. Esta es una propiedad de sólo lectura y no puede utilizarse para manipular nodos secundarios. Utilice los métodos `appendChild`, `insertBefore` y `removeNode` para manipular nodos secundarios.

Reproductor
Flash 5 o posterior.

Véase también
"XML.appendChild" a pagina 395
"XML.insertBefore" a pagina 401
"XML.removeNode" a pagina 407

XML.nodeName

Sintaxis
myXML`.nodeName;`

Argumentos
Ninguno.

Descripción
Propiedad; toma o devuelve el nombre de nodo del objeto XML. Si el objeto XML es un elemento XML (`nodeType` == 1), `nodeName` es el nombre de la etiqueta que representa el nodo en el archivo XML. Por ejemplo, `TITLE` es el nodeName de una etiqueta HTML `TITLE`. Si el objeto XML es un nodo de texto (`nodeType` == 3), nodeName es `null`.

Reproductor
Flash 5 o posterior.

Véase también
"XML.nodeType" a pagina 404

XML.nodeType

Sintaxis
```
myXML.nodeType;
```

Argumentos
Ninguno.

Descripción
Propiedad (de sólo lectura); toma o devuelve un valor `nodeType`, donde 1 es un elemento XML y 3 es un nodo de texto.

Reproductor
Flash 5 o posterior.

Véase también
"XML.nodeValue" a pagina 404

XML.nodeValue

Sintaxis
```
myXML.nodeValue;
```

Argumentos
Ninguno.

Descripción
Propiedad; devuelve el valor de nodo del objeto XML. Si el objeto XML es un nodo de texto, el nodo Tipo es 3 y `nodeValue` es el texto del nodo. Si el objeto XML es un elemento XML, tienen un `null nodeValue` y es de sólo lectura.

Reproductor
Flash 5 o posterior.

Véase también
"XML.nodeType" a pagina 404

XML.onLoad

Sintaxis

myXML.onLoad(*éxito*);

Argumentos

éxito Un valor Booleano que indica si el objeto XML se ha cargado con éxito con una operación XML.load o XML.sendAndLoad.

Descripción

Método; invocado por Flash Player cuando se recibe del servidor un documento XML. Si el documento XML se recibe con éxito, el argumento *éxito* es true. Si el documento no se ha recibido o se ha producido un error al recibir la respuesta del servidor, el argumento *éxito* es false. La implantación predeterminada de este método no está activa. Para ignorar la implantación predeterminada, debe asignar una función que contenga sus propias acciones.

Reproductor

Flash 5 o posterior.

Ejemplo

El ejemplo siguiente crea una película de Flash sencilla para una aplicación de escaparate de tienda de comercio electrónico sencilla. Se utiliza el método sendAndLoad para transmitir un elemento que contiene el nombre y la contraseña del usuario e instalar un controlador onLoad para manejar la respuesta del servidor.

```
var myLoginReply = new XML();
myLoginReply.onLoad = myOnLoad;
myXML.sendAndLoad("http://www.samplestore.com/login.cgi",
                  myLoginReply);
function myOnLoad(success) {
  if (success) {
    if (e.firstChild.nodeName == "LOGINREPLY" &&
        e.firstChild.attributes.status == "OK") {
      gotoAndPlay("loggedIn")
    } else {
      gotoAndStop("loginFailed")
    }
  } else {
    gotoAndStop("connectionFailed")
  }
}
```

Véase también

"function" a pagina 272
"XML.load" a pagina 402
"XML.sendAndLoad" a pagina 407

XML.parentNode

Sintaxis
myXML.parentNode;

Argumentos
Ninguno.

Descripción
Propiedad (de sólo lectura); hace referencia al nodo principal del objeto XML especificado o devuelve `null` si el nodo no tiene nodo principal. Esta es una propiedad de sólo lectura y no puede utilizarse para manipular nodos secundarios; utilice los métodos `appendChild`, `insertBefore` y `removeNode` para manipular secundarios.

Reproductor
Flash 5 o posterior.

XML.parseXML

Sintaxis
myXML.parseXML(*origen*);

Argumentos
origen El texto XML que se va a analizar y pasar al objeto `XML` especificado.

Descripción
Método; analiza el texto XML especificado en el argumento *origen* y rellena el objeto XML especificado con el árbol XML resultante.
Se descartan cualquiera de los árboles existentes del objeto XML.

Reproductor
Flash 5 o posterior.

XML.previousSibling

Sintaxis
myXML.previousSibling;

Argumentos
Ninguno.

Descripción
Propiedad (de sólo lectura); evalúa el objeto XML y hace referencia al valor colateral anterior en la lista de nodos secundarios del nodo principal. Devuelve `null` si el nodo no tiene un nodo de valor colateral anterior. Esta es una propiedad de sólo lectura y no puede utilizarse para manipular nodos secundarios; utilice los métodos `appendChild`, `insertBefore` y `removeNode` para manipular nodos secundarios.

Reproductor
Flash 5 o posterior.

XML.removeNode

Sintaxis
```
myXML.removeNode();
```

Argumentos
Ninguno.

Descripción
Método; elimina el objeto XML especificado de su principal.

Reproductor
Flash 5 o posterior.

XML.send

Sintaxis
```
myXML.send(url);
myXML.send(url, ventana);
```

Argumentos
url La URL de destino para el objeto XML especificado.

ventana La ventana del navegador en la que se van a mostrar los datos devueltos por el servidor: `_self` especifica el fotograma actual en la ventana actual, `_blank` especifica una nueva ventana, `_parent` especifica el nivel principal del fotograma actual y `_top`especifica el fotograma de nivel superior en la ventana actual.

Descripción
Método; codifica el objeto XML especificado en un documento XML y lo envía a la URL especificada utilizando el método `POST`.

Reproductor
Flash 5 o posterior.

XML.sendAndLoad

Sintaxis
```
myXML.sendAndLoad(url,objeto_XML_destino);
```

Argumentos
url La URL de destino para el objeto XML especificado. La URL debe estar en el mismo subdominio que la URL de donde se descargó la película.

objeto_XML_destino Un objeto XML creado con el método constructor XML, que recibirá la información devuelta del servidor.

Descripción
Método; codifica el objeto XML especificado en un documento XML, lo envía a la URL especificada utilizando el método `POST`, descarga la respuesta del servidor y después la carga en el *`objeto_XML_destino`* especificado en los argumentos. La respuesta del servidor se carga del mismo modo utilizado por el método `load`.

Reproductor
Flash 5 o posterior.

Véase también
"`XML.load`" a pagina 402

XML.status

Sintaxis

```
myXML.status;
```

Argumentos
Ninguno.

Descripción
Propiedad; establece automáticamente y devuelve un valor numérico que indica si un documento XML se analizó con éxito en un objeto XML. A continuación se muestra una lista de los códigos de estado numérico y una descripción de cada uno.

- 0 Sin error; el análisis se completó con éxito.
- -2 Una sección de CDATA no se finalizó correctamente.
- -3 La declaración XML no se finalizó correctamente.
- -4 La declaración DOCTYPE no se finalizó correctamente.
- -5 Un comentario no se finalizó correctamente.
- -6 Un elemento XML estaba mal formado.
- -7 Memoria insuficiente.
- -8 Un valor de atributo no se finalizó correctamente.
- -9 Una etiqueta de inicio no coincidía con una etiqueta final.
- -10 Se ha encontrado una etiqueta final que no coincide con una etiqueta de inicio.

Reproductor
Flash 5 o posterior.

XML.toString

Sintaxis
myXML.toString();

Argumentos
Ninguno.

Descripción
Método; evalúa el objeto XML especificado, construye una representación de textura de la estructura XML incluidos el nodo principal, nodos secundarios y atributos y devuelve el resultado como una cadena.

Para objetos de nivel superior XML (los creados con el constructor), `XML.toString` produce de salida la declaración XML del documento (almacenada en `XML.xmlDecl`), seguida por la declaración del documento `DOCTYPE` (almacenada en `XML.docTypeDecl`), seguida por la representación de todos los nodos XML del objeto. Si `XML.xmlDecl` está sin definir, no se realiza salida de declaración XML. Si `XML.docTypeDecl` está sin definir, no se realiza salida de declaración `XML.docTypeDecl`.

Reproductor
Flash 5 o posterior.

Ejemplo
A continuación se muestra un ejemplo del código del método `XML.toString`.

```
node = new XML("<h1>test</h1>");
trace(node.toString());
sends
<H1>test</H1>
to the output window
```

Véase también
"XML.xmlDecl" a pagina 409
"XML.docTypeDecl" a pagina 399

XML.xmlDecl

Sintaxis
myXML.xmlDecl;

Argumentos
Ninguno.

Descripción

Propiedad; establece y devuelve la información sobre la declaración de un documento XML. Después de que se ha analizado el documento en un objeto XML, esta propiedad se establece utilizando el texto de la declaración XML del documento. Esta propiedad se establece utilizando una representación de cadena de la declaración XML, no un nodo de objeto XML. Si no se ha encontrado declaración XML durante la operación de análisis, la propiedad se establece en sin definir. `XML.toString` produce la salida del contenido de `XML.xmlDecl` antes de cualquier otro texto del objeto XML. Si `XML.xmlDecl` contiene el tipo `undefined`, no se produce salida de declaración XML.

Reproductor

Flash 5 o posterior.

Ejemplo

El ejemplo siguiente utiliza `XML.xmlDecl` para establecer la declaración de documento XML de un objeto XML.

```
myXML.xmlDecl = "<?xml version=\"1.0\" ?>";
```

Véase también

"XML.toString" a pagina 409
"XML.docTypeDecl" a pagina 399

XMLSocket (objeto)

El objeto XMLSocket implanta sockets clientes que permiten que el equipo que ejecuta Flash Player se comunique con el equipo servidor identificado por una dirección IP o un nombre de dominio.

Utilización del objeto XMLSocket

Para utilizar el objeto XMLSocket, el equipo servidor debe ejecutar un daemon que comprenda el protocolo utilizado por el objeto XMLSocket. El protocolo es el que se muestra a continuación:

- Los mensajes XML se envían por una conexión socket en flujo TCP/IP de dúplex completo.
- Cada mensaje XML es un documento XML completo, finalizado en un byte cero.
- Se pueden enviar y recibir un número ilimitado de mensajes XML por una sola conexión XMLSocket.

El objeto XMLSocket es útil para las aplicaciones cliente-servidor que requieren un estado latente bajo, como los sistemas de chat en tiempo real. Una solución de chat basada en HTTP tradicional consulta frecuentemente al servidor y descarga los nuevos mensajes utilizando una solicitud HTTP. Por el contrario, una solución de chat XMLSocket mantiene una conexión abierta con el servidor, lo que permite a éste enviar inmediatamente los mensajes entrantes sin una solicitud del cliente.

Puede suponer un reto establecer un servidor para que se comunique con el objeto XMLSocket. Si su aplicación no requiere interactividad en tiempo real, utilice la acción `loadVariables`, o la conectividad de servidor XML basada en HTTP de Flash (`XML.load`, `XML.sendAndLoad`, `XML.send`), en lugar del objeto XMLSocket.

Para utilizar los métodos del objeto XMLSocket, primero debe utilizar el constructor, `new XMLSocket`, para crear un nuevo objeto XMLSocket.

XMLSocket y la seguridad

Debido a que el objeto XMLSocket establece y mantiene una conexión abierta con el servidor, se han puesto las restricciones siguientes al objeto XMLSocket por razones de seguridad.

- El método `XMLSocket.connect` puede conectar solamente números de puerto TCP mayores o iguales que 1024. Una consecuencia de esta restricción es que los daemon del servidor que se comunican con el objeto XMLSocket también pueden asignarse únicamente a números de puerto mayores o iguales que 1024. Los números de puerto por debajo de 1024 se utilizan con frecuencia para los servicios del sistema como FTP, Telnet y HTTP, bloqueando así el objeto XMLSocket en estos puertos. La restricción de número de puerto limita la posibilidad de que se acceda y se usen estos recursos de modo inapropiado.
- El método `XMLSocket.connect` puede conectar solamente con un equipo en el mismo subdominio donde reside el archivo SWF (película). Esta restricción no se aplica a las películas que se ejecutan fuera de un disco local. (Esta restricción es idéntica a las reglas de seguridad para `loadVariables`, `XML.sendAndLoad` y `XML.load`).

Resumen de los métodos del objeto XMLSocket

Método	Descripción
`close`	Cierra una conexión de socket abierta.
`connect`	Establece una conexión con el servidor especificado.
`onClose`	Una función de repetición de llamada invocada cuando se cierra una conexión XMLSocket.
`onConnect`	Una función de repetición de llamada invocada cuando se establece una conexión XMLSocket.
`onXML`	Una función de repetición de llamada que se invoca cuando llega un objeto XML del servidor.
`send`	Envía un objeto XML al servidor.

Constructor del objeto XMLSocket

Sintaxis
```
new XMLSocket();
```

Argumentos
Ninguno.

Descripción
Constructor; crea un nuevo objeto XMLSocket. El objeto XMLSocket no está conectado inicialmente con ningún servidor. Debe llamar al método `XMLSocket.connect` para conectar el objeto con un servidor.

Reproductor
Flash 5 o posterior.

Ejemplo
```
myXMLSocket = new XMLSocket();
```

Véase también
"XMLSocket.connect" a pagina 413

XMLSocket.close

Sintaxis
```
myXMLSocket.close();
```

Argumentos
Ninguno.

Descripción
Método; cierra la conexión especificada por el objeto XMLSocket.

Reproductor
Flash 5 o posterior.

Véase también
"XMLSocket.connect" a pagina 413

XMLSocket.connect

Sintaxis
```
myXMLSocket.connect(anfitrión, puerto);
```

Argumentos
anfitrión Un nombre de dominio DNS completamente capacitado o una dirección IP en el formato *aaa.bbb.ccc.ddd*. Puede especificar `null` para conectar el servidor anfitrión en el que reside la película.

puerto El número de puerto TCP en el anfitrión utilizado para establecer una conexión. El número de puerto debe ser 1024 o superior.

Descripción
Método; establece una conexión con el anfitrión de Internet especificado utilizando el puerto TCP especificado (debe ser 1024 o superior) y devuelve `true` o `false` dependiendo de si la conexión se ha establecido con éxito. Si no conoce el número de puerto de su equipo de Internet anfitrión, póngase en contacto con su administrador de red. Si se está utilizando la conexión Flash Netscape o el control ActiveX, el anfitrión especificado en el argumento debe tener el mismo subdominio que el anfitrión de donde se descargó la película.

Si especifica `null` para el argumento *anfitrión*, el anfitrión con el que se contacta será el anfitrión donde reside la película que llama a `XMLSocket.connect`. Por ejemplo, si la película se descargó de http://www.yoursite.com, especificar `null` para el argumento *anfitrión* es lo mismo que introducir la dirección IP de www.yoursite.com.

Si `XMLSocket.connect` devuelve un valor de `true`, la fase inicial del proceso de conexión se realiza con éxito, más tarde, se invoca al método `XMLSocket.onConnect` para determinar si la conexión final tuvo éxito o falló. Si `XMLSocket.connect` devuelve `false`, no se pudo establecer una conexión.

Reproductor
Flash 5 o posterior.

Ejemplo

El ejemplo siguiente utiliza `XMLSocket.connect` para conectar con el anfitrión donde reside la película y utiliza `trace` para mostrar el valor devuelto que indica el éxito o el fallo de la conexión.

```
function myOnConnect(success) {
  if (success) {
    trace ("Connection succeeded!")
  } else {
    trace ("Connection failed!")
  }
}
socket = new XMLSocket()
socket.onConnect = myOnConnect
if (!socket.connect(null, 2000)) {
  trace ("Connection failed!")
}
```

Véase también

"function" a pagina 272
"XMLSocket.onConnect" a pagina 415

XMLSocket.onClose

Sintaxis

myXMLSocket.onClose();

Argumentos

Ninguno.

Descripción

Método; una función de repetición de llamada que se invoca solamente cuando una conexión abierta es cerrada por el servidor. La implantación predeterminada de este método no realiza acciones. Para ignorar la implantación predeterminada, debe asignar una función que contenga sus propias acciones.

Reproductor

Flash 5 o posterior.

Véase también

"function" a pagina 272
"XMLSocket.onConnect" a pagina 415

XMLSocket.onConnect

Sintaxis

myXMLSocket.onConnect(*éxito*);

Argumentos

éxito Un valor Booleano que indica si una conexión de socket se estableció con éxito (`true` o `false`).

Descripción

Método; una función de repetición de llamada que es invocada por Flash Player cuando una solicitud de conexión iniciada por medio del método `XMLSocket.connect` ha tenido éxito o ha fallado. Si la conexión ha tenido éxito, el argumento *éxito* es `true`; en caso contrario el argumento *éxito* es `false`.

La implantación predeterminada de este método no realiza acciones. Para ignorar la implantación predeterminada, debe asignar una función que contenga sus propias acciones.

Reproductor

Flash 5 o posterior.

Ejemplo

El ejemplo siguiente muestra el proceso de especificación de una función de sustitución para el método `onConnect` en una aplicación de chat sencilla.

La función controla a qué pantalla se lleva al usuario, dependiendo de si se ha establecido con éxito una conexión. Si la conexión se ha establecido con éxito, se lleva a los usuarios a la pantalla principal de chat en el fotograma con la etiqueta `startChat`. Si la conexión no tiene éxito, los usuarios van a una pantalla con información de resolución de problemas en el fotograma con la etiqueta`connectionFailed`.

```
function myOnConnect(success) {
  if (success) {
    gotoAndPlay("startChat")
  } else {
    gotoAndStop("connectionFailed")
  }
}
```

Después de crear el objeto XMLSocket utilizando el método constructor, el script instala el método `onConnect` utilizando el operador de asignación:

```
socket = new XMLSocket()

socket.onConnect = myOnConnect
```

Finalmente, la conexión se inicia. Si `connect` devuelve `false`, la película se envía directamente al fotograma con la etiqueta `connectionFailed` y nunca se invoca a `onConnect`. Si `connect` devuelve `true`, la película salta al fotograma con la etiqueta `waitForConnection`, que es la pantalla "Por favor, espere ". La película permanece en el fotograma `waitForConnection` hasta que se invoca al controlador `onConnect`, lo que sucede en algún momento futuro dependiendo de la latencia de la red.

```
if (!socket.connect(null, 2000)) {
  gotoAndStop("connectionFailed")
} else {
  gotoAndStop("waitForConnection")
}
```

Véase también

"XMLSocket.connect" a pagina 413
"function" a pagina 272

XMLSocket.onXML

Sintaxis

myXMLSocket.onXML(*objeto*);

Argumento

objeto Una instancia del objeto XML que contiene un documento XML analizado recibido de un servidor.

Descripción

Método; una función de repetición de llamada invocada por Flash Player cuando el objeto XML especificado que contiene un documento XML llega por una conexión XMLSocket abierta. Una conexión XMLSocket puede utilizarse para transferir un número ilimitado de documentos XML entre el cliente y el servidor. Cada documento termina en un byte cero. Cuando Flash Player recibe el byte cero, analiza todos los XML recibidos desde el byte cero anterior o desde que se estableció la conexión si este es el primer mensaje que se recibe. Cada lote de XML analizados se trata como un solo documento XML y se pasa al método `onXML`.

La implantación predeterminada de este método no realiza acciones. Para ignorar la implantación predeterminada, debe asignar una función que contenga acciones que defina.

Reproductor

Flash 5 o posterior.

Ejemplo

La función siguiente suplanta a la implantación predeterminada del método `onXML` en una aplicación de chat sencilla. La función `myOnXML` da instrucciones a la aplicación de chat para que reconozca un solo elemento XML, `MESSAGE`, con el formato siguiente:

```
<MESSAGE USER="John" TEXT="Hello, my name is John!" />.
```

El controlador `onXML` debe instalarse primero en el objeto XMLSocket como se muestra a continuación:

```
socket.onXML = myOnXML;
```

La función `displayMessage` se supone que es una función definida por el usuario que muestra el mensaje recibido al usuario.

```
function myOnXML(doc) {
  var e = doc.firstChild;
  if (e != null && e.nodeName == "MESSAGE") {
    displayMessage(e.attributes.user, e.attributes.text);
  }
}
```

Véase también

"function" a pagina 272

XMLSocket.send

Sintaxis

myXMLSocket.send(*objeto*);

Argumentos

objeto Un objeto XML u otros datos que se van a transmitir al servidor.

Descripción

Método; convierte el objeto XML o los datos especificados en el argumento *objeto* en una cadena y la transmite al servidor, seguida de un byte cero. Si *objeto* es un objeto XML, la cadena es la representación textual XML del objeto XML. La operación de envío es asíncrona, vuelve inmediatamente, pero los datos pueden transmitirse más tarde. El método `XMLSocket.send` no devuelve un valor que indica si los datos se transmitieron con éxito.

Si el objeto *myXMLSocket* no está conectado con el servidor (utilizando `XMLSocket.connect`), fallará la operación `XMLSocket.send`.

Reproductor

Flash 5 o posterior.

Ejemplo

El ejemplo siguiente muestra como podría especificar un nombre de usuario y una contraseña para enviar el objeto XML `myXML` al servidor.

```
var myXML = new XML();
var myLogin = myXML.createElement("login");
myLogin.attributes.username = usernameTextField;
myLogin.attributes.password = passwordTextField;
myXML.appendChild(myLogin);
myXMLSocket.send(myXML);
```

Véase también

"XMLSocket.connect" a pagina 413

_xmouse

Sintaxis

nombre_instancia._xmouse

Argumentos

nombre_instancia El nombre de una instancia de clip de película.

Descripción

Propiedad (de sólo lectura); devuelve la coordenada *x* de la posición del ratón.

Reproductor

Flash 5 o posterior.

Véase también

"Mouse (objeto)" a pagina 312
"_ymouse" a pagina 420

_xscale

Sintaxis

nombre_instancia._xscale
nombre_instancia._xscale = *porcentaje*;

Argumentos

porcentaje Un valor de porcentaje que especifica el porcentaje de escala horizontal de la película. El valor predeterminado es 100.

nombre_instancia El nombre de una instancia de clip de película.

Descripción

Propiedad; determina la escala horizontal (*porcentaje*) del clip de película que se aplica desde el punto de registro del clip de película. El punto de registro predeterminado es (0,0).

Cambiar la escala del sistema de coordenadas local afecta a la configuración de la propiedades _x e _y, que están definidas en píxeles completos. Por ejemplo, si se cambia la escala del clip de película principal al 50%, establecer la propiedad _x mueve un objeto en el clip de película la mitad del número de píxeles que lo haría si la película estuviera al 100%.

Reproductor

Flash 4 o posterior.

Véase también

"_xscale" a pagina 418

_y

Sintaxis

```
nombre_instancia._y
nombre_instancia._y = entero;
```

Argumentos

entero La coordenada local *y* del clip de película.

nombre_instancia El nombre de una instancia de clip de película.

Descripción

Propiedad; establece la coordenada *y* de la película en relación a las coordenadas locales del clip de película principal. Si un clip de película está en la Línea de tiempo principal, su sistema de coordenadas hace referencia a la esquina superior izquierda del Escenario como (0, 0). Si el clip de película se encuentra dentro de otro clip de película que tiene transformaciones, el clip de película está en el sistema de coordenadas local del clip de película que lo contiene. Así, para un clip de película girado 90 grados en sentido contrario a las agujas del reloj, el clip de película secundario hereda un sistema de coordenadas girado 90 grados en sentido contrario a las agujas del reloj. Las coordenadas del clip de película hacen referencia a la posición del punto de registro.

Reproductor

Flash 3 o posterior.

Véase también

"_yscale" a pagina 420

_ymouse

Sintaxis
nombre_instancia._ymouse

Argumentos
nombre_instancia El nombre de una instancia de clip de película.

Descripción
Propiedad (de sólo lectura); indica la coordenada *y* de la posición del ratón.

Reproductor
Flash 5 o posterior.

Véase también
"Mouse (objeto)" a pagina 312
"_xmouse" a pagina 418

_yscale

Sintaxis
nombre_instancia._yscale
nombre_instancia._yscale = *porcentaje*;

Argumentos
porcentaje Un valor de porcentaje que especifica el porcentaje de escala vertical de la película. El valor predeterminado es 100.

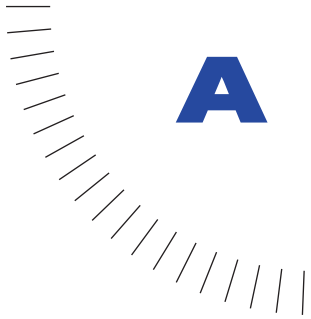
nombre_instancia El nombre de una instancia de clip de película.

Descripción
Propiedad; establece la escala vertical (*porcentaje*) del clip de película que se aplica desde el punto de registro del clip de película. El punto de registro predeterminado es (0,0).

Cambiar la escala del sistema de coordenadas local afecta a la configuración de la propiedades _x e _y, que están definidas en píxeles completos. Por ejemplo, si se cambia la escala del clip de película principal al 50%, establecer la propiedad _x mueve un objeto en el clip de película la mitad del número de píxeles que lo haría si la película estuviera al 100%.

Reproductor
Flash 4 o posterior.

Véase también
"_x" a pagina 392
"_y" a pagina 419



APÉNDICE A
Precedencia de operadores y asociatividad

Lista de operadores

Esta tabla enumera todos los operadores de Action Script y su asociatividad, desde la precedencia más alta a la más baja.

Operador	Descripción	Asociatividad
	Precedencia más alta	
+	Más unario	De derecha a izquierda
-	Menos unario	De derecha a izquierda
~	Complemento a uno como bit	De derecha a izquierda
!	NOT lógico	De derecha a izquierda
not	NOT lógico (estilo de Flash 4)	De derecha a izquierda
++	Incremento posterior	De izquierda a derecha
--	Decremento posterior	De izquierda a derecha
( )	Llamada a función	De izquierda a derecha

Operador	Descripción	Asociatividad
[ ]	Elemento de matriz	De izquierda a derecha
.	Miembro de una estructura	De izquierda a derecha
++	Incremento previo	De derecha a izquierda
--	Decremento previo	De derecha a izquierda
new	Asignar objeto	De derecha a izquierda
delete	Anular la asignación de objeto	De derecha a izquierda
typeof	Tipo de objeto	De derecha a izquierda
void	Devuelve un valor no definido	De derecha a izquierda
*	Multiplicación	De izquierda a derecha
/	División	De izquierda a derecha
%	Módulo	De izquierda a derecha
+	Suma	De izquierda a derecha
add	Concatenación de cadenas (anteriormente &)	De izquierda a derecha
-	Resta	De izquierda a derecha
«	Desplazamiento a la izquierda como bit	De izquierda a derecha
»	Desplazamiento a la derecha como bit	De izquierda a derecha

Operador	Descripción	Asociatividad
»»	Desplazamiento a la derecha como bit (sin signo)	De izquierda a derecha
<	Menor que	De izquierda a derecha
<=	Menor o igual que	De izquierda a derecha
>	Mayor que	De izquierda a derecha
>=	Mayor o igual que	De izquierda a derecha
lt	Menor que (versión para cadenas)	De izquierda a derecha
le	Menor o igual que (versión para cadenas)	De izquierda a derecha
gt	Mayor que (versión para cadenas)	De izquierda a derecha
ge	Mayor o igual que (versión para cadenas)	De izquierda a derecha
==	Igual	De izquierda a derecha
!=	Distinto	De izquierda a derecha
eq	Igual (versión para cadenas)	De izquierda a derecha
ne	Distinto (versión para cadenas)	De izquierda a derecha
&	AND como bit	De izquierda a derecha
^	XOR como bit	De izquierda a derecha
|	OR como bit	De izquierda a derecha

Operador	Descripción	Asociatividad
&&	AND lógico	De izquierda a derecha
and	AND lógico (estilo de Flash 4)	De izquierda a derecha
||	OR lógico	De izquierda a derecha
or	OR lógico (estilo de Flash 4)	De izquierda a derecha
?:	Condicional	De derecha a izquierda
=	Asignación	De derecha a izquierda
“*=, /=, %o=, +=, -=, &=, |=, ^=, ‹‹=, ››=, ›››=”	Asignación compuesta	De derecha a izquierda
,	Evaluación múltiple	De izquierda a derecha
	Precedencia más baja	

B

APÉNDICE B

Teclas del teclado y valores de códigos de tecla

Las tablas siguientes enumeran todas las teclas de un teclado estándar y sus valores de código de tecla correspondientes usados para identificar teclas en ActionScript. Si desea obtener más información, consulte la descripción del objeto Key en el Capítulo 7, “Diccionario de ActionScript.”

Letras de la A a la Z y números estándar del 0 al 9

Tecla de letra o número	Código de tecla
A	65
B	66
C	67
D	68
E	69
F	70
G	71
H	72
I	73
J	74
K	75
L	76
M	77
N	78
O	79
P	80
Q	81
R	82
S	83
T	84

Tecla de letra o número	Código de tecla
U	85
V	86
W	87
X	88
Y	89
Z	90
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57

Teclas del teclado numérico

Tecla del teclado numérico	Código de tecla
0 del teclado numérico	96
1 del teclado numérico	97
2 del teclado numérico	98
3 del teclado numérico	99
4 del teclado numérico	100
5 del teclado numérico	101
6 del teclado numérico	102
7 del teclado numérico	103
8 del teclado numérico	104
9 del teclado numérico	105
Multiplicar	106
Sumar	107
Intro	108
Restar	109
Decimal	110
Dividir	111

Teclas de función

Tecla de función	Código de tecla
F1	112
F2	113
F3	114
F4	115
F5	116
F6	117
F7	118
F8	119
F9	120
F10	121
F11	122
F12	123

Otras teclas

Tecla	Código de tecla
Retroceso	8
Tabulador	9
Borrar	12
Intro	13
Mayús	16
Control	17
Alt	18
Bloq Mayús	20
Esc	27
Barra espaciadora	32
Re Pág	33
Av Pág	34
Fin	35
Inicio	36
Flecha izquierda	37
Flecha arriba	38

Tecla	Código de tecla
Flecha derecha	39
Flecha abajo	40
Insert	45
Supr	46
Ayuda	47
Bloq Num	144
; :	186
= +	187
- _	189
/ ?	191
` ~	192
[{	219
\ |	220
] }	221
‘ “	222

APÉNDICE C
Mensajes de error

La tabla siguiente contiene una lista de los mensajes de error que devuelve el compilador de Flash. Se incluye una explicación de cada mensaje para ayudarle a solucionar los problemas en sus archivos de película.

Mensaje de error	Descripción
La propiedad ‹propiedad› no existe	Se ha encontrado una propiedad que no existe. Por ejemplo, `x = _green` no es válida ya que no existe la propiedad `_green`.
El operador ‹operador› debe ir seguido de un operando	Se ha encontrado un operador sin un operando. Por ejemplo, `x = 1 +` requiere un operando tras el operador +. Un operando no válido sigue a un operador. Por ejemplo, `trace(1+);` es sintácticamente incorrecto.
Error de sintaxis	Este mensaje aparece cuando se encuentra un error de sintaxis no específico.
Se esperaba un nombre de campo después del operador '.'	Debe especificar un nombre de campo válido cuando se utiliza la sintaxis *`object.field`*.

Se esperaba ‹token›	Se ha encontrado un token no válido o inesperado. Por ejemplo, en la sintaxis que se muestra a continuación, el token `foo` no es válido. El token que se esperaba es `while`.
	`do {`
	`  trace (i)`
	`} foo (i < 100)`

‹terminador› debe finalizar la lista del iniciador	A una lista de inicializador de objeto o de matriz le falta el signo] o el signo } de cierre.

Mensaje de error	Descripción
Se esperaba un identificador	Se ha encontrado un token no esperado en lugar de un identificador. En el ejemplo siguiente, 3 no es un identificador válido. `var 3 = 4;`
No se admite la construcción de JavaScript '<construcción>'	Se ha encontrado una construcción de JavaScript no admitida por ActionScript. Este mensaje aparece si se utilizan cualquiera de las siguientes construcciones de JavaScript: `void`, `switch`, `try`, `catch` o `throw`.
El margen izquierdo del operador de asignación debe ser una variable o propiedad	Se ha utilizado un operador de asignación, pero el margen izquierdo de la asignación no era una variable o propiedad legal.
'}' debe finalizar el bloque de declaración	Se ha declarado un grupo de declaraciones dentro de llaves, pero falta la llave de cierre.
Se esperaba un evento	Se ha declarado un controlador `On(MouseEvent)` o `onClipEvent`, pero no se ha especificado ningún evento, o se ha encontrado un token inesperado donde debería aparecer un evento.
Evento no válido	Este script contiene un evento de ratón o de clip no válido. Si desea ver una lista de los eventos de ratón y de clip válidos, consulte las entradas de On(MouseEvent) y OnClipEvent en el capítulo correspondiente al diccionario de ActionScript.
Se esperaba un código de tecla	Es necesario especificar un código de tecla. Consulte el Apéndice B para ver una lista de los códigos de tecla.
Código de tecla no válido	El código de tecla especificado no existe.
Rastro de basura encontrado	El script o la expresión se analizó correctamente pero contenía caracteres finales adicionales que no se pudieron analizar.
Función ilegal	Se ha utilizado una declaración de función con nombre como una expresión. Las declaraciones de función con nombre deben ser sentencias. Válida: `function sqr (x) { return x * x; }` No válida: `var v = function sqr (x) { return x * x; }`
Se espera nombre de función	El nombre especificado para esta función no es un nombre de función válido.

Mensaje de error	Descripción
Se espera nombre de parámetro	Se esperaba un nombre de parámetro (argumento) en una declaración de función, pero se ha encontrado un token no esperado.
Se ha encontrado un operador 'else' que no coincide con 'if'	Se ha encontrado una sentencia `else`, pero no aparecía `if` delante de ella. Puede utilizar `else` solamente junto con una sentencia `if`.
Error de tipo de escena	El argumento de escena de una acción `gotoAndPlay`, `gotoAndStop`, o `ifFrameLoaded` era de tipo incorrecto. El argumento de escena debe ser una constante de cadena.
Error interno	Ha ocurrido un error interno en el compilador de ActionScript. Por favor, envíe el archivo FLA que ha generado este error a Macromedia, junto con instrucciones detalladas sobre como reproducir el mensaje.
Se esperan dígitos hexadecimales después de 0x	Se ha encontrado la secuencia 0x, pero la secuencia no estaba seguida por dígitos hexadecimales válidos.
Error al abrir el archivo #include	Se produjo un error al abrir un archivo incluido con la directiva `include`. El error puede haberse producido debido a que el archivo no estaba presente o debido a un error de disco.
Directiva #include mal formada	Una directiva `include` no se ha escrito correctamente. Una directiva `include` debe utilizar la sintaxis siguiente: `#include "somefile.as"`
No se ha finalizado el comentario multilínea	A un comentario multilínea que comenzaba con `/*` le falta la etiqueta `*/` de cierre.
No se ha finalizado correctamente el literal de cadena	A un literal de cadena que comenzaba con un signo de interrogación de apertura (sencilla o doble) le falta el signo de interrogación de cierre.
La función ‹function› necesita ‹count› parámetros	Se ha llamado a una función, pero se han encontrado un número de parámetros no esperados.
Se esperaba nombre de propiedad en GetProperty	Se ha llamado a la función `getProperty`, pero el segundo argumento no era el nombre de una propiedad de un clip de película.
El parámetro ‹parameter› no puede declararse varias veces	Un nombre de parámetro ha aparecido varias veces en la lista de parámetros de una declaración de función. Todos los nombres de parámetros deben ser únicos.

Mensaje de error	Descripción
La variable ‹variable› no puede declararse varias veces	Ha aparecido un nombre de variable varias veces en una sentencia `var`. Todos los nombres de variable de una sola sentencia `var` deben ser únicos.
Los controladores 'on' no pueden anidar dentro de otros controladores 'on'	Se ha declarado un controlador `on` dentro de otro controlador `on`. Todos los controladores `on` deben aparecer en el nivel superior de una lista de acciones.
La declaración debe aparecer dentro del controlador on	En las acciones para una instancia de botón, se ha declarado una sentencia sin un bloque `on` que la rodee. Todas las acciones de una instancia de botón deben aparecer dentro de un bloque `on`.
La declaración debe aparecer dentro de un controlador onClipEvent	En las acciones para una instancia de clip de película, se ha declarado una sentencia sin un bloque `onClipEvent` que la rodee. Todas las acciones para una instancia de clip de película deben aparecer dentro de un bloque `onClipEvent`.
Sólo se permiten los eventos de ratón para las instancias de botón	Se ha declarado un controlador de evento de botón en una lista de acciones de fotograma o en una lista de acciones de una instancia de clip. Los eventos de botón se permiten solamente en las listas de acciones de instancias de botón.
Sólo se permiten los eventos de clip para instancias de clip de película	Se ha declarado un controlador de evento de clip en una lista de acciones de fotograma o en una lista de acciones de una instancia de botón. Los eventos de clip se permiten solamente en las listas de acciones de las instancias de clip de película.

ÍNDICE ALFABÉTICO

A

B

C

D

R

U

V

W

X