

REFERENCIA DEL LENGUAJE ACTIONSCRIPT™ DE FLASH® LITE™ 1.x

© 2007 Adobe Systems Incorporated. Todos los derechos reservados.

Referencia del lenguaje ActionScript™ de Adobe® Flash® Lite™ 1.x

Si este manual se distribuye con un software que contenga un contrato para usuarios finales, el manual, así como el software que en él se describe, se suministran bajo licencia y pueden utilizarse y copiarse según lo dispuesto en los términos de dicha licencia. Salvo en los casos permitidos por dicha licencia, queda prohibida la reproducción de cualquier parte de este manual, su almacenamiento en un sistema de recuperación, su transmisión (de cualquier forma o modo) electrónica, mecánica o mediante grabación, sin el consentimiento previo por escrito de Adobe Systems Incorporated. Tenga en cuenta que el contenido de este manual está protegido por las leyes de copyright, incluso si no se distribuye con el software que incluye el contrato de licencia para usuarios finales.

El contenido de este manual se suministra exclusivamente con fines informativos, está sujeto a cambios sin previo aviso y no debe entenderse como ningún compromiso por parte de Adobe Systems Incorporated. Adobe Systems Incorporated declina toda responsabilidad por los posibles errores o imprecisiones que puedan aparecer en el contenido informativo de este manual.

Recuerde que las ilustraciones o las imágenes que tal vez quiera incluir en su proyecto pueden estar protegidas por las leyes de copyright. La incorporación no autorizada de este material en nuevos trabajos podría suponer una violación de los derechos del propietario del copyright. Asegúrese de haber obtenido los permisos necesarios del propietario del copyright.

Cualquier referencia a nombres de empresas en las plantillas de muestras se incluyen únicamente con fines de demostración y no pretenden hacer mención directa a ninguna empresa real.

Adobe, el logotipo de Adobe, Macromedia, Dreamweaver y Flash son marcas comerciales o marcas comerciales registradas de Adobe Systems Incorporated en los Estados Unidos y en otros países.

Información de terceros

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



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

Fraunhofer-IIS/Thomson Multimedia: tecnología de compresión de audio MPEG Layer-3 con licencia de Fraunhofer IIS y Thomson Multimedia (<http://www.iis.fhg.de/amm/>).

Independent JPEG Group: este software está basado en parte del trabajo del Independent JPEG Group.

Nellymoser, Inc.: tecnología de compresión y descompresión de voz con licencia de Nellymoser, Inc. (<http://www.nelly-moser.com>).

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

Macromedia Flash 8 utiliza tecnología de vídeo de On2 TrueMotion. © 1992-2005 On2 Technologies, Inc. Todos los derechos reservados. <http://www.on2.com>.

Visual SourceSafe es una marca registrada o una marca comercial de Microsoft Corporation en los Estados Unidos y en otros países.

Información actualizada e información adicional de códigos de otros fabricantes disponible en http://www.adobe.com/go/thirdparty_es/.

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, EE.UU.

Aviso para los usuarios finales del gobierno de EE.UU. Este Software y su Documentación son "Artículos comerciales", según se define este término en 48 C.F.R. §2.101, y consta de "Software comercial informático" y "Documentación de software comercial informático", según se definen estos términos en 48 C.F.R. §12.212 o 48 C.F.R. §227.7202. En coherencia con lo descrito en 48 C.F.R. §12.212 o 48 C.F.R. de §227.7202-1 a 227.7202-4, según sea aplicable, el Software comercial informático y la Documentación de software comercial informático se otorgan bajo licencia a los usuarios finales del gobierno de EE.UU. (a) únicamente como artículos comerciales y (b) únicamente con los derechos garantizados al resto de usuarios finales de acuerdo con lo estipulado en sus términos y condiciones. Los derechos no publicados quedan reservados al amparo de las leyes de copyright de los Estados Unidos. Adobe Systems Incorporated, 345 Park Avenue, San Jose, CA 95110-2704, EE.UU. Para los usuarios finales del gobierno de EE.UU., Adobe acuerda cumplir todas las leyes de igualdad de oportunidades aplicables, incluyendo (si procede) lo estipulado en la Executive Order 11246, corregida, en la sección 402 de la Vietnam Era Veterans Readjustment Assistance Act de 1974 (38 USC 4212) y en la sección 503 de la Rehabilitation Act de 1973, corregida, así como las normativas de 41 CFR partes 60-1 a 60-60, 60-250 y 60-741. La cláusula de acción afirmativa y las normativas incluidas en la frase anterior se pueden incluir como referencia.

Contenido

Introducción	11
Entrada de muestra para la mayoría de los elementos de ActionScript	11
Carpeta Samples	12
Convenciones tipográficas	12
Capítulo 1: Funciones globales de Flash Lite	13
call()	16
chr()	17
duplicateMovieClip()	17
eval ()	18
getProperty()	19
getTimer()	20
getURL()	21
gotoAndPlay()	24
gotoAndStop()	25
ifframeLoaded()	26
int()	27
length()	27
loadMovie()	28
loadMovieNum()	29
loadVariables()	31
loadVariablesNum()	32
mbchr()	33
mblength()	34
mbord()	35
mbsubstring()	35
nextFrame()	36
nextScene()	37
Number()	37
on()	38
ord()	39
play()	40
prevFrame()	40
prevScene()	41
random()	42

removeMovieClip()	43
set()	43
setProperty()	44
stop()	45
stopAllSounds()	46
String()	46
substring()	47
tellTarget()	48
toggleHighQuality()	49
trace()	49
unloadMovie()	50
unloadMovieNum()	51
Capítulo 2: Propiedades de Flash Lite	53
/ (Barra diagonal)	55
_alpha	56
_currentframe	56
_focusrect	57
_framesloaded	58
_height	58
_highquality	59
_level	60
maxscroll	61
_name	61
_rotation	62
scroll	62
_target	63
_totalframes	63
_visible	64
_width	65
_x	65
_xscale	66
_y	67
_yscale	68
Capítulo 3: Sentencias de Flash Lite	69
break	70
case	71
continue	72
do..while	74
else	74
else if	75
for	76
if	77

switch	78
while	80
Capítulo 4: Operadores de Flash Lite	83
add (concatenación de cadenas)	86
+= (asignación de suma)	87
and	87
= (asignación)	88
/* (comentario en bloque)	89
, (coma)	90
// (comentario)	91
?: (condicional)	92
-- (decremento)	92
/ (dividir)	93
/= (asignación de división)	94
. (punto)	94
++ (incremento)	95
&& (AND lógico)	96
! (NOT lógico)	97
(OR lógico)	98
% (módulo)	99
%= (asignación de módulo)	100
*= (asignación de multiplicación)	100
* (multiplicar)	101
+ (suma numérica)	102
== (igualdad numérica)	103
> (numérico mayor que)	103
>= (numérico mayor o igual que)	104
<> (desigualdad numérica)	105
< (numérico menor que)	105
<= (numérico menor o igual que)	106
() (paréntesis)	107
" " (delimitador de cadena)	108
eq (igualdad de cadenas)	108
gt (cadena mayor que)	109
ge (cadena mayor o igual que)	110
ne (desigualdad de cadenas)	111
lt (cadena menor que)	111
le (cadena menor o igual que)	112
- (resta)	113
-= (asignación de resta)	114

Capítulo 5: Elementos del lenguaje específicos de Flash Lite	117
Capacidades	121
_capCompoundSound	121
_capEmail	121
_capLoadData	122
_capMFi	123
_capMIDI	123
_capMMS	124
_capMP3	125
_capSMAF	125
_capSMS	126
_capStreamSound	127
_cap4WayKeyAS	127
\$version	128
fscommand()	129
Launch	129
fscommand2()	130
Escape	132
FullScreen	132
GetBatteryLevel	133
GetDateDay	134
GetDateMonth	134
GetDateWeekday	135
GetDateYear	136
GetDevice	136
GetDeviceID	138
GetFreePlayerMemory	138
GetLanguage	139
GetLocaleLongDate	141
GetLocaleShortDate	142
GetLocaleTime	143
GetMaxBatteryLevel	143
GetMaxSignalLevel	144
GetMaxVolumeLevel	144
GetNetworkConnectStatus	145
GetNetworkName	146
GetNetworkRequestStatus	147
GetNetworkStatus	149
GetPlatform	150
GetPowerSource	151
GetSignalLevel	151
GetTimeHours	152
GetTimeMinutes	152
GetTimeSeconds	153

GetTimeZoneOffset	154
GetTotalPlayerMemory	155
GetVolumeLevel	155
Quit	156
ResetSoftKeys.....	156
SetInputTextType	157
SetQuality	158
SetSoftKeys.....	159
StartVibrate	160
StopVibrate	160
Unescape	161
Índice alfabético.....	163

Introducción

En este manual se describen la sintaxis y el uso de elementos de ActionScript para desarrollar aplicaciones para el software Macromedia® Flash® Lite™ 1.0 y Macromedia® Flash® Lite™ 1.1, denominados de forma conjunta Flash Lite 1.x. Flash Lite 1.x ActionScript se basa en la versión de ActionScript utilizada en Macromedia® Flash® 4 de Adobe. Para usar ejemplos en un script, copie el ejemplo de código de este manual y péguelo en el panel Script o en un archivo de script externo. En este manual se describen todos los elementos de ActionScript: operadores, palabras clave, sentencias, comandos, propiedades, funciones, clases y métodos.

Entrada de muestra para la mayoría de los elementos de ActionScript

El ejemplo siguiente explica las convenciones utilizadas para la mayoría de los elementos de ActionScript.

Título de la entrada

Las entradas de cada capítulo están ordenadas alfabéticamente. El orden alfabético no distingue entre mayúsculas y minúsculas, pasa por alto los signos de subrayado iniciales, etc.

Disponibilidad

A no ser que se indique lo contrario, en la sección Disponibilidad se especifica qué versiones de Flash Lite admiten el elemento.

Uso

En esta sección se proporciona la sintaxis correcta para utilizar el elemento de ActionScript en el código. La parte necesaria de la sintaxis está en *fuentes de código*. El código que introduce y la información de tipo de datos están en *fuentes de código cursiva*. Los tipos de datos se distinguen del código que se introduce por los dos puntos (:) situados al principio. Los corchetes ([]) indican parámetros opcionales.

Operandos

En esta sección se describen los parámetros listados en la sintaxis.

Descripción

En esta sección se identifica el tipo de elemento (por ejemplo, operador, función, etc.) y los valores que devuelve, y se describe cómo utilizar el elemento.

Ejemplo

En esta sección aparece un ejemplo de código que muestra cómo utilizar el elemento.

Véase también

En esta sección se muestra una lista de las entradas del diccionario de ActionScript relacionadas.

Carpeta Samples

Para obtener ejemplos de proyectos completos de Flash Lite projects con código ActionScript ejecutable, consulte la página de tutoriales y ejemplos de Flash Lite: www.adobe.com/go/learn_ftl_samples_and_tutorials_es. Busque el archivo .zip para su versión de ActionScript, descargue y descomprima el archivo y, posteriormente, acceda a la carpeta Samples para utilizar los archivos de ejemplo.

Convenciones tipográficas

En este manual se utilizan las siguientes convenciones tipográficas:

- La *fuentes en cursiva* indica un valor que se debe sustituir (por ejemplo, en una ruta de carpeta).
- La *fuentes para código* indica que se trata de código de ActionScript.
- La *fuentes para código en cursiva* indica un parámetro de ActionScript.
- La **fuentes en negrita** indica una entrada de caracteres.
- Las comillas dobles (" ") en los ejemplos de código indican cadenas delimitadas. Sin embargo, los programadores pueden utilizar también comillas simples.

Funciones globales de Flash Lite

En esta sección se describe la sintaxis y el uso de las funciones globales de ActionScript de Macromedia Flash Lite 1.1 de Adobe. Incluye las funciones siguientes:

Función	Descripción
<code>call()</code>	Ejecuta el script en el fotograma llamado sin mover la cabeza lectora a ese fotograma.
<code>chr()</code>	Convierte números de código ASCII en caracteres.
<code>duplicateMovieClip()</code>	Crea una instancia de un clip de película durante la reproducción del archivo SWF.
<code>eval()</code>	Accede a las variables, propiedades, objetos o clips de película por su nombre.
<code>getProperty()</code>	Devuelve el valor de la propiedad especificada para el clip de película especificado.
<code>getTimer()</code>	Devuelve el número de milisegundos transcurridos desde que se inició la reproducción del archivo SWF.
<code>getURL()</code>	Carga un documento de una URL específica en una ventana o pasa variables a otra aplicación en una URL definida.
<code>gotoAndPlay()</code>	Envía la cabeza lectora al fotograma especificado en una escena y comienza la reproducción desde dicho fotograma. Si no se especifica ninguna escena, la cabeza lectora se desplaza al fotograma especificado de la escena actual.
<code>gotoAndStop()</code>	Envía la cabeza lectora al fotograma especificado en una escena y la detiene. Si no se especifica ninguna escena, la cabeza lectora se envía al fotograma de la escena actual.
<code>ifFrameLoaded()</code>	Comprueba si el contenido de un fotograma específico está disponible localmente.
<code>int()</code>	Trunca un número decimal a un valor entero.
<code>length()</code>	Devuelve el número de caracteres de la cadena o el nombre de variable especificado.

Función	Descripción
<code>loadMovie()</code>	Carga un archivo SWF en Flash Lite durante la reproducción del archivo SWF original.
<code>loadMovieNum()</code>	Carga un archivo SWF en un nivel de Flash Lite durante la reproducción del archivo SWF que se cargó originalmente.
<code>loadVariables()</code>	Lee datos de un archivo externo, como un archivo de texto o texto generado por un script de ColdFusion, CGI ASP, PHP o Perl, y establece los valores de las variables en un nivel de Flash Lite. Esta función puede además actualizar las variables del archivo SWF activo con nuevos valores.
<code>loadVariablesNum()</code>	Lee datos de un archivo externo, como un archivo de texto o texto generado por un script de ColdFusion, CGI, ASP, PHP o Perl, y establece los valores de las variables en un nivel de Flash Lite. Esta función puede además actualizar las variables del archivo SWF activo con nuevos valores.
<code>mbchr()</code>	Convierte un número de código ASCII en un carácter multibyte.
<code>mbLength()</code>	Devuelve la longitud de la cadena de caracteres multibyte.
<code>mbord()</code>	Convierte el carácter especificado en un número multibyte.
<code>mbsubstring()</code>	Extrae una cadena de caracteres multibyte nueva de una cadena de caracteres multibyte.
<code>nextFrame()</code>	Traslada la cabeza lectora al siguiente fotograma y la detiene en dicho lugar.
<code>nextScene()</code>	Envía la cabeza lectora al fotograma 1 de la siguiente escena y la detiene.
<code>Number()</code>	Convierte una expresión en un número y devuelve un valor.
<code>on()</code>	Especifica el evento de usuario o la pulsación de tecla que activa un evento.
<code>ord()</code>	Convierte caracteres en números de código ASCII.
<code>play()</code>	Mueve la cabeza lectora hacia delante en la línea de tiempo.
<code>prevFrame()</code>	Traslada la cabeza lectora al fotograma anterior y la detiene en dicho lugar. Si el fotograma actual es el fotograma 1, la cabeza lectora no se mueve.
<code>prevScene()</code>	Envía la cabeza lectora al fotograma 1 de la escena anterior y la detiene.
<code>removeMovieClip()</code>	Elimina el clip de película que se creó originalmente mediante <code>duplicateMovieClip()</code> .
<code>set()</code>	Asigna un valor a una variable.

Función	Descripción
<code>setProperty()</code>	Cambia un valor de propiedad de un clip de película durante la reproducción de la película.
<code>stop()</code>	Detiene el archivo SWF que se está reproduciendo.
<code>stopAllSounds()</code>	Detiene todos los sonidos que se están reproduciendo en un archivo SWF sin detener la cabeza lectora.
<code>String()</code>	Devuelve una representación de cadena del parámetro especificado.
<code>substring()</code>	Extrae parte de una cadena.
<code>tellTarget()</code>	Aplica las instrucciones especificadas en el parámetro <i>statement(s)</i> a la línea de tiempo especificada en el parámetro <i>target</i> .
<code>toggleHighQuality()</code>	Activa y desactiva el suavizado en Flash Lite. El suavizado alisa los bordes de los objetos, pero ralentiza la reproducción del SWF.
<code>trace()</code>	Evalúa la expresión y muestra el resultado en el panel Salida en modo de prueba.
<code>unloadMovie()</code>	Elimina de Flash Lite un clip de película que se cargó mediante <code>loadMovie()</code> , <code>loadMovieNum()</code> o <code>duplicateMovieClip()</code> .
<code>unloadMovieNum()</code>	Elimina de Flash Lite un clip de película que se cargó mediante <code>loadMovie()</code> , <code>loadMovieNum()</code> o <code>duplicateMovieClip()</code> .

call()

Disponibilidad

Flash Lite 1.0.

Uso

```
call(frame)
```

```
call(movieClipInstance: frame)
```

Operandos

frame La etiqueta o el número de un fotograma en la línea de tiempo.

movieClipInstance Nombre de instancia de un clip de película.

Descripción

Función; ejecuta el script en el fotograma llamado sin desplazar la cabeza lectora a ese fotograma. Las variables locales no existen después de ejecutar el script. La función `call()` puede funcionar de dos formas:

- De forma predeterminada, ejecuta el script en el fotograma especificado en la misma línea de tiempo en la que se ejecutó la función `call()`, sin desplazar la cabeza lectora a ese fotograma.
- La instancia de clip especificada ejecuta el script en el fotograma especificado de la instancia de clip de película, sin desplazar la cabeza lectora a ese fotograma.

NOTA

La función `call()` funciona de forma síncrona; el código ActionScript que siga a la función `call()` no se ejecuta hasta que haya finalizado todo el código ActionScript del fotograma especificado.

Ejemplo

Los ejemplos siguientes ejecutan el script del fotograma `myScript`:

```
// para ejecutar funciones en el fotograma con la etiqueta "myScript"  
thisFrame = "myScript";  
trace ("Calling the script in frame: " + thisFrame);
```

```
// para ejecutar funciones de otros fotogramas en la misma línea de tiempo  
call("myScript");
```

chr()

Disponibilidad

Flash Lite 1.0.

Uso

`chr(number)`

Operandos

number Un número de código ASCII.

Descripción

Función String; convierte números de código ASCII en caracteres.

Ejemplo

El ejemplo siguiente convierte el número 65 en la letra *A* y asigna dicha letra a la variable `myVar`:

```
myVar = chr(65);  
trace (myVar); // Salida: A
```

duplicateMovieClip()

Disponibilidad

Flash Lite 1.0.

Uso

`duplicateMovieClip(target, newname, depth)`

Operandos

target La ruta de destino del clip de película que se va a duplicar.

newname Identificador exclusivo del clip de película duplicado.

depth Nivel de profundidad exclusivo del clip de película duplicado. El nivel de profundidad indica un orden de apilamiento de los clips de película duplicados. Este orden de apilamiento es similar al de las capas de la línea de tiempo; los clips de película con un nivel de profundidad inferior se ocultan bajo los clips con un nivel de profundidad superior. Debe asignar a cada clip de película duplicado un nivel de profundidad exclusivo para que no sobrescriba los clips de película en niveles de profundidad ocupados.

Descripción

Función; crea una instancia de un clip de película durante la reproducción del archivo SWF. No devuelve nada. En los clips de película duplicados, la cabeza lectora siempre empieza en el fotograma 1, independientemente del lugar en el que se encuentre la cabeza lectora del clip de película original (principal). Las variables del clip de película principal no se copian en el clip de película duplicado. Si se elimina el clip de película principal, el clip de película duplicado también se elimina. Utilice la función o el método `removeMovieClip()` para eliminar una instancia de clip de película creada mediante `duplicateMovieClip()`. Para hacer referencia al nuevo clip de película, use la cadena pasada como operando *newname*.

Ejemplo

El ejemplo siguiente duplica un clip de película denominado `originalClip` para crear un nuevo clip con el nombre `newClip`, en un nivel de profundidad de 10. La posición *x* del nuevo clip se define en 100 píxeles.

```
duplicateMovieClip("originalClip", "newClip", 10);
setProperty("newClip", _x, 100);
```

El ejemplo siguiente utiliza `duplicateMovieClip()` en un bucle `for` para crear varios clips de película de forma simultánea. Una variable `index` controla la máxima profundidad de apilamiento ocupada. Cada nombre de clip de película duplicado contiene un sufijo numérico que corresponde a su profundidad de apilamiento (`clip1`, `clip2`, `clip3`).

```
for (i = 1; i <= 3; i++) {
    newName = "clip" + i;
    duplicateMovieClip("originalClip", newName); }
```

Véase también

[removeMovieClip\(\)](#)

eval ()

Disponibilidad

Flash Lite 1.0.

Uso

```
eval(expression)
```

Operandos

expression Una cadena que contiene el nombre de una variable, propiedad, objeto o clip de película que debe recuperarse.

Descripción

Función; accede a las variables, propiedades, objetos o clips de película por su nombre.

Si *expression* es una variable o una propiedad, se devuelve el valor de la variable o propiedad. Si *expression* es un objeto o clip de película, se devuelve una referencia al objeto o clip de película. Si no se encuentra el elemento denominado en *expression*, se devuelve `undefined`.

Puede utilizar `eval()` para simular matrices o para definir y recuperar de forma dinámica el valor de una variable.

Ejemplo

El ejemplo siguiente utiliza `eval()` para determinar el valor de la expresión "piece" + x.

El resultado es un nombre de variable, `piece3`, `eval()` devuelve el valor de la variable y lo asigna a y:

```
piece3 = "dangerous";
x = 3;
y = eval("piece" + x);
trace(y); // Salida: dangerous.
```

El ejemplo siguiente muestra cómo se puede simular una matriz:

```
name1 = "mike";
name2 = "debbie";
name3 = "logan";
for(i = 1; i <= 3; i++) {
    trace (eval("name" + i)); // Salida: mike, debbie, logan
}
```

getProperty()

Disponibilidad

Flash Lite 1.0.

Uso

```
getProperty(my_mc, property)
```

Operandos

my_mc El nombre de instancia del clip de película para el que se recupera la propiedad.

property Propiedad de un clip de película.

Descripción

Función; devuelve el valor de la propiedad especificada para el clip de película *my_mc*.

Ejemplo

El ejemplo siguiente recupera la coordenada del eje horizontal (`_x`) para el clip de película `my_mc` en la línea de tiempo raíz de la película:

```
xPos = getProperty("person_mc", _x);  
trace (xPos); // salida: -75
```

Véase también

[setProperty\(\)](#)

getTimer()

Disponibilidad

Flash Lite 1.0.

Uso

```
getTimer()
```

Operandos

Ninguno.

Descripción

Función; devuelve el número de milisegundos transcurridos desde que se inició la reproducción del archivo SWF.

Ejemplo

El ejemplo siguiente define la variable `timeElapsed` como el número de milisegundos transcurridos desde que comenzó a reproducirse el archivo SWF:

```
timeElapsed = getTimer();  
trace (timeElapsed); // Salida: milisegundos que se ha estado  
                        // reproduciendo una película
```

getURL()

Disponibilidad

Flash Lite 1.0.

Uso

```
getURL(url [ , window [ , "variables" ] ] )
```

Operandos

url URL del cual se obtiene el documento.

window Parámetro opcional que especifica la ventana o marco HTML donde debería cargarse el documento.

NOTA

El parámetro *window* no se especifica para aplicaciones de Flash Lite, ya que los navegadores de los teléfonos móviles no admiten varias ventanas.

Puede introducir una cadena vacía o el nombre de una ventana específica, o bien seleccionarlo entre los siguientes nombres de destino reservados:

- `_self` especifica el fotograma actual en la ventana actual.
- `_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 método GET o POST para enviar variables. Si no hay ninguna variable, omite este parámetro. El método GET añade las variables al final de la URL y se utiliza para números reducidos de variables. El método POST envía las variables en un encabezado HTTP independiente y se utiliza para enviar cadenas de variable largas.

Descripción

Función; carga un documento de una URL específica en una ventana o pasa variables a otra aplicación en una URL definida. Para probar esta función, 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`), se necesita una conexión de red.

Flash Lite 1.0 sólo reconoce los protocolos HTTP, HTTPS, mailto y tel. Flash Lite 1.1 reconoce estos protocolos y además protocolos file, SMS (del inglés, short message service) y MMS (del inglés, multimedia message service).

Flash Lite pasa la llamada al sistema operativo, y éste gestiona la llamada con la aplicación predeterminada registrada para el protocolo especificado.

Sólo se procesa una función `getURL()` por fotograma o por controlador de evento.

Algunos teléfonos limitan la función `getURL()` a eventos de pulsación de tecla únicamente, en cuyo caso la llamada `getURL()` se procesa solamente si es activada por un controlador de evento de tecla. Incluso en este caso, sólo se procesa una función `getURL()` por controlador de evento.

Ejemplo

En el siguiente código ActionScript, el reproductor Flash Lite abre `mobile.example.com` en el navegador predeterminado:

```
myURL = "http://mobile.example.com";
on(keyPress "1") {
    getURL(myURL);
}
```

También puede utilizar `GET` o `POST` para enviar variables desde la línea de tiempo actual.

El ejemplo siguiente utiliza el método `GET` para añadir variables a una URL:

```
firstName = "Gus";
lastName = "Richardson";
age = 92;
getURL("http://www.example.com", "_blank", "GET");
```

El código ActionScript siguiente utiliza `POST` para enviar variables en un encabezado HTTP:

```
firstName = "Gus";
lastName = "Richardson";
age = 92;
getURL("http://www.example.com", "POST");
```

Se puede asignar un botón de acción para abrir una ventana de correo electrónico con los campos de texto de dirección, asunto y cuerpo (`address`, `subject` y `body`) rellenos. Utilice uno de los métodos siguientes para asignar un botón de función: Utilice el Método 1 para Shift-JIS o codificación de caracteres en inglés; el Método 2 sólo para codificación de caracteres en inglés.

Método 1. Establezca las variables de cada uno de los parámetros, como en este ejemplo:

```
on (release, keyPress "#"){
    subject = "email subject";
    body = "email body";
    getURL("mailto:somebody@anywhere.com", "", "GET");
}
```

Método 2: Defina cada parámetro en la acción `getURL()` como en este ejemplo:

```
on (release, keyPress "#"){
    getURL("mailto:somebody@anywhere.com?cc=cc@anywhere.com&bcc=bcc@anywhere.com&subject=I am the email subject&body=I am the email body");
}
```

El Método 1 produce la codificación de URL automática y el Método 2 conserva los espacios en las cadenas. Por ejemplo, las cadenas que se obtienen si se usa el Método 1 son las siguientes:

```
email+subject  
email+body
```

En cambio, con el Método 2 el resultado sería:

```
email subject  
email body
```

El ejemplo siguiente emplea el protocolo tel:

```
on (release, keyPress "#"){  
    getURL("tel:117");  
}
```

En el ejemplo siguiente, se utiliza `getURL()` para enviar un mensaje SMS:

```
mySMS = "sms:4156095555?body=sample sms message";  
on(keyPress "5") {  
    getURL(mySMS);  
}
```

En el ejemplo siguiente, se utiliza `getURL()` para enviar un mensaje MMS:

```
// ejemplo de mms  
myMMS = "mms:4156095555?body=sample mms message";  
on(keyPress "6") {  
    getURL(myMMS);  
}
```

En el ejemplo siguiente, se utiliza `getURL()` para abrir un archivo de texto almacenado en el sistema de archivos local:

```
// ejemplo de protocolo file  
filePath = "file:///c:/documents/flash/myApp/myvariables.txt";  
on(keyPress "7") {  
    getURL(filePath);  
}
```

gotoAndPlay()

Disponibilidad

Flash Lite 1.0.

Uso

```
gotoAndPlay([scene,] frame)
```

Operandos

scene Cadena opcional que especifica el nombre de la escena donde se envía la cabeza lectora.

frame Número que representa el número de fotograma o la cadena que representa la etiqueta del fotograma al que se envía la cabeza lectora.

Descripción

Función; envía la cabeza lectora al fotograma especificado en una escena y comienza la reproducción desde dicho fotograma. Si no se especifica ninguna escena, la cabeza lectora se desplaza al fotograma especificado de la escena actual.

Puede utilizar el parámetro *scene* únicamente en la línea de tiempo raíz, no en las líneas de tiempo de los clips de película u otros objetos del documento.

Ejemplo

En el ejemplo siguiente, cuando el usuario hace clic en un botón al que se ha asignado `gotoAndPlay()`, la cabeza lectora se desplaza al fotograma 16 de la escena actual y comienza la reproducción del archivo SWF:

```
on(keyPress "7") {  
    gotoAndPlay(16);  
}
```

gotoAndStop()

Disponibilidad

Flash 1.0.

Uso

```
gotoAndStop([scene,] frame)
```

Operandos

scene Cadena opcional que especifica el nombre de la escena donde se envía la cabeza lectora.

frame Número que representa el número de fotograma o la cadena que representa la etiqueta del fotograma al que se envía la cabeza lectora.

Descripción

Función; envía la cabeza lectora al fotograma especificado en una escena y la detiene. Si no se especifica ninguna escena, la cabeza lectora se envía al fotograma de la escena actual. Puede utilizar el parámetro *scene* únicamente en la línea de tiempo raíz, no en las líneas de tiempo de los clips de película u otros objetos del documento.

Ejemplo

En el ejemplo siguiente, cuando el usuario hace clic en un botón al que se ha asignado `gotoAndStop()`, la cabeza lectora se desplaza al fotograma 5 de la escena actual y detiene la reproducción del archivo SWF:

```
on(keyPress "8") {  
    gotoAndStop(5);  
}
```

ifFrameLoaded()

Disponibilidad

Flash Lite 1.0.

Uso

```
ifFrameLoaded([scene,] frame) {  
    statement(s);  
}
```

Operandos

scene Una cadena opcional que identifica el nombre de la escena que se va a cargar.

frame Es necesario cargar el número de fotograma o la etiqueta de fotograma antes de ejecutar la sentencia siguiente.

statement(s) Las instrucciones que se deben ejecutar si se carga el fotograma, o escena y fotograma.

Descripción

Función; comprueba si el contenido de un fotograma específico está disponible localmente. Utilice la función `ifFrameLoaded` para comenzar a reproducir una animación sencilla mientras se descarga el resto del archivo SWF en un equipo local. También puede utilizar la propiedad `_framesloaded` para comprobar el progreso de descarga de un archivo SWF externo. La diferencia entre `_framesloaded` y `ifFrameLoaded` radica en que `_framesloaded` permite añadir sentencias `if` o `else` personalizadas.

Ejemplo

El ejemplo siguiente utiliza la función `ifFrameLoaded` para comprobar si se ha cargado el fotograma 10 del archivo SWF. Si el fotograma está cargado, el comando `trace()` imprime “frame number 10 is loaded” en el panel Salida. La variable `output` también se define con una variable `frame loaded: 10`.

```
ifFrameLoaded(10) {  
    trace ("frame number 10 is loaded");  
    output = "frame loaded: 10";  
}
```

Véase también

[_framesloaded](#)

int()

Disponibilidad

Flash Lite 1.0.

Uso

`int(value)`

Operandos

value Un número o cadena que se va a truncar a un entero.

Descripción

Función; trunca un número decimal a un valor entero.

Ejemplo

El código siguiente trunca los números de las variables `distance` y `myDistance`:

```
distance = 6.04 - 3.96;  
// trace ("distance = " add distance add " and rounded to:"  
// add int(distance));  
// Salida: distance = 2.08 y redondeado a: 2  
myDistance = "3.8";  
// trace ("myDistance = " add int(myDistance));  
// Salida: 3
```

length()

Disponibilidad

Flash Lite 1.0.

Uso

`length(expression)`

`length(variable)`

Operandos

expression Una cadena.

variable El nombre de una variable.

Descripción

Función String; devuelve el número de caracteres de la cadena o la variable especificada.

Ejemplo

El ejemplo siguiente devuelve la longitud de la cadena "Hello":

```
length("Hello");
```

El resultado es 5.

El ejemplo siguiente valida una dirección de correo electrónico comprobando que contiene al menos seis caracteres:

```
email = "someone@example.com";
if (length(email) > 6) {
    // trace ("parece que el mensaje tiene suficientes caracteres
    // para ser válido");
}
```

loadMovie()

Disponibilidad

Flash Lite 1.1.

Uso

```
loadMovie(url, target [, method])
```

Operandos

url Una cadena que especifica la URL absoluta o relativa del archivo SWF que se va a cargar. Una ruta relativa debe ser relativa al archivo SWF en el nivel 0. Las URL absolutas deben incluir la referencia al protocolo, como `http://` o `file:///`.

target Referencia a un clip de película o a una cadena que representa la ruta de acceso a un clip de película de destino. El clip de película de destino se sustituye por el archivo SWF que se carga.

method Parámetro opcional que especifica un método HTTP para enviar variables.

El parámetro debe ser la cadena `GET` o `POST`. Si no hay ninguna variable para enviar, omita este parámetro. El método `GET` añade las variables al final de la URL y se utiliza para números reducidos de variables. El método `POST` envía las variables en un encabezado HTTP independiente y se utiliza para enviar cadenas de variables largas.

Descripción

Función; carga un archivo SWF en Flash Lite durante la reproducción del archivo SWF original.

Para cargar un archivo SWF en un nivel específico, utilice la función `loadMovieNum()` en lugar de `loadMovie()`.

Cuando se carga un archivo SWF en un clip de película de destino, se puede utilizar la ruta de destino de dicho clip de película para buscar el archivo SWF cargado. Un archivo SWF que se carga en un destino hereda las propiedades de posición, rotación y escala del clip de película de destino. La esquina superior izquierda de la imagen o archivo SWF cargado se alinea con el punto de registro del clip de película de destino. Sin embargo, si el destino se encuentra en la línea de tiempo raíz, la esquina superior izquierda de la imagen o del archivo SWF se alinea con la esquina superior izquierda del escenario.

Utilice la función `unloadMovie()` para eliminar los archivos SWF que se cargaron con `loadMovie()`.

Ejemplo

El ejemplo siguiente carga el archivo SWF `circle.swf` desde el mismo directorio y reemplaza un clip de película llamado `mySquare` que ya existe en el escenario:

```
loadMovie("circle.swf", "mySquare");  
// Sentencia equivalente: loadMovie("circle.swf", _level0.mySquare);
```

Véase también

[_level](#), [loadMovieNum\(\)](#), [unloadMovie\(\)](#), [unloadMovieNum\(\)](#)

loadMovieNum()

Disponibilidad

Flash Lite 1.1.

Uso

```
loadMovieNum(url, level [, method])
```

Operandos

url Una cadena que especifica la URL absoluta o relativa del archivo SWF que se va a cargar. Una ruta relativa debe ser relativa al archivo SWF en el nivel 0. Para utilizar Flash Lite de forma independiente o para usarlo en el modo de prueba de la aplicación de edición de Flash, todos los archivos SWF deben estar guardados en la misma carpeta y los nombres de archivo no pueden contener especificaciones de carpeta o unidad de disco.

level Un entero que especifica el nivel en Flash Lite donde se carga el archivo SWF.

method Parámetro opcional que especifica un método HTTP para enviar variables.

Debe tener el valor `GET` o `POST`. Si no hay ninguna variable para enviar, omite este parámetro. El método `GET` añade las variables al final de la URL y se utiliza para números reducidos de variables. El método `POST` envía las variables en un encabezado HTTP independiente y se utiliza para enviar cadenas de variable largas.

Descripción

Función; carga un archivo SWF en un nivel de Flash Lite durante la reproducción del archivo SWF que se cargó originalmente.

Normalmente, Flash Lite muestra un solo archivo SWF y se cierra. La función `loadMovieNum()` permite mostrar varios archivos SWF simultáneamente y cambiar entre archivos SWF sin cargar otro documento HTML.

Para especificar un destino en lugar de un nivel, utilice la función `loadMovie()` en vez de `loadMovieNum()`.

Flash Lite tiene un orden de apilamiento de niveles a partir del nivel 0. Estos niveles son como capas de acetato: son transparentes, excepto en los objetos de cada nivel. Cuando utilice `loadMovieNum()`, debe especificar un nivel en Flash Lite donde se cargará el archivo SWF. Cuando se carga un archivo SWF en un nivel, puede utilizar la sintaxis `_levelN`, donde *N* es el número de nivel, para buscar el archivo SWF.

Al cargar un archivo SWF, puede especificar cualquier número de nivel. Puede cargar archivos SWF en un nivel que ya tenga un archivo SWF cargado y el nuevo SWF reemplaza al existente. Si carga un archivo SWF en el nivel 0, se descargarán todos los niveles de Flash Lite y se sustituirá el nivel 0 por el nuevo archivo. El archivo SWF en el nivel 0 establece la velocidad de fotogramas, el color de fondo y el tamaño de fotograma de todos los demás archivos SWF cargados.

Utilice `unloadMovieNum()` para eliminar los archivos SWF o las imágenes que se cargaron con `loadMovieNum()`.

Ejemplo

El ejemplo siguiente carga el archivo SWF en el nivel 2:

```
loadMovieNum("http://www.someserver.com/flash/circle.swf", 2);
```

Véase también

[_level](#), [loadMovie\(\)](#), [unloadMovieNum\(\)](#)

loadVariables()

Disponibilidad

Flash Lite 1.1.

Uso

```
loadVariables(url, target [, variables])
```

Operandos

url URL absoluta o relativa donde se ubican las variables. Si el archivo SWF que realiza esta llamada se ejecuta en un navegador Web, el valor *url* debe pertenecer al mismo dominio que el archivo SWF.

target Ruta de destino de un clip de película que recibe las variables que se cargan.

variables Parámetro opcional que especifica un método HTTP para enviar variables. El parámetro debe ser la cadena GET o POST. Si no hay ninguna variable para enviar, omita este parámetro. El método GET añade las variables al final de la URL y se utiliza para números reducidos de variables. El método POST envía las variables en un encabezado HTTP independiente y se utiliza para enviar cadenas de variable largas.

Descripción

Función; lee datos de un archivo externo, como un archivo de texto o texto generado por un script de ColdFusion, CGI, Active Server Pages (ASP), PHP o Perl, y establece los valores de las variables en un clip de película de destino. Esta función puede además actualizar las variables del archivo SWF activo con nuevos valores.

El texto y la URL especificada deben tener el formato MIME estándar *application/x-www-form-urlencoded* (formato estándar que se utiliza en los scripts CGI). Se puede especificar cualquier número de variables. Por ejemplo, la siguiente frase define varias variables:

```
company=Adobe&address=600+Townsend&city=San+Francisco&zip=94103
```

Para cargar variables en un nivel determinado, utilice la función `loadVariablesNum()` en lugar de `loadVariables()`.

Ejemplo

Los ejemplos siguientes cargan variables desde un archivo de texto y desde un servidor:

```
// cargar variables de archivo de texto en sistema de archivos local
// (Symbian Series 60)
on(release, keyPress "1") {
    filePath = "file://c:/documents/flash/myApp/myvariables.txt";
    loadVariables(filePath, _root);
}
```

```
// cargar variables (de servidor) en clip de película
urlPath = "http://www.someserver.com/myvariables.txt";
loadVariables(urlPath, "myClip_mc");
```

Véase también

[loadMovieNum\(\)](#), [loadVariablesNum\(\)](#), [unloadMovie\(\)](#)

loadVariablesNum()

Disponibilidad

Flash Lite 1.1.

Uso

```
loadVariablesNum(url, level [, variables])
```

Operandos

url Cadena que representa una URL absoluta o relativa donde se ubican las variables que se van a cargar. Si el archivo SWF que realiza esta llamada se ejecuta en un navegador Web, el valor *url* debe pertenecer al mismo dominio que el archivo SWF; para ver más detalles, consulte la sección Descripción.

level Entero que especifica el nivel de Flash Lite donde se van a recibir las variables.

variables Parámetro opcional que especifica un método HTTP para enviar variables.

El parámetro debe ser la cadena GET o POST. Si no hay ninguna variable para enviar, omite este parámetro. El método GET añade las variables al final de la URL y se utiliza para números reducidos de variables. El método POST envía las variables en un encabezado HTTP independiente y se utiliza para enviar cadenas de variable largas.

Descripción

Función; lee datos de un archivo externo, como un archivo de texto o texto generado por un script de ColdFusion, CGI, ASP, PHP o Perl, y establece los valores de las variables en un nivel de Flash Lite. Esta función puede además actualizar las variables del archivo SWF activo con nuevos valores.

El texto y la URL especificada deben tener el formato MIME estándar *application/x-www-form-urlencoded* (formato estándar que se utiliza en los scripts CGI). Se puede especificar cualquier número de variables. El siguiente ejemplo define varias variables:

```
company=Adobe&address=600+Townsend&city=San+Francisco&zip=94103
```

Normalmente, Flash Lite muestra un solo archivo SWF y se cierra. La función `loadVariablesNum()` permite mostrar varios archivos SWF simultáneamente y cambiar entre archivos SWF sin cargar otro documento HTML.

Para cargar variables en un clip de película de destino, utilice la función `loadVariables()` en lugar de `loadVariablesNum()`.

Véase también

[getURL\(\)](#), [loadMovie\(\)](#), [loadMovieNum\(\)](#), [loadVariables\(\)](#)

mbchr()

Disponibilidad

Flash Lite 1.0.

Uso

```
mbchr(number)
```

Operandos

number El número que se va a convertir en un carácter multibyte.

Descripción

Función String; convierte un número de código ASCII en un carácter multibyte.

Ejemplo

El ejemplo siguiente convierte números de código ASCII en sus caracteres multibyte equivalentes:

```
trace (mbchr(65));// Salida: A
trace (mbchr(97));// Salida: a
trace (mbchr(36));// Salida: $

myString = mbchr(51) - mbchr(49);
trace ("result = " add myString);// Salida: result = 2
```

Véase también

[mblength\(\)](#), [mbsubstring\(\)](#)

mblength()

Disponibilidad

Flash Lite 1.0.

Uso

```
mblength(string)
```

Operandos

string Una cadena.

Descripción

Función String; devuelve la longitud de la cadena de caracteres multibyte.

Ejemplo

El ejemplo siguiente muestra la longitud de la cadena en la variable `myString`:

```
myString = mbchr(36) add mbchr(50);
trace ("string length = " add mblength(myString));
// Salida: string length = 2
```

Véase también

[mbchr\(\)](#), [mbsubstring\(\)](#)

mbord()

Disponibilidad

Flash Lite 1.0.

Uso

`mbord(character)`

Operandos

character El carácter que se va a convertir en un número multibyte.

Descripción

Función String; convierte el carácter especificado en un número multibyte.

Ejemplo

Los ejemplos siguientes convierten los caracteres de la variable `myString` en números multibyte:

```
myString = "A";
trace ("ord = " + mbord(myString)); // Salida: 65

myString = "$120";
for (i=1; i<=length(myString); i++)
    char = substr(myString, i, 1);
    trace ("char ord = " + mbord(char)); // Salida: 36, 49, 50, 48
}
```

Véase también

[mbchr\(\)](#), [mbsubstring\(\)](#)

mbsubstring()

Disponibilidad

Flash Lite 1.0.

Uso

`mbsubstring(value, index, count)`

Operandos

value La cadena multibyte de la que se va a extraer una nueva cadena multibyte.

index El número del primer carácter que se va a extraer.

count El número de caracteres que se va a incluir en la cadena extraída, excluido el carácter de índice.

Descripción

Función `String`; extrae una cadena de caracteres multibyte nueva de la cadena de caracteres multibyte.

Ejemplo

El ejemplo siguiente extrae una nueva cadena de caracteres multibyte de la cadena que contiene la variable `myString`:

```
myString = mbchr(36) add mbchr(49) add mbchr(50) add mbchr(48);  
trace (mbsubstring(myString, 0, 2));// Salida: $1
```

Véase también

[mbchr\(\)](#)

nextFrame()

Disponibilidad

Flash Lite 1.0.

Uso

```
nextFrame()
```

Operandos

Ninguno.

Descripción

Función; envía la cabeza lectora al siguiente fotograma y la detiene en dicho lugar.

Ejemplo

En el ejemplo siguiente, cuando el usuario hace clic en el botón, la cabeza lectora se desplaza al siguiente fotograma y se detiene:

```
on (release) {  
    nextFrame();  
}
```

Véase también

[prevFrame\(\)](#)

nextScene()

Disponibilidad

Flash Lite 1.0.

Uso

```
nextScene()
```

Operandos

Ninguno.

Descripción

Función; envía la cabeza lectora al fotograma 1 de la siguiente escena y la detiene.

Ejemplo

En el ejemplo siguiente, cuando un usuario suelta el botón, la cabeza lectora se desplaza al fotograma 1 de la siguiente escena:

```
on(release) {  
    nextScene();  
}
```

Véase también

[prevScene\(\)](#)

Number()

Disponibilidad

Flash Lite 1.0.

Uso

```
Number(expression)
```

Operandos

expression Expresión para convertir en un número.

Descripción

Función; convierte el parámetro *expression* en un número y devuelve un valor como se describe en la siguiente lista:

- Si *expression* es un número, el valor devuelto es *expression*.
- Si *expression* es un valor booleano, el valor devuelto es 1 si *expression* es `true` y 0 si *expression* es `false`.
- Si *expression* es una cadena, la función intenta analizar *expression* como un número decimal con un exponente final opcional (es decir, `1.57505e-3`).
- Si *expression* es `undefined`, el valor devuelto es `-1`.

Ejemplo

El ejemplo siguiente convierte la cadena de la variable `myString` en un número, almacena el número en la variable `myNumber`, añade 5 al número y almacena el resultado en la variable `myResult`. La línea final muestra el resultado cuando se llama a `Number()` en un valor booleano.

```
myString = "55";
myNumber = Number(myString);
myResult = myNumber + 5;

trace (myResult); // Salida: 60

trace (Number(true)); // Salida: 1
```

on()

Disponibilidad

Flash Lite 1.0.

Uso

```
on(event) {
    // statement(s)
}
```

Operandos

statement(s) Las instrucciones que se van a ejecutar cuando se produzca *event*.

event Este parámetro se denomina *event* (evento). Cuando se produce un evento de usuario, se ejecutan las sentencias que aparecen a continuación entre llaves (`{ }`). Puede especificarse cualquiera de los siguientes valores para el parámetro *event*:

- `press` Se presiona el botón del ratón cuando el puntero se encuentra sobre el botón.

- `release` Se suelta el botón del ratón cuando el puntero se encuentra sobre el botón.
- `rollOut` El puntero se desplaza fuera del área del botón.
- `rollOver` El puntero del ratón se desplaza sobre el botón.
- `keyPress "key"` Se presiona la tecla especificada. Para la parte del parámetro referente a la tecla (`key`), especifique un código o constante de tecla.

Descripción

Controlador de eventos; especifica el evento de usuario o la pulsación de tecla que activa una función. No se admiten todos los eventos.

Ejemplo

El código siguiente, que desplaza el campo `myText` una línea hacia abajo cuando el usuario presiona la tecla 8, prueba `maxscroll` antes del desplazamiento:

```
on (keyPress "8") {  
    if (myText.scroll < myText.maxscroll) {  
        myText.scroll++;  
    }  
}
```

ord()

Disponibilidad

Flash Lite 1.0.

Uso

`ord(character)`

Operandos

character El carácter que se va a convertir en un número del código ASCII.

Descripción

Función String; convierte caracteres en números de código ASCII.

Ejemplo

El ejemplo siguiente utiliza la función `ord()` para mostrar el código ASCII correspondiente al carácter *A*:

```
trace ("multibyte number = " + add ord("A")); // Salida: multibyte number = 65
```

play()

Disponibilidad

Flash Lite 1.0.

Uso

play()

Operandos

Ninguno.

Descripción

Función; mueve la cabeza lectora hacia delante en la línea de tiempo.

Ejemplo

El ejemplo siguiente utiliza una sentencia `if` para comprobar el valor de un nombre que introduce el usuario. Si el usuario escribe `Steve`, se llama a la función `play()` y la cabeza lectora avanza en la línea de tiempo. Si el usuario introduce cualquier otra cadena distinta de `Steve`, el archivo SWF no se reproduce y aparece un campo de texto con la variable `alert`.

```
stop();
if (name == "Steve") {
    play();
} else {
    alert="You are not Steve!";
}
```

prevFrame()

Disponibilidad

Flash Lite 1.0.

Uso

prevFrame()

Operandos

Ninguno.

Descripción

Función; envía la cabeza lectora al fotograma anterior y la detiene en dicho lugar. Si el fotograma actual es el fotograma 1, la cabeza lectora no se mueve.

Ejemplo

Cuando el usuario hace clic en un botón con el siguiente controlador asociado, se envía la cabeza lectora al fotograma anterior:

```
on(release) {  
    prevFrame();  
}
```

Véase también

[nextFrame\(\)](#)

prevScene()

Disponibilidad

Flash Lite 1.0.

Uso

```
prevScene()
```

Operandos

Ninguno.

Descripción

Función; envía la cabeza lectora al fotograma 1 de la escena anterior y la detiene.

Ejemplo

En este ejemplo, cuando el usuario hace clic en un botón con el siguiente controlador asociado, se envía la cabeza lectora a la escena anterior:

```
on(release) {  
    prevScene();  
}
```

Véase también

[nextScene\(\)](#)

random()

Disponibilidad

Flash Lite 1.0.

Uso

```
random(value)
```

Operandos

value Un entero.

Descripción

Función; devuelve un entero aleatorio entre 0 y uno menos que el entero especificado en el parámetro *value*.

Ejemplo

Los ejemplos siguientes generan un número basado en un entero que especifica el rango:

```
//escoger número aleatorio entre 0 y 5
myNumber = random(5);
trace (myNumber); // Salida: podría ser 0,1,2,3,4
```

```
//escoger número aleatorio entre 5 y 10
myNumber = random(5) + 5;
trace (myNumber); // Salida: podría ser 5,6,7,8,9
```

Los ejemplos siguientes generan un número y, a continuación, lo concatenan al final de una cadena evaluada como un nombre de variable. Este es un ejemplo de cómo puede utilizarse la sintaxis de Flash Lite 1.1 para simular matrices.

```
// seleccionar nombre aleatorio de la lista
myNames1 = "Mike";
myNames2 = "Debbie";
myNames3 = "Logan";

ran = random(3) + 1;
ranName = "myNames" + ran;
trace (eval(ranName)); // Salida: será mike, debbie o logan
```

removeMovieClip()

Disponibilidad

Flash Lite 1.0.

Uso

```
removeMovieClip(target)
```

Operandos

target La ruta de destino de una instancia de clip de película creada con `duplicateMovieClip()`.

Descripción

Función, elimina el clip de película especificado que se creó originalmente mediante `duplicateMovieClip()`.

Ejemplo

El ejemplo siguiente elimina el clip de película `second_mc`:

```
duplicateMovieClip("person_mc", "second_mc", 1);  
second_mc:_x = 55;  
second_mc:_y = 85;  
removeMovieClip("second_mc");
```

set()

Disponibilidad

Flash Lite 1.0.

Uso

```
set(variable, expression)
```

Operandos

variable Un identificador para el valor del parámetro *expression*.

expression Valor asignado a la variable.

Descripción

Sentencia; asigna un valor a una variable. Una *variable* es un contenedor que almacena datos. El contenedor es siempre el mismo, pero el contenido puede cambiar. La modificación del valor de una variable a medida que se reproduce el archivo SWF permite registrar y guardar información sobre las acciones del usuario, registrar valores que se modifican conforme se reproduce el archivo SWF o comprobar si una determinada condición es `true` o `false`.

Las variables pueden contener cualquier tipo de datos (por ejemplo, `String`, `Number`, `Boolean` o `MovieClip`). La línea de tiempo de cada archivo SWF y clip de película tiene su propio conjunto de variables y cada variable tiene su propio valor independiente de las variables de otras líneas de tiempo.

Ejemplo

El ejemplo siguiente establece una variable denominada `orig_x_pos`, que almacena la posición del eje *x* original del clip de película `ship` de forma que se restablezca a su posición de inicio más adelante en el archivo SWF:

```
on(release) {  
    set("orig_x_pos", getProperty("ship", _x));  
}
```

El código anterior obtiene el mismo resultado que el siguiente:

```
on(release) {  
    orig_x_pos = ship._x;  
}
```

setProperty()

Disponibilidad

Flash Lite 1.0.

Uso

`setProperty(target, property, value/expression)`

Operandos

target Ruta al nombre de instancia del clip de película cuya propiedad va a establecerse.

property Propiedad que va a establecerse.

value Nuevo valor literal de la propiedad.

expression Una ecuación que halla el nuevo valor de la propiedad.

Descripción

Función; cambia un valor de propiedad de un clip de película durante la reproducción de la película.

Ejemplo

La sentencia siguiente define la `_alpha` del clip de película `star` como el 30 por ciento cuando el usuario hace clic en el botón asociado a este controlador de eventos:

```
on(release) {  
    setProperty("star", _alpha, "30");  
}
```

Véase también

[getProperty\(\)](#)

stop()

Disponibilidad

Flash Lite 1.0.

Uso

```
stop()
```

Operandos

Ninguno.

Descripción

Función; detiene el archivo SWF que se está reproduciendo. El uso más común de esta función es controlar clips de película mediante botones.

Ejemplo

La sentencia siguiente llama a la función `stop()` cuando el usuario hace clic en el botón asociado al controlador de eventos:

```
on(release) {  
    stop();  
}
```

stopAllSounds()

Disponibilidad

Flash Lite 1.0.

Uso

```
stopAllSounds()
```

Operandos

Ninguno.

Descripción

Función; detiene todos los sonidos que se están reproduciendo en un archivo SWF sin detener la cabeza lectora. Se reanuda la reproducción de los sonidos que deben transmitirse mientras la cabeza lectora se mueve sobre los fotogramas donde se encuentran.

Ejemplo

El código siguiente podría aplicarse a un botón que detuviera todos los sonidos del archivo SWF :

```
on(release) {  
    stopAllSounds();  
}
```

String()

Disponibilidad

Flash Lite 1.0.

Uso

```
String(expression)
```

Operandos

expression Expresión para convertir en una cadena.

Descripción

Función; devuelve una representación de cadena del parámetro especificado, tal y como se describe en la siguiente lista:

- Si *expression* es un número, la cadena devuelta es una representación en texto del número.
- Si *expression* es una cadena, la cadena devuelta es *expression*.

- Si *expression* es un valor booleano, la cadena devuelta es `true` o `false`.
- Si *expression* es un clip de película, el valor devuelto es la ruta de destino del clip de película en notación con barras (*/*).

Ejemplo

El ejemplo siguiente define `birthYearNum` como `1976`, lo convierte en una cadena mediante la función `String()` y lo compara con la cadena "1976" utilizando el operador `eq`.

```
birthYearNum = 1976;
birthYearStr = String(birthYearNum);
if (birthYearStr eq "1976") {
    trace ("birthYears match");
}
```

substring()

Disponibilidad

Flash Lite 1.0.

Uso

`substring(string, index, count)`

Operandos

string La cadena de la que se va a extraer la nueva cadena.

index El número del primer carácter que se va a extraer.

count El número de caracteres que se va a incluir en la cadena extraída, excluido el carácter de índice.

Descripción

Función; extrae parte de una cadena. Esta función se basa en uno, mientras que los métodos de la clase `String` se basan en cero.

Ejemplo

El siguiente ejemplo extrae los cinco primeros caracteres de la cadena "Hello World":

```
origString = "Hello World!";
newString = substring(origString, 0, 5);
trace (newString); // Salida: Hello
```

tellTarget()

Disponibilidad

Flash Lite 1.0.

Uso

```
tellTarget(target) {  
    statement(s);  
}
```

Operandos

target Una cadena que especifica la ruta de destino de la línea de tiempo que se va a controlar.

statement(s) Instrucciones que se ejecutarán si la condición da como resultado `true`.

Descripción

Función, aplica las instrucciones especificadas en el parámetro *statement(s)* a la línea de tiempo especificada en el parámetro *target*. La función `tellTarget()` resulta útil para los controles de navegación. Asigne `tellTarget()` a botones que detengan o inicien clips de película en otros lugares del escenario. También puede hacer que los clips de película pasen a un determinado fotograma de ese clip. Por ejemplo, puede asignar `tellTarget()` a botones que detienen o inician clips de película en el escenario o que hacen que clips de película pasen a un determinado fotograma.

Ejemplo

En el ejemplo siguiente, `tellTarget()` controla la instancia de clip de película `ball` en la línea de tiempo principal. El fotograma 1 de la instancia `ball` está en blanco y tiene una acción `stop()` para que no sea visible en el escenario. Cuando el usuario presiona la tecla 5, `tellTarget()` indica a la cabeza lectora en `ball` que vaya al fotograma 2 donde comienza la animación.

```
on(keyPress "5") {  
    tellTarget("ball") {  
        gotoAndPlay(2);  
    }  
}
```

toggleHighQuality()

Disponibilidad

Flash Lite 1.0.

Uso

```
toggleHighQuality()
```

Operandos

Ninguno.

Descripción

Función; activa y desactiva el suavizado en Flash Lite. El suavizado alisa los bordes de los objetos, pero ralentiza la reproducción del SWF. Esta función afecta a todos los archivos SWF de Flash Lite.

Ejemplo

El código siguiente puede aplicarse a un botón que, cuando se hace clic en él, activa o desactiva el suavizado:

```
on(release) {  
    toggleHighQuality();  
}
```

trace()

Disponibilidad

Flash Lite 1.0.

Uso

```
trace(expression)
```

Operandos

expression Expresión que se va a evaluar. Cuando se abre un archivo SWF en una herramienta Flash (mediante el comando Probar película), aparece en el panel Salida el valor del parámetro *expression*.

Descripción

Función; evalúa la expresión y muestra el resultado en el panel Salida en modo de prueba.

Utilice esta función para registrar las notas de programación o para mostrar mensajes en el panel Salida mientras prueba un archivo SWF. Utilice el parámetro *expression* para comprobar si existe una condición o para mostrar valores en el panel Salida. La función `trace()` es similar a la función `alert` de JavaScript.

Puede utilizar el comando Omitir acciones de trazado de la configuración de publicación para eliminar funciones `trace()` del archivo SWF exportado.

Ejemplo

El ejemplo siguiente utiliza la función `trace()` para observar el comportamiento de un bucle `while`:

```
i = 0;
while (i++ < 5){
    trace("this is execution " + i);
}
```

unloadMovie()

Disponibilidad

Flash Lite 1.0.

Uso

```
unloadMovie(target)
```

Operandos

target Ruta de destino de un clip de película.

Descripción

Función; elimina de Flash Lite un clip de película que se cargó mediante `loadMovie()`, `loadMovieNum()` o `duplicateMovieClip()`.

Ejemplo

Cuando el usuario presiona la tecla 3, responde el siguiente código descargando el clip de película `draggable_mc` en la línea de tiempo principal y cargando `movie.swf` en el nivel 4 de la pila de documentos:

```
on (keypress "3") {
    unloadMovie ("/draggable_mc");
    loadMovieNum("movie.swf", 4);
}
```

Cuando el usuario presiona la tecla 3, el siguiente ejemplo descarga la película que se cargó en el nivel 4:

```
on (keypress "3") {  
    unloadMovieNum(4);  
}
```

Véase también

[loadMovie\(\)](#)

unloadMovieNum()

Disponibilidad

Flash Lite 1.0.

Uso

```
unloadMovieNum(level)
```

Operandos

level El nivel (*_levelN*) de una película cargada.

Descripción

Función; elimina de Flash Lite un clip de película que se cargó mediante [loadMovie\(\)](#), [loadMovieNum\(\)](#) o [duplicateMovieClip\(\)](#).

Normalmente, Flash Lite muestra un solo archivo SWF y se cierra. La función [loadMovieNum\(\)](#) permite mostrar varios archivos SWF simultáneamente y cambiar entre archivos SWF sin cargar otro documento HTML.

Véase también

[loadMovieNum\(\)](#)

En esta sección se describen las propiedades que reconoce Macromedia Flash Lite 1.x de Adobe. Las entradas se muestran en orden alfabético, ignorando los caracteres de subrayado iniciales. Las propiedades figuran en la tabla siguiente:

Propiedad	Descripción
<code>/</code> (Barra diagonal)	Especifica o devuelve una referencia a la línea de tiempo del clip de película principal.
<code>_alpha</code>	Devuelve el valor de transparencia alfa de un clip de película.
<code>_currentframe</code>	Devuelve el número del fotograma en el que está situada la cabeza lectora en la línea de tiempo.
<code>_focusrect</code>	Especifica si aparece un rectángulo amarillo alrededor del botón o campo de texto seleccionado.
<code>_framesloaded</code>	Devuelve el número de fotogramas que se han cargado de un archivo SWF cargado de forma dinámica.
<code>_height</code>	Especifica la altura del clip de película, expresada en píxeles.
<code>_highquality</code>	Especifica el nivel de visualización suavizada que se aplica al archivo SWF actual.
<code>_level</code>	Devuelve una referencia a la línea de tiempo raíz de <code>_levelN</code> . Debe utilizar la función <code>loadMovieNum()</code> para cargar archivos SWF en el reproductor de Flash Lite antes de utilizar la propiedad <code>_level</code> para buscarlos. También puede utilizar <code>_levelN</code> para buscar un archivo SWF cargado en el nivel asignado por <i>N</i> .
<code>maxscroll</code>	Indica el número de la primera línea de texto visible en un campo de texto desplazable cuando la última línea del campo también está visible.
<code>_name</code>	Devuelve el nombre de instancia de un clip de película. Sólo se aplica a los clips de película y no a la línea de tiempo principal.
<code>_rotation</code>	Devuelve el giro del clip de película, expresado en grados, con respecto a su orientación original.

Propiedad	Descripción
<code>scroll</code>	Controla la visualización de información en un campo de texto asociado a una variable. La propiedad <code>scroll</code> define dónde comienza el campo de texto a mostrar contenido; después de establecerla, Flash Lite la actualiza a medida que el usuario se desplaza por el campo de texto.
<code>_target</code>	Devuelve la ruta de destino de la instancia de clip de película.
<code>_totalframes</code>	Devuelve el número total de fotogramas de un clip de película.
<code>_visible</code>	Indica si un clip de película es visible.
<code>_width</code>	Devuelve la anchura del clip de película, expresada en píxeles.
<code>_x</code>	Contiene un entero que define la coordenada x de un clip de película.
<code>_xscale</code>	Determina la escala horizontal (<i>porcentaje</i>) del clip de película aplicada desde el punto de registro del clip de película.
<code>_y</code>	Contiene un entero que establece la coordenada y de un clip de película en relación a las coordenadas locales del clip de película principal.
<code>_yscale</code>	Determina la escala vertical (<i>porcentaje</i>) del clip de película aplicada desde el punto de registro del clip de película.

/ (Barra diagonal)

Disponibilidad

Flash Lite 1.0

Uso

/

/targetPath

/:varName

Descripción

Identificador; especifica o devuelve una referencia a la línea de tiempo del clip de película principal. La funcionalidad que proporciona esta propiedad es similar a la de la propiedad `_root` en Flash 5.

Ejemplo

Para especificar una variable en una línea de tiempo, utilice la sintaxis (/) combinada con dos puntos (:).

Ejemplo 1: La variable `car` en la línea de tiempo principal:

```
/:car
```

Ejemplo 2: La variable `car` en la instancia de clip de película `mc1` que reside en la línea de tiempo principal:

```
/mc1/:car
```

Ejemplo 3: La variable `car` en la instancia de clip de película `mc2` anidada en la instancia `mc1` que reside en la línea de tiempo principal:

```
/mc1/mc2/:car
```

Ejemplo 4: La variable `car` en la instancia de clip de película `mc2` que reside en la línea de tiempo principal:

```
mc2/:car
```

_alpha

Disponibilidad

Flash Lite 1.0.

Uso

my_mc._alpha

Propiedad; valor de transparencia alfa del clip de película especificado por la variable *my_mc*. Los valores válidos son de 0 (totalmente transparente) a 100 (totalmente opaco), que es el valor predeterminado. Los objetos existentes en un clip de película que tenga configurado *_alpha* con el valor 0 continuarán activos aunque no sean visibles. Por ejemplo, puede hacer clic en un botón en un clip de película cuya propiedad *_alpha* sea 0.

Ejemplo

El código siguiente para un controlador de eventos de botón define la propiedad *_alpha* del clip de película *my_mc* como 30% cuando el usuario hace clic en el botón:

```
on(release) {  
    tellTarget("my_mc") {  
        _alpha = 30;  
    }  
}
```

_currentframe

Disponibilidad

Flash Lite 1.0.

Uso

my_mc._currentframe

Descripción

Propiedad (de sólo lectura); devuelve el número del fotograma en el que está situada la cabeza lectora en la línea de tiempo especificada por la variable *my_mc*.

Ejemplo

El ejemplo siguiente utiliza la propiedad `_currentframe` y la función `gotoAndStop()` para ordenar a la cabeza lectora del clip de película `my_mc` que avance cinco fotogramas con respecto a su posición actual:

```
tellTarget("my_mc") {  
    gotoAndStop(_currentframe + 5);  
}
```

Véase también

[gotoAndStop\(\)](#)

_focusrect

Disponibilidad

Flash Lite 1.0.

Uso

```
_focusrect = Boolean;
```

Descripción

Propiedad (global); especifica si aparece un rectángulo amarillo alrededor del botón o campo de texto seleccionado. El valor predeterminado, `true`, muestra un rectángulo amarillo alrededor del botón o campo de texto seleccionado cuando el usuario presiona las teclas de flecha arriba o abajo del teléfono o dispositivo móvil para desplazarse por los objetos de un archivo SWF. Especifique `false` si no desea mostrar el rectángulo amarillo.

Ejemplo

El ejemplo siguiente desactiva el rectángulo amarillo para que no aparezca en la aplicación:

```
_focusrect = false;
```

_framesloaded

Disponibilidad

Flash Lite 1.0.

Uso

my_mc:_framesloaded

Descripción

Propiedad (de sólo lectura); devuelve el número de fotogramas que se han cargado de un archivo SWF cargado de forma dinámica. Esta propiedad es útil para determinar si se ha cargado el contenido de un fotograma específico y de todos los que le preceden y si está disponible localmente en el navegador. También resulta útil para controlar el progreso de la descarga de archivos SWF de gran tamaño. Por ejemplo, puede que desee mostrar un mensaje a los usuarios para indicar que el archivo SWF se está cargando hasta que un fotograma concreto del archivo SWF haya terminado de cargarse.

Ejemplo

El ejemplo siguiente utiliza la propiedad `_framesloaded` para iniciar un archivo SWF cuando se han cargado todos los fotogramas. Si no están cargados todos los fotogramas, la propiedad `_xscale` de la instancia de clip de película `loader` aumenta proporcionalmente para crear una barra de progreso.

```
if (_framesloaded >= _totalframes) {
    gotoAndPlay ("Scene 1", "start");
} else {
    tellTarget("loader") {
        _xscale = (_framesloaded/_totalframes)*100;
    }
}
```

_height

Disponibilidad

Flash Lite 1.0.

Uso

my_mc:_height

Descripción

Propiedad (de sólo lectura); altura del clip de película, expresada en píxeles.

Ejemplo

El ejemplo siguiente del código de controlador de eventos define la altura de un clip de película cuando el usuario hace clic con el botón del ratón:

```
on(release) {  
    tellTarget("my_mc") {  
        _height = 200;  
    }  
}
```

_highquality

Disponibilidad

Flash Lite 1.0.

Uso

`_highquality`

Descripción

Propiedad (global); especifica el nivel de visualización suavizada que se aplica al archivo SWF actual. Especifique 2 para obtener suavizado de calidad óptima. Especifique 1 para obtener suavizado de calidad alta. Especifique 0 para evitar la visualización suavizada.

Ejemplo

La sentencia siguiente aplica suavizado de alta calidad al archivo SWF actual:

```
_highquality = 1;
```

Véase también

[toggleHighQuality\(\)](#)

`_level`

Disponibilidad

Flash Lite 1.0.

Uso

`_levelN`

Descripción

Identificador; referencia a la línea de tiempo raíz de `_levelN`. Debe utilizar la función `loadMovieNum()` para cargar archivos SWF en el reproductor de Flash Lite antes de utilizar la propiedad `_level` para buscarlos. También puede utilizar `_levelN` para buscar un archivo SWF cargado en el nivel asignado por *N*.

El archivo SWF inicial cargado en una instancia de Flash Player Lite se carga automáticamente en `_level0`. El archivo SWF que se encuentra en `_level0` establece la velocidad de fotogramas, el color de fondo y el tamaño de fotograma de todos los archivos SWF que se cargan posteriormente. A continuación, los archivos SWF se apilan en niveles de numeraciones superiores, por encima del archivo SWF de `_level0`.

Debe asignar un nivel a cada archivo SWF que cargue en el reproductor de Flash Lite mediante `loadMovieNum()`. Puede asignar niveles en cualquier orden. Si asigna un nivel que ya contiene un archivo SWF (incluido `_level0`), se descargará el archivo SWF de dicho nivel y será sustituido por el nuevo archivo SWE.

Ejemplo

El ejemplo siguiente carga un archivo SWF en el Nivel 1 y, a continuación, detiene la cabeza lectora del archivo SWF cargado en el fotograma 6:

```
loadMovieNum("mySWF.swf", 1);

// al menos 1 fotograma más tarde
tellTarget(_level1) {
    gotoAndStop(6);
}
```

Véase también

[loadMovie\(\)](#)

maxscroll

Disponibilidad

Flash Lite 1.1

Uso

variable_name:maxscroll

Descripción

Propiedad (de sólo lectura); indica el número de la primera línea de texto visible en un campo de texto desplazable cuando la última línea del campo también está visible. La propiedad `maxscroll` funciona con la propiedad `scroll` para controlar cómo aparece la información en un campo de texto. Esta propiedad puede recuperarse, pero no modificarse.

Ejemplo

El código siguiente, que desplaza el campo `myText` una línea hacia abajo cuando el usuario presiona la tecla 8, prueba `maxscroll` antes del desplazamiento:

```
on(keyPress "8") {
    if (myText:scroll < myText:maxscroll) {
        myText:scroll++;
    }
}
```

Véase también

[scroll](#)

_name

Disponibilidad

Flash Lite 1.0.

Uso

my_mc:_name

Descripción

Propiedad; el nombre de instancia del clip de película especificado por *my_mc*. Sólo se aplica a los clips de película y no a la línea de tiempo principal.

Ejemplo

El ejemplo siguiente muestra el nombre del clip de película `bigRose` en el panel Salida como una cadena:

```
trace(bigRose:_name);
```

_rotation

Disponibilidad

Flash Lite 1.0.

Uso

my_mc:_rotation

Descripción

Propiedad; giro del clip de película, expresado en grados, con respecto a su orientación original. Los valores comprendidos entre 0 y 180 representan un giro en el sentido de las agujas del reloj, mientras que los comprendidos entre 0 y -180 representan un giro en sentido contrario al de las agujas del reloj. Los valores situados fuera de este rango se suman o restan de 360 para obtener un valor que sí esté comprendido en el rango. Por ejemplo, la sentencia `my_mc:_rotation = 450` es igual que `my_mc:_rotation = 90`.

Ejemplo

El ejemplo siguiente gira el clip de película `my_mc` 15 grados hacia la derecha cuando el usuario presiona la tecla 2:

```
on (keyPress "2") {  
    my_mc:_rotation += 15;  
}
```

scroll

Disponibilidad

Flash Lite 1.1.

Uso

textFieldVariableName:scroll

Descripción

Propiedad; controla la visualización de información en un campo de texto asociado a una variable. La propiedad `scroll` define dónde comienza el campo de texto a mostrar contenido; después de establecerla, Flash Lite la actualiza a medida que el usuario se desplaza por el campo de texto. Puede utilizar la propiedad `scroll` para crear un campo de texto desplazable con el fin de dirigir al usuario a un párrafo específico en un fragmento de texto largo.

Ejemplo

El código siguiente desplaza el campo `myText` hacia arriba una línea cada vez que el usuario presiona la tecla 2:

```
on(keyPress "2") {  
    if (myText:scroll > 1) {  
        myText:scroll--;  
    }  
}
```

Véase también

[maxscroll](#)

_target

Disponibilidad

Flash Lite 1.0.

Uso

`my_mc:_target`

Descripción

Propiedad (de sólo lectura); devuelve la ruta de destino de la instancia de clip de película especificada por `my_mc`.

_totalframes

Disponibilidad

Flash Lite 1.0.

Uso

`my_mc:_totalframes`

Descripción

Propiedad (de sólo lectura); devuelve el número total de fotogramas del clip de película `my_mc`.

Ejemplo

El código siguiente carga `mySWF.swf` en el Nivel 1 y 25 más adelante, comprueba si se ha cargado:

```
loadMovieNum("mySWF.swf", 1);

// 25 fotogramas más adelante en la línea de tiempo
if (_level1._framesloaded >= _level1._totalframes) {
    tellTarget("_level1/") {
        gotoAndStop("myLabel");
    }
} else {
    // bucle...
}
```

_visible

Disponibilidad

Flash Lite 1.0.

Uso

`my_mc:_visible`

Descripción

Propiedad; valor booleano que indica si el clip de película especificado por `my_mc` es visible. Los clips de película no visibles (que tienen la propiedad `_visible` configurada como `false`) se desactivan. Por ejemplo, no es posible hacer clic en un botón de un clip de película con `_visible` configurado con el valor `false`. Los clips de película son visibles a menos que se hagan invisibles de forma explícita como se ha descrito.

Ejemplo

El código siguiente desactiva el clip de película `my_mc` cuando el usuario presiona la tecla 3 y lo activa cuando el usuario presiona 4:

```
on(keyPress "3") {
    my_mc:_visible = 0;
}

on(keyPress "4") {
    my_mc:_visible = 1;
}
```

_width

Disponibilidad

Flash Lite 1.0.

Uso

my_mc:_width

Descripción

Propiedad; la anchura del clip de película, expresada en píxeles.

Ejemplo

El ejemplo siguiente define las propiedades de anchura de un clip de película cuando el usuario presiona la tecla 5:

```
on(keyPress "5") {  
    my_mc:_width = 10;  
}
```

_X

Disponibilidad

Flash Lite 1.0.

Uso

my_mc:_x

Descripción

Propiedad; entero que establece la coordenada *x* de un clip de película (representada aquí por *my_mc*) en relación a las coordenadas locales del clip de película principal. Si un clip de película se encuentra en la línea de tiempo principal, su sistema de coordenadas hará referencia a la esquina superior izquierda del escenario como (0, 0).

Si el clip de película está dentro de otro clip de película que incluye transformaciones, el clip de película estará en el sistema de coordenadas local del clip de película en el que está contenido. Por ejemplo, si un clip de película se gira 90 grados en sentido contrario a las agujas del reloj, el clip de película secundario hereda un sistema de coordenadas que se gira 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.

Ejemplo

El ejemplo siguiente cambia la posición horizontal del clip de película `my_mc` cuando el usuario presiona la tecla 6:

```
on(keyPress "6") {  
    my_mc:_x = 10;  
}
```

Véase también

[_xscale](#), [_y](#), [_yscale](#)

_xscale

Disponibilidad

Flash Lite 1.0.

Uso

`my_mc:_xscale`

Descripción

Propiedad; establece la escala horizontal (*porcentaje*) del clip de película aplicada desde el punto de registro del clip de película. El punto de registro predeterminado es (0, 0).

La escala del sistema de coordenadas local afecta a la configuración de las propiedades de `_x` e `_y`, que se definen en píxeles. Por ejemplo, si se aplica una escala del 50% al clip de película principal, la configuración de la propiedad `_x` desplazará un objeto situado en el clip de película la mitad de píxeles que si la película tuviera una escala del 100%.

Ejemplo

El ejemplo siguiente cambia la escala horizontal del clip de película `my_mc` cuando el usuario presiona la tecla 7:

```
on(keyPress "7") {  
    my_mc:_xscale = 10;  
}
```

Véase también

[_x](#), [_y](#), [_yscale](#)

_y

Disponibilidad

Flash Lite 1.0.

Uso

my_mc:_y

Descripción

Propiedad; entero que establece la coordenada *y* de un clip de película (representada aquí por *my_mc*) en relación a las coordenadas locales del clip de película principal. Si un clip de película se encuentra en la línea de tiempo principal, su sistema de coordenadas hará referencia a la esquina superior izquierda del escenario como (0, 0).

Si el clip de película está dentro de otro clip de película que incluye transformaciones, el clip de película estará en el sistema de coordenadas local del clip de película en el que está contenido. Por ejemplo, si un clip de película se gira 90 grados en sentido contrario a las agujas del reloj, el clip de película secundario hereda un sistema de coordenadas que se gira 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.

Ejemplo

El código siguiente define las coordenadas *y* del clip de película *my_mc* 10 píxeles debajo de la coordenada (0, 0) del clip principal cuando el usuario presiona la tecla 1:

```
on(keyPress "1") {  
    my_mc:_y = 10;  
}
```

Véase también

[_x](#), [_xscale](#), [_yscale](#)

_yscale

Disponibilidad

Flash Lite 1.0.

Uso

`my_mc:_yscale`

Descripción

Propiedad; establece la escala vertical (*porcentaje*) del clip de película aplicada desde el punto de registro del clip de película. El punto de registro predeterminado es (0, 0).

La escala del sistema de coordenadas local afecta a la configuración de las propiedades de `_x` e `_y`, que se definen en píxeles. Por ejemplo, si se aplica una escala del 50% al clip de película principal, la configuración de la propiedad `_y` desplazará un objeto situado en el clip de película la mitad de píxeles que si la película tuviera una escala del 100%.

Ejemplo

El ejemplo siguiente cambia la escala vertical del clip de película `my_mc` a 10% cuando el usuario presiona la tecla 1:

```
on(keyPress "1") {  
    my_mc:_yscale = 10;  
}
```

Véase también

[_x](#), [_xscale](#), [_y](#)

En esta sección se describe la sintaxis y el uso de las sentencias de ActionScript de Macromedia Flash Lite 1.x de Adobe, que son elementos del lenguaje que realizan o especifican una acción. Las sentencias figuran en la tabla siguiente:

Sentencia	Descripción
<code>break</code>	Indica a Flash Lite que omita el resto del cuerpo del bucle, detenga su acción y ejecute la sentencia que sigue a la del bucle.
<code>case</code>	Define una condición para la sentencia <code>switch</code> . Las sentencias del parámetro <code>statements</code> se ejecutan si el parámetro <code>expression</code> que sigue a la palabra clave <code>case</code> coincide con el parámetro <code>expression</code> de la sentencia <code>switch</code> .
<code>continue</code>	Salta por encima de todas las sentencias restantes en el bucle interior e inicia la siguiente repetición del bucle como si se hubiera pasado el control hasta el final del bucle de la forma habitual.
<code>do..while</code>	Ejecuta las sentencias y, a continuación, evalúa la condición en un bucle siempre que sea <code>true</code> .
<code>else</code>	Especifica las sentencias que deben ejecutarse si la condición de la sentencia <code>if</code> es <code>false</code> .
<code>else if</code>	Evalúa una condición y especifica las sentencias que se ejecutarán si la condición en la sentencia <code>if</code> inicial devuelve <code>false</code> .
<code>for</code>	Evalúa la expresión <code>init</code> (inicializar) una vez y, a continuación, inicia una secuencia de bucle por la que, siempre que <code>condition</code> sea <code>true</code> , se ejecuta <code>statement</code> , y se evalúa la sentencia siguiente.
<code>if</code>	Evalúa una condición para determinar la siguiente acción en un archivo SWF. Si la condición es <code>true</code> , Flash Lite ejecuta las sentencias que hay entre llaves (<code>{}</code>), a continuación de la condición. Si la condición es <code>false</code> , Flash Lite omite las sentencias entre llaves y ejecuta las sentencias que hay a continuación.

Sentencia	Descripción
switch	Al igual que la sentencia <code>if</code> , la sentencia <code>switch</code> prueba una condición y ejecuta sentencias si la condición devuelve un valor <code>true</code> .
while	Prueba una expresión y ejecuta una sentencia o una serie de sentencias de forma repetida en un bucle, siempre que la expresión sea <code>true</code> .

break

Disponibilidad

Flash Lite 1.0.

Uso

`break`

Parámetros

Ninguno.

Descripción

Sentencia; aparece en un bucle (`for`, `do..while` o `while`) o en un bloque de sentencias asociadas con un determinado caso de una sentencia `switch`. La sentencia `break` indica a Flash Lite que omita el resto del cuerpo del bucle, detenga la acción del bucle y ejecute la sentencia que sigue a la del bucle. Cuando se utiliza la sentencia `break`, el intérprete de ActionScript omite el resto de las sentencias de ese bloque `case` y salta a la primera sentencia que sigue a la sentencia `switch`. Use esta sentencia para interrumpir una serie de bucles anidados.

Ejemplo

El ejemplo siguiente utiliza la sentencia `break` para salir de un bucle que, de otro modo, sería infinito:

```
i = 0;
while (true) {
    if (i >= 100) {
        break;
    }
    i++;
}
```

Véase también

[case](#), [do..while](#), [for](#), [switch](#), [while](#)

case

Disponibilidad

Flash Lite 1.0.

Uso

case expression: statements

Parámetros

expression Cualquier expresión.

statements Cualquier sentencia.

Descripción

Sentencia; define una condición para la sentencia `switch`. Las sentencias del parámetro *statements* se ejecutan si el parámetro *expression* que sigue a la palabra clave `case` coincide con el parámetro *expression* de la sentencia `switch`.

Si se utiliza la sentencia `case` fuera de una sentencia `switch`, se produce un error y el código no se compila.

Ejemplo

En el ejemplo siguiente, si el parámetro `myNum` devuelve 1, se ejecuta la sentencia `trace()` que sigue a `case 1`; si el parámetro `myNum` es 2, se ejecuta la sentencia `trace()` que sigue a `case 2`, y así sucesivamente. Si ninguna expresión `case` coincide con el parámetro `number`, se ejecuta la sentencia `trace()` que sigue a la palabra clave `default`.

```
switch (myNum) {
    case 1:
        trace ("case 1 tested true");
        break;
    case 2:
        trace ("case 2 tested true");
        break;
    case 3:
        trace ("case 3 tested true");
        break;
    default:
        trace ("no case tested true")
}
```

En el ejemplo siguiente, no se produce interrupción en el primer grupo, por lo que si el número es 1, aparecen A y B en el panel Salida:

```
switch (myNum) {
    case 1:
        trace ("A");
    case 2:
        trace ("B");
        break;
    default:
        trace ("D")
}
```

Véase también

[switch](#)

continue

Disponibilidad

Flash Lite 1.0.

Uso

`continue`

Parámetros

Ninguno.

Descripción

Sentencia; salta por encima de todas las sentencias restantes en el bucle interior e inicia la siguiente repetición del bucle como si se hubiera pasado el control hasta el final del bucle de la forma habitual. No tiene ningún efecto fuera de un bucle.

- En un bucle `while`, `continue` hace que el intérprete de Flash omita el resto del cuerpo del bucle y salte al principio del bucle, donde se prueba la condición.
- En un bucle `do..while`, `continue` hace que el intérprete de Flash omita el resto del cuerpo del bucle y salte al final del bucle, donde se prueba la condición.
- En un bucle `for`, `continue` hace que el intérprete de Flash omita el resto del cuerpo del bucle y salte a la evaluación de la expresión posterior del bucle `for`.

Ejemplo

En el siguiente bucle `while`, `continue` hace que Flash Lite omita el resto del cuerpo del bucle y salte al principio del bucle, donde se prueba la condición.

```
i = 0;
while (i < 10) {
    if (i % 3 == 0) {
        i++;
        continue;
    }
    trace(i);
    i++;
}
```

En el siguiente bucle `do..while`, `continue` hace que Flash Lite omita el resto del cuerpo del bucle y salte al final del bucle, donde se prueba la condición.

```
i = 0;
do {
    if (i % 3 == 0) {
        i++;
        continue;
    }
    trace(i);
    i++;
} while (i < 10);
```

En un bucle `for`, `continue` hace que Flash Lite omita el resto del cuerpo del bucle. En el ejemplo siguiente, si el módulo `i 3` es `0`, se omite la sentencia `trace(i)`:

```
for (i = 0; i < 10; i++) {
    if (i % 3 == 0) {
        continue;
    }
    trace(i);
}
```

Véase también

[do..while](#), [for](#), [while](#)

do..while

Disponibilidad

Flash Lite 1.0.

Uso

```
do {  
    statement(s)  
} while (condition)
```

Parámetros

statement(s) Las sentencias que se ejecutan si el parámetro *condition* es true.

condition La condición que se va a evaluar.

Descripción

Sentencia; ejecuta las sentencias y, a continuación, evalúa la condición en un bucle, siempre que sea true.

Ejemplo

El ejemplo siguiente incrementa la variable index siempre que el valor de la variable sea inferior a 10:

```
i = 0;  
do {  
    //trace (i);    // salida: 0,1,2,3,4,5,6,7,8,9  
    i++;  
} while (i<10);
```

Véase también

[break](#), [continue](#), [for](#), [while](#)

else

Disponibilidad

Flash Lite 1.0.

Uso

```
if (condition){  
    t-statement(s);  
} else {  
    f-statement(s);  
}
```

Parámetros

condition Expresión que devuelve true o false.

t-statement(s) Instrucciones que se ejecutan si la condición da como resultado true.

f-statement(s) Una serie alternativa de instrucciones que se ejecutan si la condición da como resultado false.

Descripción

Sentencia; especifica las sentencias que se ejecutarán si la condición en la sentencia `if` devuelve false.

Ejemplo

El ejemplo siguiente muestra el uso de la sentencia `else` con una condición. Un ejemplo real incluiría código para realizar alguna acción basada en la condición.

```
currentHighestDepth = 1;
if (currentHighestDepth == 2) {
    //trace ("currentHighestDepth is 2");
} else {
    //trace ("currentHighestDepth is not 2");
}
```

Véase también

[if](#)

else if

Disponibilidad

Flash Lite 1.0.

Uso

```
if (condition){
    statement(s);
} else if (condition){
    statement(s);
}
```

Parámetros

condition Expresión que devuelve true o false.

statement(s) Una serie de sentencias que se ejecutan si la condición especificada en la sentencia `if` es false.

Descripción

Sentencia; evalúa una condición y especifica las sentencias que se ejecutarán si la condición en la sentencia `if` inicial devuelve `false`. Si la condición `else if` devuelve un valor `true`, el intérprete de Flash ejecuta las sentencias que siguen a la condición `else if` entre llaves (`{}`). Si la condición `else if` es `false`, Flash omite las sentencias entre llaves y ejecuta las sentencias que hay a continuación. Utilice la sentencia `else if` para crear una lógica ramificada en los scripts.

Ejemplo

El ejemplo siguiente utiliza las sentencias `else if` para comprobar si cada lado de un objeto está dentro de un límite específico:

```
person_mc.xPos = 100;
leftBound = 0;
rightBound = 100;

if (person_mc.xPos <= leftBound) {
    //trace ("Clip is to the far left");
} else if (person_mc.xPos >= rightBound) {
    //trace ("Clip is to the far right");
} else {
    //trace ("Your clip is somewhere in between");
}
```

Véase también

[if](#)

for

Disponibilidad

Flash Lite 1.0.

Uso

```
for (init; condition; next) {
    statement(s);
}
```

Parámetros

init Expresión que se va a evaluar antes de iniciar la secuencia de bucle; normalmente es una expresión de asignación.

condition Expresión que devuelve `true` o `false`. La condición se evalúa antes de cada repetición del bucle; el bucle finaliza cuando la condición da como resultado `false`.

next Expresión que se va a evaluar después de cada repetición del bucle; normalmente es una expresión de asignación donde se utilizan los operadores de incremento (++) o decremento (--).

statement(s) Una o varias instrucciones para ejecutar en el bucle.

Descripción

Sentencia; un bucle que evalúa la expresión *init* (inicializar) una vez y, a continuación, inicia una secuencia de bucle por la que, siempre que *condition* sea *true*, se ejecuta *statement*, y se evalúa la sentencia siguiente

Las sentencias `for` o `for..in` no pueden enumerar algunas propiedades. Por ejemplo, las propiedades de clips de película, como `_x` e `_y`, no se incluyen.

Ejemplo

El ejemplo siguiente utiliza el bucle `for` para sumar números del 1 al 100.

```
sum = 0;
for (i = 1; i <= 100; i++) {
    sum = sum + i;
}
```

Véase también

[++ \(incremento\)](#), [-- \(decremento\)](#), [do..while](#), [while](#)

if

Disponibilidad

Flash Lite 1.0.

Uso

```
if (condition) {
    statement(s);
}
```

Parámetros

condition Expresión que devuelve *true* o *false*.

statement(s) Instrucciones que se ejecutarán si la condición da como resultado *true*.

Descripción

Sentencia; evalúa una condición para determinar la siguiente acción en un archivo SWF. Si la condición es `true`, Flash Lite ejecuta las sentencias que hay entre llaves (`{}`), a continuación de la condición. Si la condición es `false`, Flash Lite omite las sentencias entre llaves y ejecuta las sentencias que hay a continuación. Utilice la sentencia `else if` para crear una lógica ramificada en los scripts.

Ejemplo

En el ejemplo siguiente, la condición entre paréntesis evalúa la variable `name` para ver si tiene el valor literal "Erica". En caso de que así sea, se ejecuta la función `play()`.

```
if(name eq "Erica"){  
    play();  
}
```

switch

Disponibilidad

Flash Lite 1.0.

Uso

```
switch (expression){  
    caseClause:  
    [defaultClause:]  
}
```

Parámetros

expression Cualquier expresión numérica.

caseClause Palabra clave `case` seguida de una expresión, dos puntos y un grupo de sentencias que se ejecutan si la expresión es igual al parámetro *expression* de `switch`.

defaultClause Una palabra clave `default` seguida por sentencias que se ejecutan si ninguna de las expresiones `case` coinciden con el parámetro *expression* de `switch`.

Descripción

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

Las sentencias `switch` contienen una opción de reserva denominada `default`. Si no otra sentencia es `true`, se ejecuta la sentencia `default`.

Ejemplo

En el ejemplo siguiente, si el parámetro `myNum` devuelve 1, se ejecuta la sentencia `trace()` que sigue a `case 1`; si el parámetro `myNum` es 2, se ejecuta la sentencia `trace()` que sigue a `case 2`, y así sucesivamente. Si ninguna expresión `case` coincide con el parámetro `number`, se ejecuta la sentencia `trace()` que sigue a la palabra clave `default`.

```
switch (myNum) {
  case 1:
    trace ("case 1 tested true");
    break;
  case 2:
    trace ("case 2 tested true");
    break;
  case 3:
    trace ("case 3 tested true");
    break;
  default:
    trace ("no case tested true")
}
```

En el ejemplo siguiente, el primer grupo `case` no contiene una interrupción, por lo que si el número es 1, aparecen A y B en el panel Salida:

```
switch (myNum) {
  case 1:
    trace ("A");
  case 2:
    trace ("B");
    break;
  default:
    trace ("D")
}
```

Véase también

[case](#)

while

Disponibilidad

Flash Lite 1.0.

Uso

```
while(condition) {  
    statement(s);  
}
```

Parámetros

condition La expresión que se evalúa cada vez que se ejecuta la sentencia `while`.

statement(s) Las instrucciones que se ejecutan si la condición da como resultado `true`.

Descripción

Sentencia; prueba una expresión y ejecuta una sentencia o una serie de sentencias de forma repetida en un bucle, siempre que la expresión sea `true`

Antes de que se ejecute el bloque de sentencias, se prueba la condición, si el resultado es `true`, se ejecuta el bloque de sentencias. Si la condición es `false`, se omite el bloque de sentencias y se ejecuta la primera sentencia después del bloque `while`.

Los bucles suelen utilizarse para ejecutar una acción cuando la variable de contador (`counter`) es inferior a un valor especificado. Al final de cada bucle se incrementa el contador hasta que se alcanza el valor especificado. En dicho punto, la `condition` ya no es `true` y finaliza el bucle.

La sentencia `while` ejecuta la siguiente serie de pasos. Cada repetición de pasos del 1 al 4 se denomina *repetición* del bucle. La condición se prueba al principio de cada repetición:

1. Se evalúa la expresión `condition`.
2. Si `condition` devuelve `true` o un valor equivalente a un valor booleano `true`, como un número distinto de cero, continúe por el paso 3.

En caso contrario, la sentencia `while` finaliza y la ejecución se reanuda en la siguiente sentencia después del bucle `while`.

3. Ejecute el bloque de sentencias `statement(s)`.
4. Vaya al paso 1.

Ejemplo

El ejemplo siguiente ejecuta un bucle siempre que el valor de la variable de índice `i` sea inferior a 10:

```
i = 0;
while(i < 10) {
    trace ("i = " add ++i); // Output: 1,2,3,4,5,6,7,8,9
}
```

Véase también

[continue](#), [do..while](#), [for](#)

En esta sección se describe la sintaxis y el uso de los operadores de ActionScript de Macromedia Flash Lite 1.x de Adobe. Todas las entradas aparecen en la lista en orden alfabético. Sin embargo, algunos operadores son símbolos y se ordenan por sus descripciones. Los operadores de esta sección se resumen en la siguiente tabla:

Operador	Descripción
<code>add</code> (concatenación de cadenas)	Concatena (combina) dos o más cadenas.
<code>+=</code> (asignación de suma)	Asigna a <i>expression1</i> el valor de <i>expression1</i> + <i>expression2</i> .
<code>and</code>	Realiza una operación AND lógica.
<code>=</code> (asignación)	Asigna el valor de <i>expression2</i> (el operando a la derecha) a la variable o propiedad de <i>expression1</i> .
<code>/*</code> (comentario en bloque)	Indica una o varias líneas de comentarios de script. Los caracteres que aparecen entre la etiqueta de apertura de comentario (<code>/*</code>) y la etiqueta de cierre de comentario (<code>*/</code>) se interpretan como un comentario y el interpretador de ActionScript los omite.
<code>,</code> (coma)	Evalúa <i>expression1</i> , a continuación, <i>expression2</i> y devuelve el valor de <i>expression2</i> .
<code>//</code> (comentario)	Indica el principio de un comentario de script. Los caracteres que aparecen entre el delimitador de comentario (<code>//</code>) y el carácter de final de línea se interpretan como un comentario y el interpretador de ActionScript los omite.
<code>?:</code> (condicional)	Indica a Flash que evalúe <i>expression1</i> y si el valor de <i>expression1</i> es <code>true</code> , el operador devuelve el valor de <i>expression2</i> ; en caso contrario, devuelve el valor de <i>expression3</i> .

Operador	Descripción
-- (decremento)	<p>Resta 1 de <i>expression</i>. La forma de decremento previo del operador (<i>--expression</i>) resta 1 de <i>expression</i> y devuelve el resultado como un número.</p> <p>La forma de decremento posterior del operador (<i>expression--</i>) resta 1 de <i>expression</i> y devuelve el valor inicial de <i>expression</i> (el valor antes de la resta).</p>
/ (dividir)	Divide <i>expression1</i> entre <i>expression2</i> .
/= (asignación de división)	Asigna a <i>expression1</i> el valor de <i>expression1</i> / <i>expression2</i> .
. (punto)	Se utiliza para navegar por las jerarquías de clips de película y acceder a variables, propiedades o clips de película anidados (secundarios).
++ (incremento)	<p>Suma 1 a <i>expression</i>. La <i>expression</i> puede ser una variable, un elemento de una matriz o una propiedad de un objeto. La forma de incremento previo del operador (<i>++expression</i>) añade 1 a <i>expression</i> y devuelve el resultado como un número. La forma de incremento posterior del operador (<i>expression++</i>) añade 1 a <i>expression</i> y devuelve el valor inicial de <i>expression</i> (el valor antes de la suma).</p>
&& (AND lógico)	<p>Evalúa <i>expression1</i> (la expresión en la parte izquierda del operador) y devuelve <i>false</i> si la expresión da como resultado <i>false</i>. Si <i>expression1</i> da como resultado <i>true</i>, se evalúa <i>expression2</i> (la expresión en la parte derecha del operador). Si <i>expression2</i> da como resultado <i>true</i>, el resultado final es <i>true</i>; de lo contrario, es <i>false</i>.</p>
! (NOT lógico)	<p>Invierte el valor booleano de una variable o expresión. Si <i>expression</i> es una variable con el valor <i>true</i> absoluto o convertido, el valor de <i>!expression</i> es <i>false</i>. Si la expresión <i>x && y</i> devuelve <i>false</i>, la expresión <i>!(x && y)</i> devuelve <i>true</i>.</p>
(OR lógico)	<p>Evalúa <i>expression1</i> y <i>expression2</i>. El resultado es <i>true</i> si al menos una de las expresiones da como resultado <i>true</i>; el resultado es <i>false</i> sólo si ambas expresiones dan como resultado <i>false</i>. Puede utilizar el operador OR lógico con cualquier número de operandos; si alguno de los operandos da como resultado <i>true</i>, el resultado es <i>true</i>.</p>
% (módulo)	<p>Calcula el resto de <i>expression1</i> dividido entre <i>expression2</i>. Si un operando de <i>expression</i> no es numérico, el operador modulo intenta convertirlo a un número.</p>
%= (asignación de módulo)	Asigna a <i>expression1</i> el valor de <i>expression1</i> % <i>expression2</i> .
*= (asignación de multiplicación)	Asigna a <i>expression1</i> el valor de <i>expression1</i> * <i>expression2</i> .

Operador	Descripción
* (multiplicar)	Multiplica dos expresiones numéricas.
+ (suma numérica)	Suma expresiones numéricas.
== (igualdad numérica)	Comprueba la igualdad, si <i>expression1</i> es igual que <i>expression2</i> , el resultado es <i>true</i>
> (numérico mayor que)	Compara dos expresiones y determina si <i>expression1</i> es mayor que <i>expression2</i> ; si lo es, el operador devuelve <i>true</i> . Si <i>expression1</i> es menor o igual que <i>expression2</i> , el operador devuelve <i>false</i> .
>= (numérico mayor o igual que)	Compara dos expresiones y determina si <i>expression1</i> es mayor o igual que <i>expression2</i> (<i>true</i>) o si <i>expression1</i> es menor que <i>expression2</i> (<i>false</i>).
<> (desigualdad numérica)	Comprueba la desigualdad, si <i>expression1</i> es igual que <i>expression2</i> , el resultado es <i>false</i> .
< (numérico menor que)	Compara dos expresiones y determina si <i>expression1</i> es menor que <i>expression2</i> ; si lo es, el operador devuelve <i>true</i> . Si <i>expression1</i> es mayor o igual que <i>expression2</i> , el operador devuelve <i>false</i> .
<= (numérico menor o igual que)	Compara dos expresiones y determina si <i>expression1</i> es menor o igual que <i>expression2</i> . Si lo es, el operador devuelve <i>true</i> ; en caso contrario, devuelve <i>false</i> .
() (paréntesis)	Agrupar uno o varios parámetros, lleva a cabo una evaluación secuencial de las expresiones o rodea uno o varios parámetros y los pasa como parámetros a una función fuera del paréntesis.
" " (delimitador de cadena)	Si se utilizan antes y después de una secuencia de cero o más caracteres, las comillas indican que los caracteres tienen un valor literal y se consideran una <i>cadena</i> y no una variable ni un valor numérico u otro elemento de ActionScript.
eq (igualdad de cadenas)	Compara la igualdad de dos expresiones y devuelve <i>true</i> si la representación de cadena de <i>expression1</i> es igual que la representación de cadena de <i>expression2</i> ; en caso contrario, devuelve <i>false</i> .
gt (cadena mayor que)	Compara la representación de cadena de <i>expression1</i> con la representación de cadena de <i>expression2</i> y devuelve <i>true</i> si <i>expression1</i> es mayor que <i>expression2</i> ; en caso contrario, devuelve <i>false</i> .
ge (cadena mayor o igual que)	Compara la representación de cadena de <i>expression1</i> con la representación de cadena de <i>expression2</i> y devuelve <i>true</i> si <i>expression1</i> es mayor o igual que <i>expression2</i> ; en caso contrario, devuelve <i>false</i> .

Operador	Descripción
<code>ne</code> (desigualdad de cadenas)	Compara la representación de cadena de <i>expression1</i> con <i>expression2</i> y devuelve <code>true</code> si <i>expression1</i> no es igual que <i>expression2</i> ; en caso contrario, devuelve <code>false</code> .
<code>lt</code> (cadena menor que)	Compara la representación de cadena de <i>expression1</i> con la representación de cadena de <i>expression2</i> y devuelve <code>true</code> si <i>expression1</i> es menor que <i>expression2</i> ; en caso contrario, devuelve <code>false</code> .
<code>le</code> (cadena menor o igual que)	Compara la representación de cadena de <i>expression1</i> con la representación de cadena de <i>expression2</i> y devuelve <code>true</code> si <i>expression1</i> es menor o igual que <i>expression2</i> ; en caso contrario, devuelve <code>false</code> .
<code>-</code> (resta)	Se emplea para negar o restar.
<code>-=</code> (asignación de resta)	Asigna a <i>expression1</i> el valor de <i>expression1</i> - <i>expression2</i> .

add (concatenación de cadenas)

Disponibilidad

Flash Lite 1.0.

Uso

```
string1 add string2
```

Operandos

string1, *string2* Cadenas.

Descripción

Operador; concatena (combina) dos o más cadenas.

Ejemplo

El ejemplo siguiente combina dos valores de cadenas para producir la cadena *catálogo*.

```
conStr = "cat" add "alog";
trace (conStr); // salida: catálogo
```

Véase también

[+ \(suma numérica\)](#)

+= (asignación de suma)

Disponibilidad

Flash Lite 1.0.

Uso

expression1 += *expression2*

Operandos

expression1, *expression2* Números o cadenas.

Descripción

Operador (asignación compuesta aritmética); asigna a *expression1* el valor de *expression1* + *expression2*. Por ejemplo, las dos sentencias siguientes tienen el mismo resultado:

```
x += y;  
x = x + y;
```

Todas las reglas del operador de suma (+) se aplican al operador de asignación de suma (+=).

Ejemplo

El ejemplo siguiente utiliza el operador de asignación de suma (+=) para aumentar el valor de x por el valor de y:

```
x = 5;  
y = 10;  
x += y;  
trace(x); // salida: 15
```

Véase también

[+ \(suma numérica\)](#)

and

Disponibilidad

Flash Lite 1.0.

Uso

condition1 and *condition2*

Operandos

condition1, *condition2* Condiciones o expresiones que dan como resultado true o false.

Descripción

Operador; realiza una operación AND lógica.

Ejemplo

El ejemplo siguiente utiliza el operador `and` para probar si un jugador ha ganado el juego. La variable `turns` y la variable `score` se actualizan cuando un jugador juega o gana puntos durante el juego. El script muestra “You Win the Game!” en el panel Salida (ficha Salida) cuando la puntuación del jugador llega a 75 o más en tres jugadas o menos.

```
turns = 2;
score = 77;
winner = (turns <= 3) and (score >= 75);
if (winner) {
    trace("You Win the Game!");
} else {
    trace("Try Again!");
}
// salida: You Win the Game!
```

Véase también

[&& \(AND lógico\)](#)

= (asignación)

Disponibilidad

Flash Lite 1.0.

Uso

expression1 = *expression2*

Operandos

expression1 Una variable o una propiedad.

expression2 Un valor.

Descripción

Operador; asigna el valor de *expression2* (el operando a la derecha) a la variable o propiedad en *expression1*.

Ejemplo

El ejemplo siguiente utiliza el operador de asignación (=) para asignar un valor numérico a la variable `weight`:

```
weight = 5;
```

El ejemplo siguiente utiliza el operador de asignación (=) para asignar un valor de cadena a la variable `greeting`:

```
greeting = "Hello, " and personName;
```

/* (comentario en bloque)

Disponibilidad

Flash Lite 1.0

Uso

```
/* comment */  
/* comment  
comment */
```

Operandos

comment Cualquier carácter.

Descripción

Delimitador de comentario; indica una o varias líneas de comentarios de script. Los caracteres que aparecen entre la etiqueta de apertura de comentario (`/*`) y la etiqueta de cierre de comentario (`*/`) se interpretan como un comentario y el interpretador de ActionScript los omite.

Utilice el delimitador de comentario `//` para identificar los comentarios de una sola línea.

Utilice el delimitador de comentario `/*` para identificar los comentarios en varias líneas sucesivas. Si no se inserta la etiqueta de cierre (`*/`) cuando se utiliza esta forma de delimitador de comentario, se obtiene un mensaje de error. También se obtiene un mensaje de error si se intenta anidar comentarios.

Después de utilizar una etiqueta de apertura de comentario (`/*`), la primera etiqueta de cierre de comentario (`*/`) finalizará el comentario, independientemente del número de etiquetas de apertura (`/*`) que haya entre ambas.

Véase también

[// \(comentario\)](#)

, (coma)

Disponibilidad

Flash Lite 1.0.

Uso

expression1, *expression2*

Operandos

expression1, *expression2* Números o expresiones que devuelven números.

Descripción

Operador; evalúa *expression1*, a continuación, *expression2* y devuelve el valor de *expression2*.

Ejemplo

El ejemplo siguiente utiliza el operador coma (,) sin el operador paréntesis () y muestra cómo el operador coma devuelve únicamente el valor de la primera expresión sin el operador paréntesis ():

```
v = 0;
v = 4, 5, 6;
trace(v); // salida: 4
```

El ejemplo siguiente utiliza el operador coma (,) con el operador paréntesis () y muestra cómo el operador coma devuelve el valor de la última expresión cuando se utiliza con el operador paréntesis ():

```
v = 0;
v = (4, 5, 6);
trace(v); // salida: 6
```

El ejemplo siguiente utiliza el operador coma (,) sin el operador paréntesis () y muestra cómo el operador coma evalúa secuencialmente todas las expresiones, pero devuelve el valor de la primera expresión. Se evalúa la segunda expresión, z++, y z se incrementa en uno.

```
v = 0;
z = 0;
v = v + 4 , z++, v + 6;
trace(v); // salida: 4
trace(z); // salida: 1
```

El ejemplo siguiente es idéntico al ejemplo anterior, salvo por la adición del operador paréntesis () y muestra una vez más que, cuando se utiliza con el operador paréntesis (), el operador coma (,) devuelve el valor de la última expresión de la serie:

```
v = 0;
z = 0;
v = (v + 4, z++, v + 6);
trace(v); // salida: 6
trace(z); // salida: 1
```

Véase también

[for, \(\) \(paréntesis\)](#)

// (comentario)

Disponibilidad

Flash Lite 1.0

Uso

```
// comentario
```

Operandos

comment Cualquier carácter.

Descripción

Delimitador de comentario; indica el principio de un comentario de script. Los caracteres que aparecen entre el delimitador de comentario (//) y el carácter de final de línea se interpretan como un comentario y el interpretador de ActionScript los omite.

Ejemplo

El ejemplo siguiente utiliza delimitadores de comentario para identificar la primera, tercera, quinta y séptima líneas como comentarios:

```
// Registrar la posición X del clip de película ball.
ballX = ball._x;
// Registrar la posición Y del clip de película ball.
ballY = ball._y;
// Registrar la posición X del clip de película bat.
batX = bat._x;
// Registrar la posición Y del clip de película bat.
batY = bat._y;
```

Véase también

[/* \(comentario en bloque\)](#)

?: (condicional)

Disponibilidad

Flash Lite 1.0.

Uso

expression1 ? *expression2* : *expression3*

Operandos

expression1 Expresión que da como resultado un valor booleano; normalmente una expresión de comparación, como $x < 5$.

expression2, *expression3* Valores de cualquier tipo.

Descripción

Operador; indica a Flash que evalúe *expression1* y si el valor de *expression1* es true, devuelve el valor de *expression2*; en caso contrario, devuelve el valor de *expression3*.

Ejemplo

El ejemplo siguiente asigna el valor de variable x a la variable z porque *expression1* da como resultado true:

```
x = 5;  
y = 10;  
z = (x < 6) ? x : y;  
trace (z); // salida: 5
```

-- (decremento)

Disponibilidad

Flash Lite 1.0.

Uso

--*expression*

expression--

Operandos

Ninguno.

Descripción

Operador (aritmético); operador unario de decremento previo y decremento posterior que resta 1 de *expression*. La forma de decremento previo del operador (*--expression*) resta 1 de *expression* y devuelve el resultado como un número. La forma de decremento posterior del operador (*expression--*) resta 1 de *expression* y devuelve el valor inicial de *expression* (el valor antes de la resta).

Ejemplo

El ejemplo siguiente muestra la forma de decremento previo del operador; resta *aWidth* a 2 (*aWidth - 1 = 2*) y devuelve el resultado como *bWidth*:

```
aWidth = 3;
bWidth = --aWidth;
// El valor de bWidth es igual que 2.
```

El ejemplo siguiente muestra la forma de decremento posterior del operador; resta *aWidth* a 2 (*aWidth - 1 = 2*) y devuelve el valor original de *aWidth* como resultado *bWidth*:

```
aWidth = 3;
bWidth = aWidth--;
// El valor de bWidth es igual que 3.
```

/ (dividir)

Disponibilidad

Flash Lite 1.0.

Uso

```
expression1 / expression2
```

Operandos

expression1, *expression2* Números o expresiones que devuelven números.

Descripción

Operador (aritmético); divide *expression1* por *expression2*. El resultado de la operación de división es un número de coma flotante de doble precisión.

Ejemplo

La sentencia siguiente divide el número de coma flotante 22,0 entre 7,0 y muestra el resultado en el panel Salida:

```
trace(22.0 / 7.0);
```

El resultado, 3,1429, es un número de coma flotante.

/= (asignación de división)

Disponibilidad

Flash Lite 1.0.

Uso

expression1 /= expression2

Operandos

expression1, *expression2* Números o expresiones que devuelven números.

Descripción

Operador (asignación compuesta aritmética); asigna a *expression1* el valor de *expression1 / expression2*. Por ejemplo, las dos sentencias siguientes son equivalentes:

```
x /= y  
x = x / y
```

Ejemplo

El ejemplo siguiente utiliza el operador /= con variables y números:

```
x = 10;  
y = 2;  
x /= y;  
// La expresión x contiene ahora el valor 5.
```

. (punto)

Disponibilidad

Flash Lite 1.0.

Uso

instancename.variable

instancename.childinstance.variable

Operandos

instancename Nombre de instancia de un clip de película.

childinstance Nombre de instancia de un clip de película que es un elemento secundario o anidado de otro clip de película.

variable Variable de la línea de tiempo del nombre de instancia de clip de película especificado.

Descripción

Operador; se utiliza para navegar por las jerarquías de clips de película y acceder a variables, propiedades o clips de película anidados (secundarios).

Ejemplo

El ejemplo siguiente identifica el valor actual de la variable `hairColor` en el clip de película `person_mc`:

```
person_mc.hairColor
```

Es equivalente al siguiente código en notación de sintaxis con barras:

```
/person_mc:hairColor
```

Véase también

[/ \(Barra diagonal\)](#)

++ (incremento)

Disponibilidad

Flash Lite 1.0.

Uso

```
++expression
```

```
expression++
```

Operandos

Ninguno.

Descripción

Operador (aritmético); operador unario de incremento previo e incremento posterior que añade 1 a *expression*. La *expression* puede ser una variable, un elemento de una matriz o una propiedad de un objeto. La forma de incremento previo del operador (`++expression`) añade 1 a *expression* y devuelve el resultado como un número. La forma de incremento posterior del operador (`expression++`) añade 1 a *expression* y devuelve el valor inicial de *expression* (el valor antes de la suma).

Ejemplo

El ejemplo siguiente utiliza ++ como operador de incremento posterior para hacer que un bucle while se ejecute cinco veces:

```
i = 0;
while (i++ < 5){
    trace("this is execution " + i);
}
```

El ejemplo siguiente utiliza ++ como operador de incremento previo:

```
a = "";
i = 0;
while (i < 10) {
    a = a add (++i) add ",";
}
trace(a);// salida: 1,2,3,4,5,6,7,8,9,10,
```

Este script muestra el siguiente resultado en el panel Salida:

1,2,3,4,5,6,7,8,9,10,

El ejemplo siguiente utiliza ++ como operador de incremento posterior:

```
a = "";
i = 0;
while (i < 10) {
    a = a add (i++) add ",";
}
trace(a);// salida: 0,1,2,3,4,5,6,7,8,9,
```

Este script muestra el siguiente resultado en el panel Salida:

0,1,2,3,4,5,6,7,8,9,

&& (AND lógico)

Disponibilidad

Flash Lite 1.0.

Uso

expression1 && *expression2*

Operandos

expression1, *expression2* Valores booleanos o expresiones que se convierten a valores booleanos.

Descripción

Operador (lógico); ejecuta una operación Boolean en los valores de una o ambas expresiones. El operador evalúa *expression1* (la expresión en la parte izquierda del operador) y devuelve *false* si la expresión da como resultado *false*. Si *expression1* da como resultado *true*, se evalúa *expression2* (la expresión en la parte derecha del operador). Si *expression2* da como resultado *true*, el resultado final es *true*; de lo contrario, es *false*.

Ejemplo

El ejemplo siguiente utiliza el operador `&&` para realizar una prueba para determinar si un jugador ha ganado la partida. La variable `turns` y la variable `score` se actualizan cuando un jugador juega o gana puntos durante el juego. El script muestra “You Win the Game!” en el panel Salida (ficha Salida) cuando la puntuación del jugador llega a 75 o más en tres jugadas o menos.

```
turns = 2;
score = 77;
winner = (turns <= 3) && (score >= 75);
if (winner) {
    trace("You Win the Game!");
} else {
    trace("Try Again!");
}
```

El ejemplo siguiente ilustra la prueba realizada para averiguar si una posición imaginaria `_x` está en un rango.

```
xPos = 50;
if (xPos >= 20 && xPos <= 80) {
    trace("the xPos is in between 20 and 80");
}
```

!(NOT lógico)

Disponibilidad

Flash Lite 1.0.

Uso

`!expression`

Operandos

Ninguno.

Descripción

Operador (lógico); invierte el valor booleano de una variable o expresión. Si *expression* es una variable con el valor `true` absoluto o convertido, el valor de `!expression` es `false`. Si la expresión `x && y` devuelve `false`, la expresión `!(x && y)` devuelve `true`.

Las siguientes expresiones ilustran el resultado del uso del operador `!`:

```
!true devuelve false
```

```
!false devuelve true
```

Ejemplo

En el ejemplo siguiente, la variable `happy` se establece como `false`. La condición `if` evalúa la condición `!happy` y si es `true`, la función `trace()` envía una cadena al panel Salida.

```
happy = false;
if (!happy) {
    trace("don't worry, be happy");
}
```

|| (OR lógico)

Disponibilidad

Flash Lite 1.0.

Uso

```
expression1 || expression2
```

Operandos

expression1, *expression2* Valores booleanos o expresiones que se convierten a valores booleanos.

Descripción

Operador (lógico); evalúa *expression1* y *expression2*. El resultado es `true` si al menos una de las expresiones da como resultado `true`; el resultado es `false` sólo si ambas expresiones dan como resultado `false`. Puede utilizar el operador OR lógico con cualquier número de operandos; si alguno de los operandos da como resultado `true`, el resultado es `true`.

Con expresiones no booleanas, el operador OR lógico hace que Flash Lite evalúe la expresión de la izquierda; si puede convertirse a `true`, el resultado es `true`. En caso contrario, evalúa la expresión de la derecha y el resultado es el valor de dicha expresión.

Ejemplo

Sintaxis 1: El ejemplo siguiente utiliza el operador `||` en una sentencia `if`. La segunda expresión da como resultado `true`, por lo que el resultado final es `true`:

```
theMinimum = 10;
theMaximum = 250;
start = false;
if (theMinimum > 25 || theMaximum > 200 || start){
    trace("the logical OR test passed");
}
```

% (módulo)

Disponibilidad

Flash Lite 1.0.

Uso

expression1 % *expression2*

Operandos

expression1, *expression2* Números o expresiones que devuelven números.

Descripción

Operador (aritmético); calcula el resto de *expression1* dividido por *expression2*. Si un operando de *expression* no es valor numérico, el operador modulo intenta convertirlo a un número. La expresión puede ser un número o una cadena que se convierte en un valor numérico.

Cuando el destino es Flash Lite 1.0 o 1.1, el compilador de Flash amplía el operador % en el archivo SWF publicado mediante la siguiente fórmula:

$$expression1 - \text{int}(expression1 / expression2) * expression2$$

Es posible que el resultado de esta aproximación no sea tan rápido o preciso como con versiones de Flash Player que admiten originariamente el operador modulo.

Ejemplo

El código siguiente muestra un ejemplo numérico que utiliza el operador (%):

```
trace (12 % 5); // salida: 2
trace (4.3 % 2.1); // salida: 0.0999...
```

%= (asignación de módulo)

Disponibilidad

Flash Lite 1.0.

Uso

expression1 %= *expression2*

Operandos

expression1, *expression2* Números o expresiones que devuelven números.

Descripción

Operador (asignación compuesta aritmética); asigna a *expression1* el valor de *expression1* % *expression2*. Por ejemplo, las dos expresiones siguientes son equivalentes:

```
x %= y  
x = x % y
```

Ejemplo

El ejemplo siguiente asigna el valor de 4 a la variable *x*:

```
x = 14;  
y = 5;  
trace(x %= y); // salida: 4
```

Véase también

[% \(módulo\)](#)

*= (asignación de multiplicación)

Disponibilidad

Flash Lite 1.0.

Uso

expression1 *= *expression2*

Operandos

expression1, *expression2* Números o expresiones que devuelven números.

Descripción

Operador (asignación compuesta aritmética); asigna a *expression1* el valor de *expression1* * *expression2*.

Por ejemplo, las dos expresiones siguientes son equivalentes:

```
x *= y
x = x * y
```

Ejemplo

Sintaxis 1: El ejemplo siguiente asigna el valor de 50 a la variable x:

```
x = 5;
y = 10;
trace(x *= y); // salida: 50
```

Sintaxis 2: Las líneas segunda y tercera del ejemplo siguiente calculan las expresiones de la parte derecha del signo igual (=) y asignan los resultados a x e y:

```
i = 5;
x = 4 - 6;
y = i + 2;
trace(x *= y); // salida: -14
```

* (multiplicar)

Disponibilidad

Flash Lite 1.0.

Uso

expression1 * *expression2*

Operandos

expression1, *expression2* Expresiones numéricas.

Descripción

Operador (aritmético); multiplica dos expresiones numéricas. Si ambas expresiones son enteros, el producto es un entero. Si una o ambas expresiones son números de coma flotante, el producto es un número de coma flotante.

Ejemplo

Sintaxis 1: La sentencia siguiente multiplica los enteros 2 y 3:

```
2 * 3
```

El resultado, 6, es un entero.

Sintaxis 2: Esta sentencia multiplica los números de coma flotante 2,0 y 3,1416:

```
2.0 * 3.1416
```

El resultado, 6,2832, es un número de coma flotante.

+ (suma numérica)

Disponibilidad

Flash Lite 1.0.

Uso

```
expression1 + expression2
```

Operandos

```
expression1, expression2  Números.
```

Descripción

Operador; suma expresiones numéricas. + es un operador numérico solamente; no puede utilizarse para concatenación de cadenas.

Si ambas expresiones son enteros, la suma es un entero; si al menos una de las expresiones es un número de coma flotante, la suma es un número de coma flotante.

Ejemplo

El ejemplo siguiente añade los enteros 2 y 3; el entero resultante, 5, aparece en el panel Salida:

```
trace (2 + 3);
```

El ejemplo siguiente suma los números de coma flotante 2,5 y 3,25; el resultado, 5,75, un número de coma flotante, aparece en el panel Salida:

```
trace (2.5 + 3.25);
```

Véase también

[add \(concatenación de cadenas\)](#)

== (igualdad numérica)

Disponibilidad

Flash Lite 1.0.

Uso

expression1 == *expression2*

Operandos

expression1, *expression2* Números, valores booleanos o variables.

Descripción

Operador (comparación); prueba la igualdad; el contrario exacto del operador <>.

Si *expression1* es igual que *expression2*, el resultado es *true*. Como ocurre con el operador <>, la definición de *igualdad* depende de los tipos de datos que se comparan:

- Los números y valores booleanos se comparan por su valor.
- Las variables se comparan por referencia.

Ejemplo

Los siguientes ejemplos muestran los valores *true* y *false*:

```
trees = 7;
bushes = "7";
shrubs = "seven";

trace (trees == "7");// salida: 1(true)
trace (trees == bushes);// salida: 1(true)
trace (trees == shrubs);// salida: 0(false)
```

Véase también

[eq \(igualdad de cadenas\)](#)

> (numérico mayor que)

Disponibilidad

Flash Lite 1.0.

Uso

expression1 > *expression2*

Operandos

expression1, *expression2* Números o expresiones que devuelven números.

Operador (comparación); compara dos expresiones y determina si *expression1* es mayor que *expression2*; si lo es, el operador devuelve `true`. Si *expression1* es menor o igual que *expression2*, el operador devuelve `false`.

Ejemplo

Los siguientes ejemplos muestran los valores `true` y `false` para comparaciones numéricas:

```
trace(3.14 > 2);// salida: 1(true)
trace(1 > 2);// salida: 0(false)
```

Véase también

[gt](#) (cadena mayor que)

>= (numérico mayor o igual que)

Disponibilidad

Flash Lite 1.0.

Uso

```
expression1 >= expression2
```

Operandos

expression1, *expression2* Enteros o números de coma flotante.

Descripción

Operador (comparación); compara dos expresiones y determina si *expression1* es mayor o igual que *expression2* (`true`) o si *expression1* es menor que *expression2* (`false`).

Ejemplo

Los siguientes ejemplos muestran los resultados `true` y `false`:

```
trace(3.14 >= 2);// salida: 1(true)
trace(3.14 >= 4);// salida: 0(false)
```

Véase también

[ge](#) (cadena mayor o igual que)

⟨⟩ (desigualdad numérica)

Disponibilidad

Flash Lite 1.0.

Uso

expression1 <> *expression2*

Operandos

expression1, *expression2* Números, valores booleanos o variables.

Descripción

Operador (comparación); prueba de desigualdad; el contrario exacto del operador de igualdad (==). Si *expression1* es igual que *expression2*, el resultado es `false`. Como ocurre con el operador de igualdad (==), la definición de *igualdad* depende de los tipos de datos que se comparan:

- Los números y valores booleanos se comparan por su valor.
- Las variables se comparan por referencia.

Ejemplo

Los siguientes ejemplos muestran los resultados `true` y `false`:

```
trees = 7;  
B = "7";
```

```
trace(trees <> 3);// salida: 1(true)  
trace(trees <> B);// salida: 0(false)
```

Véase también

[ne \(desigualdad de cadenas\)](#)

< (numérico menor que)

Disponibilidad

Flash Lite 1.0.

Uso

expression1 < *expression2*

Operandos

expression1, *expression2* Números.

Descripción

Operador (comparación); compara dos expresiones y determina si *expression1* es menor que *expression2*; si lo es, el operador devuelve `true`. Si *expression1* es mayor o igual que *expression2*, el operador devuelve `false`. `<` (menor que) es un operador numérico.

Ejemplo

Los ejemplos siguientes muestran resultados de `true` y `false` para comparaciones numéricas y de cadena:

```
trace (3 < 10);// salida: 1(true)
```

```
trace (10 < 3);// salida: 0(false)
```

Véase también

[lt \(cadena menor que\)](#)

<= (numérico menor o igual que)

Flash Lite 1.0.

Uso

```
expression1 <= expression2
```

Operandos

expression1, *expression2* Números.

Descripción

Operador (comparación); compara dos expresiones y determina si *expression1* es menor o igual que *expression2*. Si lo es, el operador devuelve `true`; en caso contrario, devuelve `false`. Este operador sólo se utiliza en comparaciones numéricas.

Ejemplo

Los siguientes ejemplos muestran los valores `true` y `false` para comparaciones numéricas:

```
trace(5 <= 10);// salida: 1(true)
```

```
trace(2 <= 2);// salida: 1(true)
```

```
trace (10 <= 3);// salida: 0(false)
```

Véase también

[le \(cadena menor o igual que\)](#)

() (paréntesis)

Disponibilidad

Flash Lite 1.0.

Uso

(expression1 [, expression2])
(expression1, expression2)

expression1, expression2 Números, cadenas, variables o texto.

parameter1, ..., parameterN Una serie de parámetros que se ejecutará antes de que se pasen los resultados como parámetros a la función que está fuera del paréntesis.

Descripción

Operador; agrupa uno o varios parámetros, lleva a cabo una evaluación secuencial de las expresiones o rodea uno o varios parámetros y los pasa como parámetros a una función fuera del paréntesis.

Sintaxis 1: Controla el orden de ejecución de los operadores en la expresión. Los paréntesis sustituyen el orden de precedencia normal y pueden hacer que las expresiones entre paréntesis se evalúen primero. Cuando se anidan los paréntesis, el contenido de los paréntesis más interiores se evalúa antes que el contenido de los más exteriores.

Sintaxis 2: Da como resultado una serie de expresiones, separadas por comas, en una secuencia y devuelve el resultado de la expresión final.

Ejemplo

Sintaxis 1: Las sentencias siguientes muestran el uso de los paréntesis para controlar el orden de ejecución de las expresiones (el valor de cada expresión aparece en el panel Salida):

```
trace((2 + 3) * (4 + 5)); // muestra 45  
trace(2 + (3 * (4 + 5))); // // muestra 29  
trace(2 + (3 * 4) + 5); // muestra 19
```

Sintaxis 1: Las sentencias siguientes muestran el uso de los paréntesis para controlar el orden de ejecución de las expresiones (el valor de cada expresión se escribe en el archivo de registro):

```
trace((2 + 3) * (4 + 5)); // escribe 45  
trace(2 + (3 * (4 + 5))); // escribe 29  
trace(2 + (3 * 4) + 5); // escribe 19
```

" " (delimitador de cadena)

Disponibilidad

Flash Lite 1.0.

Uso

`"text"`

Operandos

text Cero o más caracteres.

Descripción

Delimitador de cadena; cuando se utiliza delante y detrás de una secuencia de cero o más caracteres, las comillas indican que los caracteres tienen un valor literal y se consideran una *cadena* y no una variable ni un valor numérico u otro elemento de ActionScript.

Ejemplo

El ejemplo siguiente utiliza comillas para indicar que el valor de la variable `yourGuess` es la cadena literal "Prince Edward Island" y no el nombre de una variable. El valor de `province` es una variable, no un literal; para determinar el valor de `province` es necesario localizar el valor de `yourGuess`.

```
yourGuess = "Prince Edward Island";

on(release){
    province = yourGuess;
    trace(province);// salida: Prince Edward Island
}
```

eq (igualdad de cadenas)

Disponibilidad

Flash Lite 1.0.

Uso

expression1 eq *expression2*

Operandos

expression1, *expression2* Números, cadenas o variables.

Descripción

Operador de comparación; compara la igualdad de dos expresiones y devuelve `true` si la representación de cadena de *expression1* es igual que la representación de cadena de *expression2*; en caso contrario, devuelve `false`.

Ejemplo

Los siguientes ejemplos muestran los resultados `true` y `false`:

```
word = "persons";  
figure = "55";  
  
trace("persons" eq "people");// salida: 0(false)  
trace("persons" eq word);// salida: 1(true)  
trace(figure eq 50 + 5);// salida: 1(true)  
trace(55.0 eq 55);// salida: 1(true)
```

Véase también

[== \(igualdad numérica\)](#)

gt (cadena mayor que)

Disponibilidad

Flash Lite 1.0.

Uso

```
expression1 gt expression2
```

Operandos

expression1, *expression2* Números, cadenas o variables.

Descripción

Operador (comparación); compara la representación de cadena de *expression1* con la representación de cadena de *expression2* y devuelve `true` si *expression1* es mayor que *expression2*; en caso contrario, devuelve `false`. Las cadenas se comparan en orden alfabético; los dígitos antes que las letras y todas las letras mayúsculas antes que las minúsculas.

Ejemplo

Los siguientes ejemplos muestran los resultados `true` y `false`:

```
animals = "cats";
breeds = 7;

trace ("persons" gt "people");// salida: 1(true)
trace ("cats" gt "cattle");// salida: 0(false)
trace (animals gt "cats");// salida: 0(false)
trace (animals gt "Cats");// salida: 1(true)
trace (breeds gt "5");// salida: 1(true)
trace (breeds gt 7);// salida: 0(false)
```

Véase también

> [\(numérico mayor que\)](#)

ge (cadena mayor o igual que)

Disponibilidad

Flash Lite 1.0.

Uso

expression1 ge *expression2*

Operandos

expression1, *expression2* Números, cadenas o variables.

Descripción

Operador (comparación); compara la representación de cadena de *expression1* con la representación de cadena de *expression2* y devuelve `true` si *expression1* es mayor o igual que *expression2*; en caso contrario, devuelve `false`. Las cadenas se comparan en orden alfabético; los dígitos antes que las letras y todas las letras mayúsculas antes que las minúsculas.

Ejemplo

Los siguientes ejemplos muestran los resultados `true` y `false`:

```
animals = "cats";
breeds = 7;

trace ("cats" ge "cattle");// salida: 0(false)
trace (animals ge "cats");// salida: 1(true)
trace ("persons" ge "people");// salida: 1(true)
trace (animals ge "Cats");// salida: 1(true)
trace (breeds ge "5");// salida: 1(true)
trace (breeds ge 7);// salida: 1(true)
```

Véase también

[>= \(numérico mayor o igual que\)](#)

ne (desigualdad de cadenas)

Disponibilidad

Flash Lite 1.0.

Uso

expression1 ne *expression2*

Operandos

expression1, *expression2* Números, cadenas o variables.

Descripción

Operador (comparación); compara la representación de cadena de *expression1* con la representación de cadena de *expression2* y devuelve `true` si *expression1* no es igual que *expression2*; en caso contrario, devuelve `false`.

Ejemplo

Los siguientes ejemplos muestran los resultados `true` y `false`:

```
word = "persons";  
figure = "55";
```

```
trace ("persons" ne "people");// salida: 1(true)  
trace ("persons" ne word);// salida: 0(false)  
trace (figure ne 50 + 5);// salida: 0(false)  
trace (55.0 ne 55); // salida: 0(false)
```

Véase también

[<> \(desigualdad numérica\)](#)

lt (cadena menor que)

Disponibilidad

Flash Lite 1.0.

Uso

expression1 lt *expression2*

Operandos

expression1, *expression2* Números, cadenas o variables.

Descripción

Operador (comparación); compara la representación de cadena de *expression1* con la representación de cadena de *expression2* y devuelve *true* si *expression1* es menor que *expression2*; en caso contrario, devuelve *false*. Las cadenas se comparan en orden alfabético; los dígitos antes que las letras y todas las letras mayúsculas antes que las minúsculas.

Ejemplo

Los siguientes ejemplos muestran la salida de distintas comparaciones de cadenas. En la última línea, observe que `lt` no devuelve un error cuando compara una cadena con un entero debido a que la sintaxis de ActionScript 1.0 intenta convertir el entero en una cadena y devuelve *false*.

```
animals = "cats";
breeds = 7;

trace ("persons" lt "people");// salida: 0(false)
trace ("cats" lt "cattle");// salida: 1(true)
trace (animals lt "cats");// salida: 0(false)
trace (animals lt "Cats");// salida: 0(false)
trace (breeds lt "5");// salida: 0(false)
trace (breeds lt 7);// salida: 0(false)
```

Véase también

[< \(numérico menor que\)](#)

le (cadena menor o igual que)

Disponibilidad

Flash Lite 1.0.

Uso

expression1 le *expression2*

Operandos

expression1, *expression2* Números, cadenas o variables.

Descripción

Operador (comparación); compara la representación de cadena de *expression1* con la representación de cadena de *expression2* y devuelve *true* si *expression1* es menor o igual que *expression2*; en caso contrario, devuelve *false*. Las cadenas se comparan en orden alfabético; los dígitos antes que las letras y todas las letras mayúsculas antes que las minúsculas.

Ejemplo

Los siguientes ejemplos muestran la salida de distintas comparaciones de cadenas:

```
animals = "cats";
breeds = 7;

trace ("persons" le "people");// salida: 0(false)
trace ("cats" le "cattle");// salida: 1(true)
trace (animals le "cats");// salida: 1(true)
trace (animals le "Cats");// salida: 0(false)
trace (breeds le "5");// salida: 0(false)
trace (breeds le 7);// salida: 1(true)
```

Véase también

[<= \(numérico menor o igual que\)](#)

- (resta)

Disponibilidad

Flash Lite 1.0.

Uso

(Negación) *-expression*

(Resta) *expression1 - expression2*

Operandos

expression1, *expression2* Números o expresiones que devuelven números.

Descripción

Operador (aritmético); se utiliza para negar o restar.

Sintaxis 1: Si se utiliza para negar, invierte el signo de la expresión numérica.

Sintaxis 2: Si se utiliza para restar, ejecuta una resta aritmética en dos expresiones numéricas, restando *expression2* de *expression1*. Si ambas expresiones son enteros, la diferencia es un entero. Si una o ambas expresiones son números de coma flotante, la diferencia es un número de coma flotante.

Ejemplo

Sintaxis 1: La sentencia siguiente invierte el signo de la expresión $2 + 3$:

```
trace(-(2 + 3));  
// salida: -5.
```

Sintaxis 2: La sentencia siguiente resta el entero 2 del entero 5:

```
trace(5 - 2);  
// salida: 3.
```

El resultado, 3, es un entero.

Sintaxis 3: La sentencia siguiente resta el número de coma flotante 1,5 del número de coma flotante 3,25:

```
trace(3.25 - 1.5);  
// salida: 1.75.
```

El resultado, 1,75, es un número de coma flotante.

-= (asignación de resta)

Disponibilidad

Flash Lite 1.0.

Uso

expression1 -= *expression2*

Operandos

expression1, *expression2* Números o expresiones que devuelven números.

Descripción

Operador (asignación compuesta aritmética); asigna a *expression1* el valor de *expression1* - *expression2*. No se devuelve ningún valor.

Por ejemplo, las dos sentencias siguientes son equivalentes:

```
x -= y;  
x = x - y;
```

Las expresiones de cadenas deben convertirse a números; en caso contrario, el resultado es -1.

Ejemplo

Sintaxis 1: El ejemplo siguiente utiliza el operador de asignación de resta -= para restarle 10 a 5 y asigna el resultado a la variable x:

```
x = 2;  
y = 3;  
x -= y  
trace(x);// salida: -1
```

Sintaxis 2: El ejemplo siguiente muestra cómo se convierten las cadenas en números:

```
x = "2";  
y = "5";  
x -= y;  
trace(x);// salida: -3
```


Elementos del lenguaje específicos de Flash Lite

En esta sección se describen las funciones de la plataforma y las variables que reconoce Macromedia Flash Lite 1.1 de Adobe, así como los comandos de Flash Lite que puede ejecutar utilizando las funciones `fscommand()` y `fscommand2()`. La funcionalidad descrita en esta sección es específica para Flash Lite.

El contenido de esta sección se resume en la siguiente tabla:

Elemento del lenguaje	Descripción
<code>_capCompoundSound</code>	Indica si Flash Lite puede procesar datos de sonido compuesto.
<code>_capEmail</code>	Indica si el cliente de Flash Lite puede enviar mensajes de correo electrónico utilizando el comando <code>GetURL()</code> de ActionScript.
<code>_capLoadData</code>	Indica si la aplicación host puede cargar datos adicionales de forma dinámica a través de llamadas a las funciones <code>loadMovie()</code> , <code>loadMovieNum()</code> , <code>loadVariables()</code> y <code>loadVariablesNum()</code> .
<code>_capMFi</code>	Indica si el dispositivo puede reproducir datos de sonido en formato de audio MFi (Melody Format for i-mode, Formato de melodía para modo i).
<code>_capMIDI</code>	Indica si el dispositivo puede reproducir datos de sonido en formato de audio MIDI (Musical Instrument Digital Interface, Interfaz Digital de Instrumentos Musicales).
<code>_capMMS</code>	Indica si Flash Lite puede enviar mensajes MMS (Multimedia Messaging Service, Servicio de Mensajería Multimedia) mediante el comando <code>GetURL()</code> de ActionScript.
<code>_capMP3</code>	Indica si el dispositivo puede reproducir datos de sonido en formato de audio MPEG (<i>MPEG Audio Layer 3</i> , Audio MPEG Capa 3).
<code>_capSMAF</code>	Indica si el dispositivo puede reproducir archivos multimedia en formato SMAF (Synthetic music Mobile Application Format, Formato de Aplicación Móvil de Música Sintética).

Elemento del lenguaje Descripción

<code>_capSMS</code>	Indica si Flash Lite puede enviar mensajes SMS (Short Message Service, Servicio de Mensajes Cortos) mediante el comando <code>GetURL()</code> de ActionScript.
<code>_capStreamSound</code>	Indica si el dispositivo puede reproducir sonido sin interrupción (sincronizado).
<code>_cap4WayKeyAS</code>	Indica si Flash Lite ejecuta expresiones ActionScript asociadas a controladores de eventos de las teclas de flecha derecha, izquierda, arriba y abajo.
<code>\$version</code>	Contiene el número de versión de Flash Lite.
<code>fscommand()</code>	Una función utilizada para ejecutar el comando <code>Launch</code> (a continuación).
<code>Launch</code>	(El único comando que admite <code>fscommand()</code>) Permite al archivo SWF comunicarse con Flash Lite o con el entorno host, como el sistema operativo del teléfono o el dispositivo.
<code>fscommand2()</code>	Una función utilizada para ejecutar el resto de comandos de esta tabla, excepto <code>fscommand()</code> .
<code>Escape</code>	Codifica una cadena arbitraria en un formato seguro para la transferencia en red.
<code>FullScreen</code>	Establece el tamaño del área de visualización que se utilizará en la representación.
<code>GetBatteryLevel</code>	Devuelve el nivel de batería actual.
<code>GetDateDay</code>	Devuelve el día de la fecha actual como un valor numérico.
<code>GetDateMonth</code>	Devuelve el mes de la fecha actual como un valor numérico.
<code>GetDateWeekday</code>	Devuelve el número del día de la fecha actual como un valor numérico.
<code>GetDateYear</code>	Devuelve un valor número de cuatro dígitos correspondiente al año de la fecha actual.
<code>GetDevice</code>	Establece un parámetro que identifica el dispositivo en el que se ejecuta Flash Lite.
<code>GetDeviceID</code>	Define un parámetro que representa el identificador exclusivo del dispositivo (por ejemplo, el número de serie).
<code>GetFreePlayerMemory</code>	Devuelve la cantidad de memoria de pila disponible para Flash Lite, expresada en kilobytes.
<code>GetLanguage</code>	Establece un parámetro que identifica el idioma que se utiliza en el dispositivo.

Elemento del lenguaje	Descripción
-----------------------	-------------

GetLocaleLongDate	Define un parámetro para una cadena que representa la fecha actual, en formato largo, según la configuración regional seleccionada.
GetLocaleShortDate	Define un parámetro para una cadena que representa la fecha actual, en formato corto, según la configuración regional seleccionada.
GetLocaleTime	Define un parámetro para una cadena que representa la hora actual, según la configuración regional seleccionada.
GetMaxBatteryLevel	Devuelve el nivel máximo de batería del dispositivo.
GetMaxSignalLevel	Devuelve el nivel máximo de intensidad de la señal.
GetMaxVolumeLevel	Devuelve el nivel máximo de volumen del dispositivo como un valor numérico.
GetNetworkConnectStatus	Devuelve un valor que indica el estado de la conexión de red activa.
GetNetworkName	Establece un parámetro para el nombre de la red actual.
GetNetworkRequestStatus	Devuelve un valor que indica el estado de la solicitud HTTP más reciente.
GetNetworkStatus	Devuelve un valor que indica el estado de la red telefónica (es decir, si hay una red registrada y si el teléfono está lejos de la red doméstica).
GetPlatform	Define un parámetro que identifica la plataforma actual, que describe ampliamente la clase de dispositivo. Para los dispositivos con sistemas operativos abiertos, este identificador es normalmente el nombre y la versión del sistema operativo.
GetPowerSource	Devuelve un valor que indica si la fuente de alimentación se obtiene de una batería o de una fuente externa.
GetSignalLevel	Devuelve la intensidad de la señal actual como un valor numérico.
GetTimeHours	Devuelve el valor de hora de la hora actual del día, basada en un reloj de 24 horas como un valor numérico.
GetTimeMinutes	Devuelve el minuto de la hora actual del día como un valor numérico.
GetTimeSeconds	Devuelve el segundo de la hora actual del día como un valor numérico.
GetTimeZoneOffset	Define un parámetro como el número de minutos de diferencia entre la zona horaria local y la hora universal (UTC).

Elemento del lenguaje Descripción

<code>GetTotalPlayerMemory</code>	Devuelve la cantidad de memoria de pila total asignada a Flash Lite, expresada en kilobytes.
<code>GetVolumeLevel</code>	Devuelve el nivel actual del volumen del dispositivo como un valor numérico.
<code>Quit</code>	Hace que el reproductor de Flash Lite detenga la reproducción y se cierre.
<code>ResetSoftKeys</code>	Restablece la configuración original de las teclas programables.
<code>SetInputTextType</code>	Especifica el modo en que debería abrirse el campo de texto de introducción.
<code>SetQuality</code>	Define la calidad de la representación de la animación.
<code>SetSoftKeys</code>	Cambia la asignación de las teclas programables izquierda y derecha del dispositivo, siempre que se pueda acceder a ellas y sea posible.
<code>StartVibrate</code>	Inicia la vibración del teléfono.
<code>StopVibrate</code>	Detiene la vibración del teléfono, si está activa.
<code>Unescape</code>	Descodifica a su formato normal una cadena arbitraria que se codificó para protegerla durante transferencias en red.

Capacidades

En esta sección se describen las capacidades y variables de la plataforma que reconoce Macromedia Flash Lite 1.1. Las entradas se muestran en orden alfabético, ignorando los caracteres de subrayado iniciales.

_capCompoundSound

Disponibilidad

Flash Lite 1.1.

Uso

`_capCompoundSound`

Descripción

Variable numérica; indica si Flash Lite puede procesar datos de sonido compuesto. En caso afirmativo, esta variable se define con el valor 1; en caso contrario, será undefined.

Por ejemplo, un solo archivo Flash puede contener el mismo sonido representado en formato MIDI y MFi. El reproductor reproducirá los datos en el formato que admita el dispositivo. Esta variable define si el reproductor de Flash Lite admite esta función en el teléfono actual.

En el ejemplo siguiente, `useCompoundSound` se define como 1 en Flash Lite 1.1, pero se deja sin definir en Flash Lite 1.0:

```
useCompoundSound = _capCompoundSound;
```

```
if (useCompoundSound == 1) {  
    gotoAndPlay("withSound");  
} else {  
    gotoAndPlay("withoutSound");  
}
```

_capEmail

Disponibilidad

Flash Lite 1.1.

Uso

`_capEmail`

Descripción

Variable numérica; indica si el cliente de Flash Lite puede enviar mensajes de correo electrónico utilizando el comando `GetURL()` de ActionScript. En caso afirmativo, esta variable se define con el valor 1; en caso contrario, será `undefined`.

Ejemplo

Si la aplicación host puede enviar mensajes de correo electrónico con el comando `GetURL()` de ActionScript, el ejemplo siguiente define `canEmail` como 1:

```
canEmail = _capEmail;

if (canEmail == 1) {
    getURL("mailto:someone@somewhere.com?subject=foo&body=bar");
}
```

_capLoadData

Disponibilidad

Flash Lite 1.1.

Uso

`_capLoadData`

Descripción

Variable numérica; indica si la aplicación host puede cargar datos adicionales de forma dinámica a través de llamadas a las funciones `loadMovie()`, `loadMovieNum()`, `loadVariables()` y `loadVariablesNum()`. En caso afirmativo, esta variable se define con el valor 1; en caso contrario, será `undefined`.

Ejemplo

Si la aplicación host puede cargar películas y variables de forma dinámica, el ejemplo siguiente define `iCanLoad` como 1:

```
canLoad = _capLoadData;

if (canLoad == 1) {
    loadVariables("http://www.somewhere.com/myVars.php", GET);
} else {
    trace("client does not support loading dynamic data");
}
```

_capMFi

Disponibilidad

Flash Lite 1.1.

Uso

_capMFi

Descripción

Variable numérica; indica si el dispositivo puede reproducir datos de sonido en formato de audio MFi (Melody Format for i-mode, Formato de melodía para modo i). En caso afirmativo, esta variable se define con el valor 1; en caso contrario, será undefined.

Ejemplo

Si el dispositivo puede reproducir datos de sonido MFi, el ejemplo siguiente define `canMFi` como 1:

```
canMFi = _capMFi;

if (canMFi == 1) {
    // enviar botones de clip de película a fotogramas con botones que
    // activan sonidos de eventos
    tellTarget("buttons") {
        gotoAndPlay(2);
    }
}
```

_capMIDI

Disponibilidad

Flash Lite 1.1.

Uso

_capMIDI

Descripción

Variable numérica; indica si el dispositivo puede reproducir datos de sonido en formato de audio MIDI (Musical Instrument Digital Interface, Interfaz Digital de Instrumentos Musicales). En caso afirmativo, esta variable se define con el valor 1; en caso contrario, será undefined.

Ejemplo

Si el dispositivo puede reproducir datos de sonido MIDI, el ejemplo siguiente define `canMidi` como 1:

```
canMIDI = _capMIDI;

if (canMIDI == 1) {
    // enviar botones de clip de película a fotogramas con botones que
    // activan sonidos de eventos
    tellTarget("buttons") {
        gotoAndPlay(2);
    }
}
```

_capMMS

Disponibilidad

Flash Lite 1.1.

Uso

`_capMMS`

Descripción

Variable numérica; indica si Flash Lite puede enviar mensajes MMS (Multimedia Messaging Service, Servicio de Mensajería Multimedia) mediante el comando `GetURL()` de ActionScript. En caso afirmativo, esta variable se define con el valor 1; en caso contrario, será `undefined`.

Ejemplo

El ejemplo siguiente define `canMMS` como 1 en Flash Lite 1.1, pero deja la variable como no definida (`undefined`) en Flash Lite 1.0 (sin embargo, no todos los teléfonos Flash Lite 1.1 pueden enviar mensajes MMS, por lo que este código depende siempre del teléfono):

```
on(release) {
    canMMS = _capMMS;
    if (canMMS == 1) {
        // enviar un MMS
        myMessage = "mms:4156095555?body=sample mms message";
    } else {
        // enviar un SMS
        myMessage = "sms:4156095555?body=sample sms message";
    }
    getURL(myMessage);
}
```

_capMP3

Disponibilidad

Flash Lite 1.1.

Uso

_capMP3

Descripción

Variable numérica; indica si el dispositivo puede reproducir datos de sonido en formato de audio MPEG (MPEG Audio Layer 3, Audio MPEG Capa 3). En caso afirmativo, esta variable se define con el valor 1; en caso contrario, será undefined.

Ejemplo

Si el dispositivo puede reproducir datos de sonido MP, el ejemplo siguiente define canMP3 como 1:

```
canMP3 = _capMP3;
if (canMP3 == 1) {
    tellTarget("soundClip") {
        gotoAndPlay(2);
    }
}
```

_capSMAF

Disponibilidad

Flash Lite 1.1.

Uso

_capSMAF

Descripción

Variable numérica; indica si el dispositivo puede reproducir archivos multimedia en formato SMAF (Synthetic music Mobile Application Format, Formato de Aplicación Móvil de Música Sintética). En caso afirmativo, esta variable se define con el valor 1; en caso contrario, será undefined.

Ejemplo

El ejemplo siguiente define `canSMAF` como 1 en Flash Lite 1.1, pero deja la variable como no definida (`undefined`) en Flash Lite 1.0 (sin embargo, no todos los teléfonos Flash Lite 1.1 pueden enviar mensajes SMAF, por lo que este código depende siempre del teléfono):

```
canSMAF = _capSMAF;

if (canSMAF) {
    // enviar botones de clip de película a fotogramas con botones que
    // activan sonidos de eventos
    tellTarget("buttons") {
        gotoAndPlay(2);
    }
}
```

_capSMS

Disponibilidad

Flash Lite 1.1.

Uso

`_capSMS`

Descripción

Variable numérica; indica si Flash Lite puede enviar mensajes SMS (*Short Message Service*, Servicio de Mensajes Cortos) mediante el comando `GetURL()` de `ActionScript`. En caso afirmativo, esta variable se define con el valor 1; en caso contrario, será `undefined`.

Ejemplo

El ejemplo siguiente define `canSMS` como 1 en Flash Lite 1.1, pero deja la variable como no definida (`undefined`) en Flash Lite 1.0 (sin embargo, no todos los teléfonos Flash Lite 1.1 pueden enviar mensajes SMS, por lo que este código depende siempre del teléfono):

```
on(release) {
    canSMS = _capSMS;
    if (canSMS) {
        // enviar un SMS
        myMessage = "sms:4156095555?body=sample sms message";
        getURL(myMessage);
    }
}
```

_capStreamSound

Disponibilidad

Flash Lite 1.1.

Uso

_capStreamSound

Descripción

Variable numérica; indica si el dispositivo puede reproducir flujos de sonido (sincronizado). En caso afirmativo, esta variable se define con el valor 1; en caso contrario, será undefined.

Ejemplo

El ejemplo siguiente reproduce flujos de sonido si canStreamSound está activada:

```
on(press) {
    canStreamSound = _capStreamSound;
    if (canStreamSound) {
        // reproducir un flujo de sonido en un clip de película con este botón
        tellTarget("music") {
            gotoAndPlay(2);
        }
    }
}
```

_cap4WayKeyAS

Disponibilidad

Flash Lite 1.1.

Uso

_cap4WayKeyAS

Descripción

Variable numérica; indica si Flash Lite ejecuta expresiones ActionScript vinculadas a controladores de eventos asociados a las teclas de flecha derecha, izquierda, arriba y abajo. Esta variable está definida y tiene un valor de 1 sólo cuando la aplicación host utiliza el modo de navegación en cuatro direcciones para cambiar entre los controles de Flash (botones y campos de introducción de texto). En caso contrario, no está definida (undefined).

Cuando se presiona una de las teclas de cuatro direcciones, si el valor de esta variable es 1, Flash Lite busca primero un controlador para esa tecla. Si no lo encuentra, se produce la navegación en Flash. Sin embargo, si se encuentra un controlador de eventos, no se produce la navegación correspondiente a esa tecla. Por ejemplo, si se encuentra un controlador de tecla presionada para la tecla de flecha abajo, el usuario no puede navegar.

Ejemplo

El ejemplo siguiente define `canUse4Way` como 1 en Flash Lite 1.1, pero deja la variable como no definida (`undefined`) en Flash Lite 1.0 (sin embargo, no todos los teléfonos Flash Lite 1.1 admiten navegación en cuatro direcciones, por lo que este código depende siempre del teléfono):

```
canUse4Way = _cap4WayKeyAS;
if (canUse4Way == 1) {
    msg = "Use your directional joypad to navigate this application";
} else {
    msg = "Please use the 2 key to scroll up, the 6 key to scroll right, the
    8 key to scroll down, and the 4 key to scroll left.";
}
```

\$version

Disponibilidad

Flash Lite 1.1.

Uso

```
$version
```

Descripción

Variable de cadena; contiene el número de versión de Flash Lite. Contiene un número de versión principal, número de versión secundario, número de compilación y número de compilación interna, que normalmente es 0 en todas las versiones publicadas.

El número principal que tienen todos los productos Flash Lite 1.x es 5. El número secundario de Flash Lite 1.0 es 1; el de Flash Lite 1.1 es 2.

Ejemplo

En el reproductor de Flash Lite 1.1, el código siguiente establece el valor de `myVersion` como "5, 2, 12, 0":

```
myVersion = $version;
```

fscommand()

Disponibilidad

Flash Lite 1.1.

Uso

```
status = fscommand("Launch", "application-path, arg1, arg2,..., argn")
```

Parámetros

"Launch" El especificador del comando. El comando Launch es el único que puede utilizar la función `fscommand()` para ejecutarse.

"*application-path, arg1, arg2, ..., argn*" El nombre de la aplicación que se va a iniciar y los parámetros correspondientes, separados por comas.

Descripción

Función; permite al archivo SWF comunicarse con Flash Lite o con el entorno host, como el sistema operativo del teléfono o el dispositivo.

Véase también

[fscommand2\(\)](#)

Launch

Disponibilidad

Flash Lite 1.1.

Uso

```
status = fscommand("Launch", "application-path, arg1, arg2,..., argn")
```

Parámetros

"Launch" El especificador del comando. En Flash Lite, solamente puede utilizar la función `fscommand()` para ejecutar el comando Launch.

"*application-path, arg1, arg2, ..., argn*" El nombre de la aplicación que se va a iniciar y los parámetros correspondientes, separados por comas.

Descripción

Comando ejecutado mediante la función `fscommand()`; inicia otra aplicación en el dispositivo. El nombre de la aplicación que se va a iniciar y los parámetros correspondientes se pasan como un solo argumento.

NOTA

Esta función depende del sistema operativo.

Este comando sólo puede utilizarse cuando el reproductor de Flash Lite se ejecuta en modo autónomo. No se admite cuando el reproductor se inicia en el contexto de otra aplicación (por ejemplo, como un complemento de un navegador).

Ejemplo

El ejemplo siguiente abriría `wap.yahoo.com` en el navegador Web/de servicios en teléfonos Series 60:

```
on(keyPress "9") {
    status = fscommand("launch",
        "z:\\system\\apps\\browser\\browser.app,http://wap.yahoo.com");
}
```

Véase también

[fscommand2\(\)](#)

fscommand2()

Disponibilidad

Flash Lite 1.1.

Uso

```
returnValue = fscommand2(command [, expression1 ... expressionN])
```

Parámetros

command Una cadena que se pasa a la aplicación host para cualquier uso o un comando que se pasa a Flash Lite.

parameter1...*parameterN* Una lista de cadenas separadas por comas que se pasan como parámetros al comando especificado en *command*.

Descripción

Función; permite al archivo SWF comunicarse con Flash Lite o con el entorno host, como el sistema operativo del teléfono o el dispositivo. El valor que devuelve `fscommand2()` depende del comando en cuestión.

La función `fscommand2()` es similar a `fscommand()`, con las siguientes diferencias:

- La función `fscommand2()` admite un número arbitrario de argumentos.
- Flash Lite ejecuta `fscommand2()` inmediatamente, mientras que `fscommand()` se ejecuta al final del fotograma que se está procesando.
- La función `fscommand2()` devuelve un valor que puede utilizarse para informar de la ejecución correcta, con error o el resultado del comando.

Las cadenas y expresiones que se pasan a la función como comandos y parámetros se describen en las tablas de esta sección.

Las tablas tienen las tres columnas siguientes:

- La columna Comando muestra el parámetro literal de cadena que identifica el comando.
- La columna Parámetros explica las clases de valores que se pasan para otros parámetros, en caso necesario.
- La columna Valor devuelto describe los valores devueltos previstos.

Ejemplo

Se proporcionan ejemplos con los comandos que ejecuta mediante la función `fscommand2()`, que se describe a continuación en esta sección.

Véase también

[fscommand\(\)](#)

Escape

Disponibilidad

Flash Lite 1.1.

Descripción

Codifica una cadena arbitraria en un formato seguro para la transferencia en red. Sustituye un carácter no alfanumérico por una secuencia de escape hexadecimal (%XX o %XX%XX en el caso de caracteres multibyte).

Comando	Parámetros	Valor devuelto
"Escape"	<i>original</i> Cadena que va a codificarse en un formato seguro para direcciones URL. <i>encoded</i> Cadena codificada resultante. Estos parámetros son nombres de variables o constantes de tipo cadena (por ejemplo, "Encoded_String").	0: Error. 1: Ejecución correcta.

Ejemplo

El ejemplo siguiente muestra la conversión de una cadena de muestra a su formato codificado:

```
original_string = "Hello, how are you?";  
status = fscommand2("escape", original_string, "encoded_string");  
trace(encoded_string); // salida: Hello%2C%20how%20are%20you%3F
```

Véase también

[Unescape](#)

FullScreen

Disponibilidad

Flash Lite 1.1.

Descripción

Establece el tamaño del área de visualización que se utilizará en la representación. El tamaño puede ser pantalla completa o inferior a pantalla completa.

Este comando sólo puede utilizarse cuando el reproductor de Flash Lite se ejecuta en modo autónomo. No se admite cuando el reproductor se inicia en el contexto de otra aplicación (por ejemplo, como un complemento de un navegador).

Comando	Parámetros	Valor devuelto
"FullScreen"	<i>size</i> Una variable definida o un valor de constante de cadena, con uno de estos valores: <code>true</code> (pantalla completa) o <code>false</code> (inferior a pantalla completa). Cualquier otro valor se trata como <code>false</code> .	-1: No admitido. 0: Admitido.

Ejemplo

El ejemplo siguiente intenta definir el área de visualización como pantalla completa. Si el valor devuelto es distinto de 0, envía la cabeza lectora al fotograma `smallScreenMode`:

```
status = fscommand2("FullScreen", true);
if(status != 0) {
    gotoAndPlay("smallScreenMode");
}
```

GetBatteryLevel

Disponibilidad

Flash Lite 1.1.

Descripción

Devuelve el nivel de batería actual. Es un valor numérico de 0 al valor máximo que devuelve la variable `GetMaxBatteryLevel`.

Comando	Parámetros	Valor devuelto
"GetBatteryLevel"	Ninguno.	-1: No admitido. Otros valores numéricos: el nivel de batería actual.

Ejemplo

El ejemplo siguiente define la variable `battLevel` como el nivel actual de la batería:

```
battLevel = fscommand2("GetBatteryLevel");
```

Véase también

[GetMaxBatteryLevel](#)

GetDateDay

Disponibilidad

Flash Lite 1.1.

Descripción

Devuelve el día de la fecha actual. Es un valor numérico (sin 0 inicial). Los días válidos son del 1 al 31.

Comando	Parámetros	Valor devuelto
"GetDateDay"	Ninguno.	-1: No admitido. De 1 a 31: El día del mes.

Ejemplo

El ejemplo siguiente obtiene la información de fecha y construye una cadena de fecha completa:

```
today = fscommand2("GetDateDay");
weekday = fscommand2("GetDateWeekday");
thisMonth = fscommand2("GetDateMonth");
thisYear = fscommand2("GetDateYear");
when = weekday add ", " add ThisMonth add " " add today add ", " add
    thisYear;
```

Véase también

[GetDateMonth](#), [GetDateWeekday](#), [GetDateYear](#)

GetDateMonth

Disponibilidad

Flash Lite 1.1.

Descripción

Devuelve el mes de la fecha actual como un valor numérico (sin 0 inicial).

Comando	Parámetros	Valor devuelto
"GetDateMonth"	Ninguno.	-1: No admitido. De 1 a 12: El número del mes actual.

Ejemplo

El ejemplo siguiente obtiene la información de fecha y construye una cadena de fecha completa:

```
today = fscommand2("GetDateDay");
weekday = fscommand2("GetDateWeekday");
thisMonth = fscommand2("GetDateMonth");
thisYear = fscommand2("GetDateYear");
when = weekday add ", " add thisMonth add " " add today add ", " add
    thisYear;
```

Véase también

[GetDateDay](#), [GetDateWeekday](#), [GetDateYear](#)

GetDateWeekday

Disponibilidad

Flash Lite 1.1.

Descripción

Devuelve un valor numérico que corresponde al nombre del día de la fecha actual.

Comando	Parámetros	Valor devuelto
"GetDateWeekday"	Ninguno.	-1: No admitido. 0: Domingo. 1: Lunes. 2: Martes. 3: Miércoles. 4: Jueves. 5: Viernes. 6: Sábado.

Ejemplo

El ejemplo siguiente obtiene la información de fecha y construye una cadena de fecha completa:

```
today = fscommand2("GetDateDay");
weekday = fscommand2("GetDateWeekday");
thisMonth = fscommand2("GetDateMonth");
thisYear = fscommand2("GetDateYear");
when = weekday add ", " add thisMonth add " " add today add ", " add
    thisYear;
```

Véase también

[GetDateDay](#), [GetDateMonth](#), [GetDateYear](#)

GetDateYear

Devuelve un valor número de cuatro dígitos correspondiente al año de la fecha actual.

Comando	Parámetros	Valor devuelto
"GetDateYear"	Ninguno.	-1: No admitido. Del 0 al 9999: El año en curso.

Disponibilidad

Flash Lite 1.1.

Ejemplo

El ejemplo siguiente obtiene la información de fecha y construye una cadena de fecha completa:

```
today = fscommand2("GetDateDay");
weekday = fscommand2("GetDateWeekday");
thisMonth = fscommand2("GetDateMonth");
thisYear = fscommand2("GetDateYear");
when = weekday add ", " add thisMonth add " " add today add ", " add
    thisYear;
```

Véase también

[GetDateDay](#), [GetDateMonth](#), [GetDateWeekday](#)

GetDevice

Define un parámetro que identifica el dispositivo en el que se está ejecutando Flash Lite. Este identificador suele ser el nombre del modelo.

Comando	Parámetros	Valor devuelto
"GetDevice"	<i>device</i> Cadena que va a recibir el identificador del dispositivo. Puede ser el nombre de una variable o una cadena que contiene el nombre de una variable.	-1: No admitido. 0: Admitido.

Disponibilidad

Flash Lite 1.1.

Ejemplo

El ejemplo de código siguiente asigna el identificador de dispositivo a la variable `statusdevice` y actualiza un campo de texto con el nombre de dispositivo genérico.

A continuación se muestran resultados de ejemplo y los dispositivos que representan:

D506i Teléfono Mitsubishi 506i.

DFOMA1 Teléfono Mitsubishi FOMA1.

F506i Teléfono Fujitsu 506i.

FFOMA1 Teléfono Fujitsu FOMA1.

N506i Teléfono NEC 506i.

NFOMA1 Teléfono NEC FOMA1.

Nokia3650 Teléfono Nokia 3650.

p506i Teléfono Panasonic 506i.

PFOMA1 Teléfono Panasonic FOMA1.

SH506i Teléfono Sharp 506i.

SHFOMA1 Teléfono Sharp FOMA1.

S0506i Teléfono Sony 506iphone.

```
statusdevice = fscommand2("GetDevice", "devicename");
switch(devicename) {
    case "D506i":
        /:myText += "device: Mitsubishi 506i" add newline;
        break;
    case "DFOMA1":
        /:myText += "device: Mitsubishi FOMA1" add newline;
        break;
    case "F506i":
        /:myText += "device: Fujitsu 506i" add newline;
        break;
    case "FFOMA1":
        /:myText += "device: Fujitsu FOMA1" add newline;
        break;
    case "N506i":
        /:myText += "device: NEC 506i" add newline;
        break;
    case "NFOMA1":
        /:myText += "device: NEC FOMA1" add newline;
        break;
    case "Nokia 3650":
        /:myText += "device: Nokia 3650" add newline;
        break;
    case "P506i":
        /:myText += "device: Panasonic 506i" add newline;
        break;
    case "PFOMA1":
        /:myText += "device: Panasonic FOMA1" add newline;
        break;
```

```

case "SH506i":
    /:myText += "device: Sharp 506i" add newline;
    break;
case "SHF0MA1":
    /:myText += "device: Sharp F0MA1" add newline;
    break;
case "S0506i":
    /:myText += "device: Sony 506i" add newline;
    break;
}

```

GetDeviceID

Define un parámetro que representa el identificador exclusivo del dispositivo (por ejemplo, el número de serie).

Comando	Parámetros	Valor devuelto
"GetDeviceID"	<i>id</i> Una cadena para recibir el identificador exclusivo del dispositivo. Puede ser el nombre de una variable o una cadena que contenga el nombre de una variable.	-1: No admitido. 0: Admitido.

Disponibilidad

Flash Lite 1.1.

Ejemplo

El ejemplo siguiente asigna el identificador exclusivo a la variable `deviceID`:

```
status = fscommand2("GetDeviceID", "deviceID");
```

GetFreePlayerMemory

Devuelve la cantidad de memoria de pila disponible para Flash Lite, expresada en kilobytes.

Comando	Parámetros	Valor devuelto
"GetFreePlayerMemory"	Ninguno.	-1: No admitido. 0 o un valor positivo: Kilobytes disponibles de memoria Heap.

Disponibilidad

Flash Lite 1.1.

Ejemplo

El ejemplo siguiente define el estado igual a la cantidad de memoria libre:

```
status = fscommand2("GetFreePlayerMemory");
```

Véase también

[GetTotalPlayerMemory](#)

GetLanguage

Disponibilidad

Flash Lite 1.1.

Establece un parámetro que identifica el idioma que se utiliza en el dispositivo. El resultado del idioma es una cadena en una variable que se pasa por nombre.

Comando	Parámetros	Valor devuelto
"GetLanguage"	<p><i>language</i> Cadena para recibir el código de idioma. Puede ser el nombre de una variable o una cadena que contenga el nombre de una variable. El valor es uno de los siguientes:</p> <ul style="list-style-type: none">cs: Checo.da: Danés.de: Alemán.en-UK: Inglés del Reino Unido o internacional.en-US: Inglés de Estados Unidos.es: Español.fi: Finlandés.fr: Francés.hu: Húngaro.it: Italiano.ja: Japonés.ko: Coreano.nl: Holandés.no: Noruego.pl: Polaco.pt: Portugués.ru: Ruso.sv: Sueco.tr: Turco.xu: un idioma no determinado.zh-CN: Chino simplificado.zh-TW: Chino tradicional.	<p>-1: No admitido. 0: Admitido.</p>

NOTA

Cuando se selecciona que los teléfonos japoneses muestren el texto en inglés, se devuelve `en_US` como *language*.

Ejemplo

El ejemplo siguiente asigna el código de idioma a la variable `language` y actualiza un campo de texto con el idioma que reconoce el reproductor de Flash Lite:

```
statuslanguage = fscommand2("GetLanguage", "language");
switch(language) {
  case "cs":
    /:myText += "language is Czech" add newline;
    break;
  case "da":
    /:myText += "language is Danish" add newline;
    break;
  case "de":
    /:myText += "language is German" add newline;
    break;
  case "en-UK":
    /:myText += "language is UK" add newline;
    break;
  case "en-US":
    /:myText += "language is US" add newline;
    break;
  case "es":
    /:myText += "language is Spanish" add newline;
    break;
  case "fi":
    /:myText += "language is Finnish" add newline;
    break;
  case "fr":
    /:myText += "language is French" add newline;
    break;
  case "hu":
    /:myText += "language is Hungarian" add newline;
    break;
  case "it":
    /:myText += "language is Italian" add newline;
    break;
  case "jp":
    /:myText += "language is Japanese" add newline;
    break;
  case "ko":
    /:myText += "language is Korean" add newline;
    break;
  case "nl":
    /:myText += "language is Dutch" add newline;
    break;
  case "no":
    /:myText += "language is Norwegian" add newline;
    break;
}
```

```

case "pl":
    /:myText += "language is Polish" add newline;
    break;
case "pt":
    /:myText += "language is Portuguese" add newline;
    break;
case "ru":
    /:myText += "language is Russian" add newline;
    break;
case "sv":
    /:myText += "language is Swedish" add newline;
    break;
case "tr":
    /:myText += "language is Turkish" add newline;
    break;
case "xu":
    /:myText += "language is indeterminable" add newline;
    break;
case "zh-CN":
    /:myText += "language is simplified Chinese" add newline;
    break;
case "zh-TW":
    /:myText += "language is traditional Chinese" add newline;
    break;
}

```

GetLocaleLongDate

Disponibilidad

Flash Lite 1.1.

Descripción

Define un parámetro para una cadena que representa la fecha actual, en formato largo, según la configuración regional seleccionada.

Comando	Parámetros	Valor devuelto
"GetLocaleLongDate"	<p><i>longdate</i> Variable de cadena para recibir el formato largo del valor de la fecha actual, como "October 16, 2004" o "16 October 2004". Puede ser el nombre de una variable o una cadena que contiene el nombre de una variable. El valor que devuelve <i>longdate</i> es una cadena de varios caracteres y longitud variable. El formato real depende del dispositivo y de la configuración local.</p>	<p>-1: No admitido. 0: Admitido.</p>

Ejemplo

El ejemplo siguiente intenta devolver el formato largo de la fecha actual en la variable `longDate`. Además, define el valor de `status` para que indique si ha sido posible.

```
status = fscommand2("GetLocaleLongDate", "longdate");  
trace (longdate); // salida: Martes, 14 de junio, 2005
```

Véase también

[GetLocaleShortDate](#), [GetLocaleTime](#)

GetLocaleShortDate

Disponibilidad

Flash Lite 1.1.

Descripción

Define un parámetro para una cadena que representa la fecha actual, en formato corto, según la configuración regional seleccionada.

Comando	Parámetros	Valor devuelto
"GetLocaleShortDate"	<p><i>shortdate</i> Variable de cadena para recibir el formato corto del valor de la fecha actual, como "10/16/2004" o "16-10-2004".</p> <p>Puede ser el nombre de una variable o una cadena que contenga el nombre de una variable.</p> <p>El valor que devuelve <i>shortdate</i> es una cadena de varios caracteres y longitud variable. El formato real depende del dispositivo y de la configuración regional.</p>	<p>-1: No admitido. 0: Admitido.</p>

Ejemplo

El ejemplo siguiente intenta obtener el formato corto de la fecha actual en la variable `shortDate`. Además, define el valor de `status` para que indique si ha sido posible.

```
status = fscommand2("GetLocaleShortDate", "shortdate");  
trace (shortdate); // salida: 06/14/05
```

Véase también

[GetLocaleLongDate](#), [GetLocaleTime](#)

GetLocaleTime

Disponibilidad

Flash Lite 1.1.

Descripción

Define un parámetro para una cadena que representa la hora actual, según la configuración regional seleccionada.

Comando	Parámetros	Valor devuelto
"GetLocaleTime"	<i>time</i> Variable de cadena para recibir el valor de la hora actual, como "6:10:44 PM" o "18:10:44". Puede ser el nombre de una variable o una cadena que contenga el nombre de una variable. El valor que devuelve <i>time</i> es una cadena de varios caracteres y longitud variable. El formato real depende del dispositivo y de la configuración regional.	-1: No admitido. 0: Admitido.

Ejemplo

El ejemplo siguiente intenta obtener la hora local en la variable *time*. Además, define el valor de *status* para que indique si ha sido posible.

```
status = fscommand2("GetLocaleTime", "time");
trace(time); // salida: 14:30:21
```

Véase también

[GetLocaleLongDate](#), [GetLocaleShortDate](#)

GetMaxBatteryLevel

Disponibilidad

Flash Lite 1.1.

Descripción

Devuelve el nivel máximo de batería del dispositivo. Es un valor numérico mayor que 0.

Comando	Parámetros	Valor devuelto
"GetMaxBatteryLevel"	Ninguno.	-1: No admitido. Otros valores: el nivel de batería máximo.

Ejemplo

El ejemplo siguiente define la variable `maxBatt` como el nivel de batería máximo:

```
maxBatt = fscommand2("GetMaxBatteryLevel");
```

GetMaxSignalLevel

Disponibilidad

Flash Lite 1.1.

Descripción

Devuelve el nivel máximo de intensidad de la señal. Es un valor numérico mayor que 0.

Comando	Parámetros	Valor devuelto
"GetMaxSignalLevel"	Ninguno.	-1: No admitido. Otros valores numéricos: nivel máximo de la señal.

Ejemplo

El ejemplo siguiente asigna la máxima intensidad de señal a la variable `sigStrengthMax`:

```
sigStrengthMax = fscommand2("GetMaxSignalLevel");
```

GetMaxVolumeLevel

Disponibilidad

Flash Lite 1.1.

Descripción

Devuelve el nivel máximo de volumen del dispositivo como un valor numérico.

Comando	Parámetros	Valor devuelto
"GetMaxVolumeLevel"	Ninguno.	-1: No admitido. Otros valores: el nivel de volumen máximo.

Ejemplo

El ejemplo siguiente define la variable `maxvolume` como el nivel máximo de volumen del dispositivo:

```
maxvolume = fscommand2("GetMaxVolumeLevel");  
trace (maxvolume); // salida: 80
```

Véase también

[GetVolumeLevel](#)

GetNetworkConnectStatus

Disponibilidad

Flash Lite 1.1.

Descripción

Devuelve un valor que indica el estado actual de la conexión de red.

Comando	Parámetros	Valor devuelto
"GetNetworkConnectStatus"	Ninguno.	-1: No admitido. 0: Hay una conexión de red activa en este momento. 1: El dispositivo intenta conectarse a la red. 2: No hay ninguna conexión de red activa en este momento. 3: Se ha interrumpido la conexión a la red. 4: No se puede determinar la conexión de red.

Ejemplo

El ejemplo siguiente asigna el estado de conexión de red a la variable `connectstatus` y, a continuación, utiliza una sentencia `switch` para actualizar el estado de conexión en un campo de texto:

```
connectstatus = fscommand2("GetNetworkConnectStatus");  
switch (connectstatus) {  
    case -1 :  
        /:myText += "connectstatus not supported" add newline;  
        break;  
    case 0 :  
        /:myText += "connectstatus shows active connection" add newline;  
        break;  
    case 1 :  
        /:myText += "connectstatus shows attempting connection" add newline;  
        break;
```

```

case 2 :
    /:myText += "connectstatus shows no connection" add newline;
    break;
case 3 :
    /:myText += "connectstatus shows suspended connection" add newline;
    break;
case 4 :
    /:myText += "connectstatus shows indeterminable state" add newline;
    break;
}

```

GetNetworkName

Disponibilidad

Flash Lite 1.1.

Descripción

Establece un parámetro para el nombre de la red actual.

Comando	Parámetros	Valor devuelto
"GetNetworkName"	<i>networkName</i> Cadena que representa el nombre de la red. Puede ser el nombre de una variable o una cadena que contenga el nombre de una variable. Si la red está registrada y se puede determinar su nombre, <i>networkname</i> será el nombre de la red; en caso contrario, será una cadena vacía.	-1: No admitido. 0: No hay ninguna red registrada. 1: Red registrada, pero se desconoce su nombre. 2: Red registrada y se conoce su nombre.

Ejemplo

El ejemplo siguiente asigna el nombre de la red actual a la variable `myNetName` y un valor de estado a la variable `netNameStatus`:

```
netNameStatus = fscommand2("GetNetworkName", myNetName);
```

GetNetworkRequestStatus

Disponibilidad

Flash Lite 1.1.

Descripción

Devuelve un valor que indica el estado de la solicitud HTTP más reciente.

Comando	Parámetros	Valor devuelto
"GetNetworkRequestStatus"	Ninguno.	-1: No se admite el comando. 0: Hay una solicitud pendiente, se ha establecido una conexión de red, se ha resuelto el nombre de host del servidor y se ha realizado la conexión con el servidor. 1: Hay una solicitud pendiente y se ha establecido una conexión de red. 2: Hay una solicitud pendiente, pero todavía no se ha establecido una conexión de red. 3: Hay una solicitud pendiente, se ha establecido una conexión de red y se está resolviendo el nombre de host del servidor. 4: La solicitud ha fallado debido a un error de la red. 5: La solicitud ha fallado debido a un error de conexión al servidor. 6: El servidor ha devuelto un error de HTTP (por ejemplo, 404). 7: La solicitud ha fallado debido a un error al acceder al servidor DNS o al resolver el nombre del servidor. 8: La solicitud se ha realizado correctamente. 9: La solicitud ha fallado debido a un error de tiempo límite agotado. 10: La solicitud aún no se ha realizado.

Ejemplo

El ejemplo siguiente asigna el estado de la última solicitud HTTP a la variable `requeststatus` y, a continuación, utiliza una sentencia `switch` para actualizar el estado en un campo de texto:

```
requeststatus = fscommand2("GetNetworkRequestStatus");
switch (requeststatus) {
    case -1:
        /:myText += "requeststatus not supported" add newline;
        break;
    case 0:
        /:myText += "connection to server has been made" add newline;
        break;
    case 1:
        /:myText += "connection is being established" add newline;
        break;
    case 2:
        /:myText += "pending request, contacting network" add newline;
        break;
    case 3:
        /:myText += "pending request, resolving domain" add newline;
        break;
    case 4:
        /:myText += "failed, network error" add newline;
        break;
    case 5:
        /:myText += "failed, couldn't reach server" add newline;
        break;
    case 6:
        /:myText += "HTTP error" add newline;
        break;
    case 7:
        /:myText += "DNS failure" add newline;
        break;
    case 8:
        /:myText += "request has been fulfilled" add newline;
        break;
    case 9:
        /:myText += "request timedout" add newline;
        break;
    case 10:
        /:myText += "no HTTP request has been made" add newline;
        break;
}
```

GetNetworkStatus

Disponibilidad

Flash Lite 1.1.

Descripción

Devuelve un valor que indica el estado de la red telefónica (es decir, si hay una red registrada y si el teléfono está lejos de la red doméstica).

Comando	Parámetros	Valor devuelto
"GetNetworkStatus"	Ninguno.	-1: No se admite el comando. 0: No hay ninguna red registrada. 1: Red doméstica. 2: Red doméstica ampliada. 3: Móvil (lejos de la red doméstica).

Ejemplo

El ejemplo siguiente asigna el estado de conexión de red a la variable `networkstatus` y, a continuación, utiliza una sentencia `switch` para actualizar el estado en un campo de texto:

```
networkstatus = fscommand2("GetNetworkStatus");
switch(networkstatus) {
    case -1:
        /:myText += "network status not supported" add newline;
        break;
    case 0:
        /:myText += "no network registered" add newline;
        break;
    case 1:
        /:myText += "on home network" add newline;
        break;
    case 2:
        /:myText += "on extended home network" add newline;
        break;
    case 3:
        /:myText += "roaming" add newline;
        break;
}
```

GetPlatform

Disponibilidad

Flash Lite 1.1.

Descripción

Define un parámetro que identifica la plataforma actual, que describe ampliamente la clase de dispositivo. Para los dispositivos con sistemas operativos abiertos, este identificador es normalmente el nombre y la versión del sistema operativo.

Comando	Parámetros	Valor devuelto
"GetPlatform"	<i>platform</i> Cadena que va a recibir el identificador de la plataforma. Puede ser el nombre de una variable o una cadena que contenga el nombre de una variable.	-1: No admitido. 0: Admitido.

Ejemplo

El ejemplo de código siguiente asigna el identificador de plataforma a la variable `statusplatform` y actualiza un campo de texto con el nombre de plataforma genérico.

Los ejemplos siguientes son resultados de muestra de `myPlatform` y las clases de dispositivos que representan:

506i Teléfono 506i.

FOMA1 Teléfono FOMA1.

Symbian6.1_s60.1 Teléfono Symbian 6.1, Series 60 versión 1.

Symbian7.0 Teléfono Symbian 7.0.

```
statusplatform = fscommand2("GetPlatform", "platform");
switch(platform){
    case "506i":
        /:myText += "platform: 506i" add newline;
        break;
    case "FOMA1":
        /:myText += "platform: FOMA1" add newline;
        break;
    case "Symbian6.1-Series60v1":
        /:myText += "platform: Symbian6.1, Series 60 version 1 phone" add
        newline;
        break;
    case "Symbian7.0":
        /:myText += "platform: Symbian 7.0" add newline;
        break;
}
```

GetPowerSource

Disponibilidad

Flash Lite 1.1.

Descripción

Devuelve un valor que indica si la fuente de alimentación es una batería o externa.

Comando	Parámetros	Valor devuelto
"GetPowerSource"	Ninguno.	-1: No admitido. 0: El dispositivo funciona con una batería. 1: El dispositivo funciona con una fuente de alimentación externa.

Ejemplo

El ejemplo siguiente define la variable `myPower` para que indique la fuente de alimentación o -1 si no puede hacerlo.

```
myPower = fscommand2("GetPowerSource");
```

GetSignalLevel

Disponibilidad

Flash Lite 1.1.

Descripción

Devuelve la intensidad de la señal actual como un valor numérico.

Comando	Parámetros	Valor devuelto
"GetSignalLevel"	Ninguno.	-1: No admitido. Otros valores numéricos: El nivel de señal actual, desde 0 al valor máximo que devuelve <code>GetMaxSignalLevel</code> .

Ejemplo

El ejemplo siguiente asigna el valor de nivel de señal a la variable `sigLevel`:

```
sigLevel = fscommand2("GetSignalLevel");
```

GetTimeHours

Disponibilidad

Flash Lite 1.1.

Descripción

Devuelve el valor de hora de la hora actual del día, basada en un reloj de 24 horas. Es un valor numérico (sin 0 inicial).

Comando	Parámetros	Valor devuelto
"GetTimeHours"	Ninguno.	-1: No admitido. De 0 a 23: la hora actual.

Ejemplo

El ejemplo siguiente define la variable `hour` como la parte correspondiente a la hora del día o -1:

```
hour = fscommand2("GetTimeHours");  
trace(hour);           // salida: 14
```

Véase también

[GetTimeMinutes](#), [GetTimeSeconds](#), [GetTimeZoneOffset](#)

GetTimeMinutes

Disponibilidad

Flash Lite 1.1.

Descripción

Devuelve el minuto de la hora actual del día. Es un valor numérico (sin 0 inicial).

Comando	Parámetros	Valor devuelto
"GetTimeMinutes"	Ninguno.	-1: No admitido. De 0 a 59: el minuto actual.

Ejemplo

El ejemplo siguiente define la variable `minutes` como la parte correspondiente a los minutos del día o -1:

```
minutes = fscommand2("GetTimeMinutes");  
trace (minutes);           // salida: 38
```

Véase también

[GetTimeHours](#), [GetTimeSeconds](#), [GetTimeZoneOffset](#)

GetTimeSeconds

Disponibilidad

Flash Lite 1.1.

Descripción

Devuelve el segundo de la hora actual del día. Es un valor numérico (sin 0 inicial).

Comando	Parámetros	Valor devuelto
"GetTimeSeconds"	Ninguno.	-1: No admitido. De 0 a 59: el segundo actual.

Ejemplo

El ejemplo siguiente define la variable `seconds` como la parte correspondiente a los segundos del día o -1:

```
seconds = fscommand2("GetTimeSeconds");  
trace (seconds);           // salida: 41
```

Véase también

[GetTimeHours](#), [GetTimeMinutes](#), [GetTimeZoneOffset](#)

GetTimeZoneOffset

Disponibilidad

Flash Lite 1.1.

Descripción

Define un parámetro como el número de minutos de diferencia entre la zona horaria local y la hora universal (UTC).

Comando	Parámetros	Valor devuelto
"GetTimeZoneOffset"	<i>timezoneOffset</i> Número de minutos entre la zona horaria local y la zona universal (UTC). Puede ser el nombre de una variable o una cadena que contenga el nombre de una variable. El resultado es un valor numérico positivo o negativo, como el siguiente: 540: hora estándar en Japón -420: Horario de verano del Pacífico	-1: No admitido. 0: Admitido.

Ejemplo

El ejemplo siguiente asigna los minutos de diferencia con la zona universal a la variable `timezoneoffset` y define `status` como 0 o bien define `status` como -1:

```
status = fscommand2("GetTimeZoneOffset", "timezoneoffset");  
trace (timezoneoffset);// salida: 300
```

Véase también

[GetTimeHours](#), [GetTimeMinutes](#), [GetTimeSeconds](#)

GetTotalPlayerMemory

Disponibilidad

Flash Lite 1.1.

Descripción

Devuelve la cantidad de memoria de pila total asignada a Flash Lite, expresada en kilobytes.

Comando	Parámetros	Valor devuelto
"GetTotalPlayerMemory"	Ninguno.	-1: No admitido. 0 o valor positivo: total de kilobytes de memoria Heap.

Ejemplo

El ejemplo siguiente define la variable `status` como la cantidad total de memoria de pila:

```
status = fscommand2("GetTotalPlayerMemory");
```

Véase también

[GetFreePlayerMemory](#)

GetVolumeLevel

Disponibilidad

Flash Lite 1.1.

Descripción

Devuelve el nivel actual del volumen del dispositivo como un valor numérico.

Comando	Parámetros	Valor devuelto
"GetVolumeLevel"	Ninguno.	-1: No admitido. Otros valores numéricos: el volumen actual, desde 0 al valor que devuelve <code>fscommand2("GetMaxVolumeLevel")</code> .

Ejemplo

El ejemplo siguiente asigna el nivel de volumen actual a la variable `volume`:

```
volume = fscommand2("GetVolumeLevel");  
trace (volume);           // salida: 50
```

Véase también

[GetVolumeLevel](#)

Quit

Disponibilidad

Flash Lite 1.1.

Descripción

Hace que el reproductor de Flash Lite detenga la reproducción y se cierre.

Este comando sólo puede utilizarse cuando el reproductor de Flash Lite se ejecuta en modo autónomo. No se admite cuando el reproductor se inicia en el contexto de otra aplicación (por ejemplo, como un complemento de un navegador).

Comando	Parámetros	Valor devuelto
"Quit"	Ninguno.	-1: No admitido.

Ejemplo

El ejemplo siguiente hace que Flash Lite detenga la reproducción y se cierre cuando se ejecute en modo independiente:

```
status = fscommand2("Quit");
```

ResetSoftKeys

Disponibilidad

Flash Lite 1.1.

Descripción

Restablece la configuración original de las teclas programables.

Este comando sólo puede utilizarse cuando el reproductor de Flash Lite se ejecuta en modo autónomo. No se admite cuando el reproductor se inicia en el contexto de otra aplicación (por ejemplo, como un complemento de un navegador).

Comando	Parámetros	Valor devuelto
"ResetSoftKeys"	Ninguno.	-1: No admitido. 0: Admitido.

Ejemplo

La siguiente sentencia restablece la configuración original de las teclas programables:

```
status = fscommand2("ResetSoftKeys");
```

Véase también

[SetSoftKeys](#)

SetInputTextType

Disponibilidad

Flash Lite 1.1.

Descripción

Especifica el modo en que debe abrirse el campo de introducción de texto:

Comando	Parámetros	Valor devuelto
"SetInputTextType"	<p><i>variableName</i> Nombre del campo de introducción de texto. Puede ser el nombre de una variable o una cadena que contenga el nombre de una variable.</p> <p><i>type</i> Uno de los valores Numeric, Alpha, Alphanumeric, Latin, NonLatin o NoRestriction.</p>	0: Error. 1: Ejecución correcta.

Flash Lite admite la funcionalidad de introducción de texto; para ello, pide a la aplicación host que inicie la interfaz de introducción de texto del dispositivo, que suele denominarse *procesador cliente* (FEP). Cuando no se utiliza el comando `SetInputTextType`, el procesador cliente se abre en modo predeterminado.

La tabla siguiente muestra el efecto de cada modo y los modos que se sustituyen:

Modo especificado	Selecciona en el FEP uno de estos modos excluyentes:	Si no se admite en el dispositivo actual, abre el FEP en este modo.
Numeric	Sólo números (del 0 al 9)	Alphanumeric
Alpha	Sólo caracteres alfabéticos (de la A a la Z, de la a a la z)	Alphanumeric
Alphanumeric	Sólo caracteres alfanuméricos (del 0 al 9, A-Z o a-z)	Latin
Latin	Sólo caracteres latinos (alfanuméricos y puntuación)	NoRestriction

Modo especificado	Selecciona en el FEP uno de estos modos excluyentes:	Si no se admite en el dispositivo actual, abre el FEP en este modo.
NonLatin	Sólo caracteres no latinos (por ejemplo Kanji y Kana)	NoRestriction
NoRestriction	Modo predeterminado (sin limitación en el FEP)	

NOTA

No todos los teléfonos móviles admiten estos tipos de campos de introducción de texto. Por este motivo, debe validar los datos introducidos.

Ejemplo

La siguiente línea de código establece el tipo de texto del campo asociado con la variable `input1` de modo que se reciban datos numéricos:

```
status = fscommand2("SetInputTextType", "input1", "Numeric");
```

SetQuality

Disponibilidad

Flash Lite 1.1.

Descripción

Establece la calidad de la presentación de la animación.

Comando	Parámetros	Valor devuelto
"SetQuality"	<i>quality</i> La calidad de representación, debe ser "high", "medium" o "low".	-1: No admitido. 0: Admitido.

Ejemplo

El ejemplo siguiente establece la calidad de representación como low:

```
status = fscommand2("SetQuality", "low");
```

SetSoftKeys

Disponibilidad

Flash Lite 1.1.

Descripción

Cambia la asignación de las teclas programables izquierda y derecha del dispositivo, siempre que se pueda acceder a ellas y sea posible.

Tras ejecutar este comando, al presionar la tecla izquierda se genera un evento `PageUp` y al presionar la tecla derecha se genera un evento `PageDown`. El código ActionScript asociado con los eventos `PageUp` y `PageDown` se ejecuta cuando se presiona la tecla correspondiente.

Este comando sólo puede utilizarse cuando el reproductor de Flash Lite se ejecuta en modo autónomo. No se admite cuando el reproductor se inicia en el contexto de otra aplicación (por ejemplo, como un complemento de un navegador).

Comando	Parámetros	Valor devuelto
"SetSoftKeys"	<i>left</i> Texto que va a mostrarse para la tecla programable izquierda. <i>right</i> Texto que va a mostrarse para la tecla programable derecha. Estos parámetros son nombres de variables o constantes de tipo cadena (por ejemplo, "Previous").	-1: No admitido. 0: Admitido.

Ejemplo

El ejemplo siguiente asigna a la tecla programable izquierda el nombre `Previous` y a la derecha, `Next`.

```
status = fscommand2("SetSoftKeys", "Previous", "Next");
```

Véase también

[ResetSoftKeys](#)

StartVibrate

Disponibilidad

Flash Lite 1.1.

Descripción

Inicia la vibración del teléfono. Si hay activa una vibración, Flash Lite la detiene antes de iniciar una nueva. Las vibraciones se detienen también cuando se detiene o interrumpe la aplicación Flash, y cuando se cierra el reproductor de Flash Lite.

Comando	Parámetros	Valor devuelto
"StartVibrate"	<i>time_on</i> Cantidad de tiempo, en milisegundos (hasta un máximo de 5 segundos) que está activa la vibración.	-1: No admitido. 0: La vibración se ha iniciado.
	<i>time_off</i> Cantidad de tiempo, en milisegundos (hasta un máximo de 5 segundos) que está desactivada la vibración.	1: Se ha producido un error y no se ha podido iniciar la vibración.
	<i>repeat</i> Número de veces (hasta un máximo de 3) que se debe repetir esta vibración.	

Ejemplo

El ejemplo siguiente intenta iniciar una secuencia de vibración activada 2,5 segundos, vibración desactivada 1 segundo, repetida dos veces. Asigna un valor a la variable `status` que indica que si se ha realizado correctamente o si se ha producido un error.

```
status = fscommand2("StartVibrate", 2500, 1000, 2);
```

Véase también

[StopVibrate](#)

StopVibrate

Disponibilidad

Flash Lite 1.1.

Descripción

Detiene la vibración del teléfono, si está activa.

Comando	Parámetros	Valor devuelto
"StopVibrate"	Ninguno.	-1: No admitido. 0: vibración detenida.

Ejemplo

El ejemplo siguiente llama a `StopVibrate` y guarda el resultado (no admitido o vibración detenida) en la variable `status`:

```
status = fscommand2("StopVibrate");
```

Véase también

[StartVibrate](#)

Unescape

Disponibilidad

Flash Lite 1.1.

Descripción

Descodifica a su formato normal una cadena arbitraria que se codificó para protegerla durante transferencias en red. Todos los caracteres en formato hexadecimal, es decir, un carácter de porcentaje seguido por dos dígitos hexadecimales, se convierten a su formato descodificado.

Comando	Parámetros	Valor devuelto
"Unescape"	<p><i>original</i> Cadena que se va a descodificar de un formato seguro para direcciones URL a un formato normal.</p> <p><i>decoded</i> Cadena descodificada resultante. (Este parámetro puede ser el nombre de una variable o una cadena que contenga el nombre de una variable.)</p>	<p>0: Error.</p> <p>1: Ejecución correcta.</p>

Ejemplo

El ejemplo siguiente muestra la descodificación de una cadena codificada:

```
encoded_string = "Hello%2C%20how%20are%20you%3F";  
status = fscommand2("unescape", encoded_string, "normal_string");  
trace(normal_string); // salida: Hello, how are you?
```

Véase también

[Escape](#)

Índice alfabético

Símbolos

! (NOT lógico), operador 97
" " (delimitador de cadena), operador 108
\$version, variable 128
% (módulo), operador 99
%= (asignación de módulo), operador 100
&& (AND lógico), operador 96
|| (OR lógico), operador 98
* (multiplicar), operador 101
*= (asignación de multiplicación), operador 100
+ (suma numérica), operador 102
++ (incremento), operador 95
+= (asignación de suma), operador 87
, (coma), operador 90
-= (asignación de resta), operador 114
. (punto), operador 94
/ (Barra diagonal - línea de tiempo raíz), propiedad 55
/ (dividir), operador 93
/* (comentario en bloque), operador 89
// (comentario), operador 91
/= (división), operador 94
< (numérico menor o igual que), operador 106
< (numérico menor que), operador 105
<> (desigualdad numérica), operador 105
= (asignación), operador 88
== (igualdad numérica), operador 103
> (mayor o igual que), operador 104
> (mayor que), operador 103
? (condicional), operador 92
_alpha, variable 56
_cap4WayKeyAS, variable 127
_capCompoundSound, variable 121
_capEmail, variable 121
_capLoadData, variable 122
_capMFi, variable 123
_capMIDI, variable 123
_capMMS, variable 124

_capSMAF, variable 125
_capSMS, variable 126
_capStreamSound, variable 127
_currentframe, propiedad 56
_focusrect, propiedad 57
_framesloaded, propiedad 58
_height, propiedad 58
_highquality, propiedad 59
_level, propiedad 60
_name, propiedad 61
_rotation, propiedad 62
_scroll, propiedad 62
_target, propiedad 63
_visible, propiedad 64
_width, propiedad 65
_x, propiedad 65
_xscale, propiedad 66
_y, propiedad 67
_yscale, propiedad 68
- (resta), operador 113
-- (decremento), operador 92

A

add (concatenación de cadenas), operador 86
_alpha, variable 56
AND lógico, operador 96
AND, operador 96
and, operador 87
asignación de división, operador 94
asignación de módulo 100
asignación de resta, operador 114
asignación de suma (+=), operador 87
asignación, operador 88

B

break, sentencia 70

C

cadena mayor o igual que 110
cadena mayor que, operador 109
cadena menor o igual que 112
call 16
_cap4WayKeyAS, variable 127
_capCompoundSound, variable 121
_capEmail, variable 121
_capLoadData, variable 122
_capMFi, variable 123
_capMMS, variable 124
_capSMAF, variable 125
_capSMS, variable 126
_capStreamSound, variable 127
case, sentencia 71
chr(), función 17
coma, operador 90
comentario en bloque, operador 89
comentarios
 block 89
 en línea 91
concatenación 86
condiciones 77
continue, sentencia 72
_currentframe, propiedad 56

D

delimitador de cadena, operador 108
desigualdad, operador 105
división 93
do..while, sentencia 74
duplicateMovieClip, función 17

E

else if, sentencia 75
else, sentencia 74
eq (igualdad de cadenas), operador 108
eval(), función 18

F

_focusrect, propiedad 57
for, bucle 76
for, sentencia 76
_framesloaded, propiedad 58
fscommand(), comando 129
funciones
 chr() 17
 duplicateMovieClip() 17
 eval() 18
 fscommand() 129
 getProperty() 19
 getTimer() 20
 getURL() 21
 gotoAndPlay() 24
 gotoAndStop() 25
 iffFrameLoaded() 26
 int() 27
 length() 27
 loadMovie() 28
 loadMovieNum() 29
 loadVariables() 31
 loadVariablesNum() 32
 mbchr() 33
 mbsubstring() 35
 nextFrame() 36
 nextScene() 37
 Number() 37
 on() 38
 ord() 39
 play() 40
 prevFrame() 40
 prevScene() 41
 random() 42
 removeMovieClip() 43
 set() 43
 setProperty() 44
 stop() 45
 stopAllSounds() 46
 String() 46
 substring() 47
 tellTarget() 48
 toggleHighQuality() 49
 trace() 49
 unloadMovie() 50
 unloadMovieNum() 51

G

ge (cadena mayor o igual a), operador 110
getProperty(), función 19
getTimer(), función 20
getURL(), función 21
gotoAndPlay(), función 24
gotoAndStop(), función 25
gt (cadena mayor que), operador 109

H

_height, propiedad 58
_highquality, propiedad 59

I

identificador de línea de tiempo raíz 55
if, sentencia 77
iframeLoaded(), función 26
igualdad de cadenas, operador 108
incremento, operador 95
int(), función 27

L

le (cadena menor o igual que), operador 112
length(), función 27
_level, propiedad 60
loadMovie(), función 28
loadMovieNum(), función 29
loadVariables(), función 31
loadVariablesNum(), función 32
lt (cadena menor que), operador 111

M

maxscroll, propiedad 61
mayor o igual que, operador 104
mayor que, operador 103
mbchr(), función 33
mbsubstring(), función 35
menor o igual que, operador 106
menor que, operador 105
mensajes MMS 124
mensajes, variables
 _capMMS 124
 _capSMS 126
módulo, operador 99
multiplicación 101

N

_name, propiedad 61
ne (cadena no igual), operador 111
nextFrame(), función 36
nextScene(), función 37
NOT lógico, operador 97
NOT, operador 97
Number(), función 37

O

on(), función 38
operador condicional 92
operadores
 and 87
 AND lógico 96
 asignación 88
 asignación de división 94
 asignación de módulo 100
 asignación de resta 114
 asignación de suma (+) 87
 cadena mayor o igual que 110
 cadena mayor que 109
 cadena menor o igual que 112
 cadena menor que 111
 coma 90
 comentario en bloque 89
 comment 91
 concatenación de cadenas 86
 condicionales 92
 delimitador de cadena 108
 desigualdad de cadenas 111
 desigualdad numérica 105
 división 93
 igualdad de cadenas 108
 igualdad numérica 103
 incremento 95
 mayor o igual que 104
 mayor que 103
 módulo 99
 multiplicar 101
 NOT lógico 97
 numérico menor o igual que 106
 numérico menor que 105
 OR lógico 98
 punto 94
 suma numérica 102
OR lógico, operador 98

OR, operador 98
ord(), función 39

P

play(), función 40
prevFrame(), función 40
prevScene(), función 41
propiedades
 _alpha 56
 _currentframe 56
 _focusrect 57
 _framesloaded 58
 _height 58
 _highquality 59
 _level 60
 _name 61
 _rotation 62
 _scroll 62
 _target 63
 _visible 64
 _width 65
 _x 65
 _xscale 66
 _y 67
 _yscale 68
 barra diagonal 55
 maxscroll 61
 scroll 62
punto, operador 94

R

random(), función 42
removeMovieClip(), función 43
_rotation, propiedad 62

S

scroll, propiedad 62
sentencias
 break 70
 case 71
 continue 72
 do..while 74
 else 74
 else if 75
 for 76
 if 77

 NOT lógico 97
 switch 78
 while 80
set(), función 43
setProperty(), función 44
sonido MFI 123
sonido MIDI 123
stop(), función 45
stopAllSounds(), función 46
String(), función 46
substring(), función 47
suma numérica 102
switch, sentencia 78

T

_target, propiedad 63
tellTarget(), función 48
toggleHighQuality(), función 49
_totalframes, propiedad 63
trace(), función 49

U

unloadMovie(), función 50
unloadMovieNum(), función 51

V

variable de capacidad de correo electrónico 121
variables
 \$version 128
 _alpha 56
 _cap4WayKeyAS 127
 _capCompoundSound 121
 _capEmail 121
 _capLoadData 122
 _capMFi 123
 _capMIDI 123
 _capMMS 124
 _capSMAF 125
 _capSMS 126
 _capStreamSound 127
 capacidad de correo electrónico 121
 capacidad para cargar datos 122
 navegación con teclas de flecha 127
 número de versión de Flash Lite 128
 variables de mensajes 124, 126

variables de sonido 121, 123, 125, 127
 _capCompoundSound 121
 _capMFi 123
 _capMIDI 123
 _capSMAF 125
 _capStreamSound 127
_visible, propiedad 64

W

while, bucle 74
while, sentencia 80
_width, propiedad 65

X

_x, propiedad 65
_xscale, propiedad 66

Y

_y, propiedad 67
_yscale, propiedad 68

