



Flash Lite 1.x ActionScript 语言参考

8

商标

1 Step RoboPDF、ActiveEdit、ActiveTest、Authorware、Blue Sky Software、Blue Sky、Breeze、Breezo、Captivate、Central、ColdFusion、Contribute、Database Explorer、Director、Dreamweaver、Fireworks、Flash、FlashCast、FlashHelp、Flash Lite、FlashPaper、Flash Video Encoder、Flex、Flex Builder、Fontographer、FreeHand、Generator、HomeSite、JRun、MacRecorder、Macromedia、MXML、RoboEngine、RoboHelp、RoboInfo、RoboPDF、Roundtrip、Roundtrip HTML、Shockwave、SoundEdit、Studio MX、UltraDev 和 WebHelp 是 Macromedia, Inc. 的注册商标或商标，可能已经在美国或其它司法辖区（包括全球范围）内注册。本出版物中提到的其它产品名称、徽标、图案、标题、文字或短语可能是 Macromedia, Inc. 或其它实体的商标、服务标志或商品名称，并且可能已经在特定的管辖区甚至世界范围内注册。

第三方信息

本指南包含指向第三方网站的链接，这些网站不在 Macromedia 的控制之下，Macromedia 不对所链接的任何站点的内容负责。如果您访问本指南提及的第三方网站，您需要自担风险。Macromedia 提供这些链接只是为您提供方便。包含这些链接并不意味着 Macromedia 为这些第三方站点的内容提供担保或承担责任。

语音压缩和解压缩技术已得到 Nellymoser, Inc. (www.nellymoser.com) 的许可。



Sorenson™ Spark™ 视频压缩和解压缩技术得到了 Sorenson Media, Inc. 的许可。

Opera® 浏览器版权所有 © 1995-2002 Opera Software ASA 和其提供商。保留所有权利。

Macromedia Flash 8 视频由 On2 TrueMotion 视频技术提供支持。© 1992-2005 On2 Technologies, Inc. 保留所有权利。 <http://www.on2.com>。

Visual SourceSafe 是 Microsoft Corporation 在美国和 / 或其它国家（地区）的注册商标或商标。

版权所有 © 2005 Macromedia, Inc. 保留所有权利。未经 Macromedia, Inc. 书面许可，本手册及其任何部分都不允许拷贝、影印、复制、翻译或转换成任何电子形式或机器可读的形式。尽管有以上规定，与本手册一起提供的软件有效副本的所有者或授权用户可以为本手册的电子版本打印一份副本，该副本只能供该所有者或授权用户学习使用该软件之用，禁止对本手册的任何部分进行打印、复制、分发、转售或传送以用于其它任何目的，包括（但不限于）商业目的，如销售本文档的副本或提供有偿支持服务。

致谢

项目管理：Mary Leigh Burke

撰稿：Guy Haas、Denise Green、Mike Krisher

总编：Rosana Francescato

编辑：Linda Adler、Geta Carlson、Evelyn Eldridge

制作管理：Patrice O'Neill、Kristin Conradi、Yuko Yagi

媒体设计和制作：Adam Barnett、Aaron Begley、Paul Benkman、John Francis、Geeta Karmarkar、Masayo Noda、Paul Rangel、Arena Reed、Mario Reynoso

特别感谢 Lisa Friendly、Bonnie Loo、Erick Vera、测试版测试人员以及 Flash Lite 工程小组和质量保证小组的全体成员。

第一版：2005 年 9 月

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

目 录

简介.....	7
大多数 ActionScript 元素的示例条目	7
Samples 文件夹	8
印刷惯例	8
 第 1 章：Flash Lite 全局函数.....	9
call()	11
chr()	12
duplicateMovieClip()	12
eval()	13
getProperty()	14
getTimer()	15
getURL()	15
gotoAndPlay()	18
gotoAndStop()	19
ifFrameLoaded()	19
int()	20
length()	21
loadMovie()	21
loadMovieNum()	22
loadVariables()	23
loadVariablesNum()	25
mbchr()	26
mblength()	26
mbord()	27
mbsubstring()	28
nextFrame()	28
nextScene()	29
Number()	30
on()	31
ord()	32
play()	32
prevFrame()	33
prevScene()	33
random()	34
removeMovieClip()	35

set()	35
setProperty()	36
stop()	37
stopAllSounds()	37
String()	38
substring()	38
tellTarget()	39
toggleHighQuality()	40
trace()	40
unloadMovie()	41
unloadMovieNum()	42
第 2 章：Flash Lite 属性	43
/（正斜杠）	44
_alpha	45
_currentframe	45
_focusrect	46
_framesloaded	46
_height	47
_highquality	47
_level	48
maxscroll	49
_name	49
_rotation	50
scroll	50
_target	51
_totalframes	51
_visible	52
_width	52
_x	53
_xscale	53
_y	54
_yscale	55
第 3 章：Flash Lite 语句	57
break	58
case	58
continue	60
do..while	61
else	62
else if	63
for	64
if	65
switch	65
while	67

第 4 章：Flash Lite 运算符.....69

add（字符串连接）	71
+=（加法赋值）	72
and	72
=（赋值）	73
/*（块注释）	74
,（逗号）	75
//（注释）	76
?:（条件）	77
--（递减）	77
/（除法）	78
/=（除法赋值）	79
.（点）	79
++（递增）	80
&&（逻辑 AND）	81
!（逻辑 NOT）	82
（逻辑 OR）	83
%（模）	84
%=（模赋值）	84
*=（乘法赋值）	85
*（乘法）	86
+（数字加法）	86
==（数值等于）	87
>（数值大于）	88
>=（数值大于或等于）	88
<>（数值不等于）	89
<（数值小于）	90
<=（数值小于或等于）	90
()（括号）	91
""（字符串分隔符）	92
eq（字符串相等）	93
gt（字符串大于）	93
ge（字符串大于或等于）	94
ne（字符串不等于）	95
lt（字符串小于）	95
le（字符串小于或等于）	96
-（减法）	97
-=（减法赋值）	98

第 5 章：特定于 Flash Lite 的语言元素	99
功能	101
fscommand()	109
fscommand2()	110
索引	141

简介

本手册说明了 **ActionScript** 元素的语法和用法，这些元素可用于开发 **Flash Lite 1.0** 和 **Flash Lite 1.1**（统称为 **Flash Lite 1.x**）应用程序。**Flash Lite 1.x ActionScript** 基于 **Flash 4** 中使用的 **ActionScript** 版本。要在脚本中使用示例，请从本手册中复制代码示例，然后将其粘贴到“脚本”面板或外部脚本文件中。本手册列出了所有 **ActionScript** 元素，包括运算符、关键字、语句、命令、属性、函数、类和方法。

大多数 ActionScript 元素的示例条目

以下示例条目解释了对所有 **ActionScript** 元素使用的惯例。

条目标题

章内的条目按字母顺序列出。这种字母顺序不考虑大写、前导下划线等。

可用性

除非另有说明，否则，“可用性”部分中将指出哪些版本的 **Flash Lite** 支持该元素。

用法

这一部分提供在代码中使用该 **ActionScript** 元素的正确语法。需要的语法部分以代码字体表示。您提供的代码和数据类型信息以*斜体代码字体*表示。可以通过始终位于数据类型信息之前的冒号 (:) 来区分数据类型和您提供的代码。括号 ([]) 表示可选参数。

操作数

这一部分说明在语法中列出的所有参数。

说明

这一部分标识元素类型（例如，运算符、函数等）、元素返回的值（如果有），并说明如何使用该元素。

示例

这一部分提供代码示例，演示如何使用该元素。

另请参见

这一部分列出相关的 **ActionScript** 字典条目。

Samples 文件夹

可以在 Flash 8 安装目录中的 /Samples and Tutorials/Samples/FlashLite/ 目录中找到一组示例文件。

此文件夹的常见路径如下：

- Windows: /Program Files/Macromedia/Flash 8/Samples and Tutorials/Samples/FlashLite/
- Macintosh: HD/Applications/Macromedia/Flash 8/Samples and Tutorials/Samples/FlashLite/

FlashLite 文件夹包含一组 **FLA** 文件，它们是使用 **ActionScript** 代码的已完成的 **Flash Lite** 项目。

印刷惯例

在本手册中采用以下排版印刷约定：

- 代码字体表示 **ActionScript** 代码。
- *斜体代码字体* 表示 **ActionScript** 参数。
- **粗体字体** 表示完全按原样采用的条目。
- 代码示例中的双引号 (" ") 表示分隔的字符串。但程序员也可以使用单引号。

本部分说明 Macromedia Flash Lite 1.1 ActionScript 全局函数的语法和用法。其中包括以下函数：

函数	说明
<code>call()</code>	执行被调用的帧中的脚本，而不将播放头移动到该帧。
<code>chr()</code>	将 ASCII 码数字转换为字符。
<code>duplicateMovieClip()</code>	在播放 SWF 文件时创建一个影片剪辑的实例。
<code>eval ()</code>	按名称访问变量、属性、对象或影片剪辑。
<code>getProperty()</code>	返回指定影片剪辑的指定属性的值。
<code>getTimer()</code>	返回自 SWF 文件开始播放时起已经过的毫秒数。
<code>getURL()</code>	将来自特定 URL 的文档加载到窗口中，或将变量传递到位于所定义 URL 的另一个应用程序。
<code>gotoAndPlay()</code>	将播放头转到场景中指定的帧并从该帧开始播放。如果未指定场景，则播放头将移动到当前场景中的指定帧。
<code>gotoAndStop()</code>	将播放头转到场景中指定的帧并停止播放。如果未指定场景，则播放头将转到当前场景中的帧。
<code>iffFrameLoaded()</code>	检查特定帧的内容是否可在本地使用。
<code>int()</code>	将小数截断为整数值。
<code>length()</code>	返回指定字符串或变量名的字符数。
<code>loadMovie()</code>	在播放原始加载的 SWF 文件时将一个 SWF 文件加载到 Flash Lite 中。
<code>loadMovieNum()</code>	在播放原始加载的 SWF 文件时将一个 SWF 文件加载到 Flash Lite 中的某个级别。
<code>loadVariables()</code>	从外部文件（如文本文件或由 ColdFusion、CGI、ASP、PHP 或 Perl 脚本生成的文本）中读取数据，并设置 Flash Lite 级别中变量的值。此函数还可以用新值更新当前 SWF 文件中的变量。

函数	说明
<code>loadVariablesNum()</code>	从外部文件（如文本文件或由 ColdFusion、CGI、ASP、PHP 或 Perl 脚本生成的文本）中读取数据，并设置 Flash Lite 级别中变量的值。此函数还可以用新值更新当前 SWF 文件中的变量。
<code>mbchr()</code>	将 ASCII 码数字转换为多字节字符。
<code>mblength()</code>	返回多字节字符串的长度。
<code>mbord()</code>	将指定字符转换为多字节数字。
<code>mbsubstring()</code>	从多字节字符串中提取一个新的多字节字符串。
<code>nextFrame()</code>	将播放头转到下一帧并停止。
<code>nextScene()</code>	将播放头转到下一场景的第 1 帧并停止。
<code>Number()</code>	将表达式转换为数字并返回值。
<code>on()</code>	指定触发某个事件的用户事件或按键。
<code>ord()</code>	将字符转换为 ASCII 码数字。
<code>play()</code>	在时间轴中向前移动播放头。
<code>prevFrame()</code>	将播放头转到前一帧并停止。如果当前帧为第 1 帧，则播放头不移动。
<code>prevScene()</code>	将播放头移到前一场景的第 1 帧并停止。
<code>removeMovieClip()</code>	删除原来使用 <code>duplicateMovieClip()</code> 创建的指定影片剪辑。
<code>set()</code>	为变量赋值。
<code>setProperty()</code>	当影片播放时，更改影片剪辑的属性值。
<code>stop()</code>	停止当前正在播放的 SWF 文件。
<code>stopAllSounds()</code>	在不停止播放头的情况下停止 SWF 文件中当前正在播放的所有声音。
<code>String()</code>	返回指定参数的字符串表示形式。
<code>substring()</code>	提取部分字符串。
<code>tellTarget()</code>	将 <i>statement</i> 参数中指定的指令应用于 <i>target</i> 参数中指定的时间轴。
<code>toggleHighQuality()</code>	在 Flash Lite 中打开和关闭消除锯齿功能。消除锯齿可使对象的边缘平滑，但会减缓 SWF 文件的播放速度。
<code>trace()</code>	在测试模式中，计算表达式并在“输出”面板中显示结果。
<code>unloadMovie()</code>	从 Flash Lite 中删除通过 <code>loadMovie()</code> <code>loadMovieNum()</code> 或 <code>duplicateMovieClip()</code> 方式加载的影片剪辑。
<code>unloadMovieNum()</code>	从 Flash Lite 的某个级别中删除通过 <code>loadMovie()</code> <code>loadMovieNum()</code> 或 <code>duplicateMovieClip()</code> 方式加载的影片剪辑。

call()

可用性

Flash Lite 1.0。

用法

`call(frame)`

`call(movieClipInstance: frame)`

操作数

frame 时间轴中帧的标签或编号。

movieClipInstance 影片剪辑的实例名称。

说明

函数；执行被调用帧中的脚本，而不将播放头移动到该帧。在脚本执行后，局部变量将消失。

`call()` 函数可能采用两种形式：

- 默认形式在执行 `call()` 函数的同一个时间轴的指定帧上执行脚本，而不将播放头移动到该帧。
- 指定影片剪辑实例形式在影片剪辑实例的指定帧上执行脚本，而不将播放头移动到该帧。



`call()` 函数同步执行：在指定帧上的所有 **ActionScript** 完成之前，不会执行 `call()` 函数后面的任何 **ActionScript**。

示例

以下示例执行 `myScript` 帧中的脚本：

```
// 执行标签为 “myScript” 的帧中的函数
thisFrame = "myScript";
trace ("Calling the script in frame: " + thisFrame);
```

```
// 执行同一时间轴上任何其它帧中的函数
call("myScript");
```

chr()

可用性

Flash Lite 1.0。

用法

`chr(number)`

操作数

number ASCII 码数字。

说明

字符串函数：将 ASCII 码数字转换为字符。

示例

以下示例将数字 65 转换为字母 A，并将其赋给变量 myVar：

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

duplicateMovieClip()

可用性

Flash Lite 1.0。

用法

`duplicateMovieClip(target, newname, depth)`

操作数

target 要直接复制的影片剪辑的目标路径。

newname 已直接复制的影片剪辑的唯一标识符。

depth 已直接复制的影片剪辑的唯一深度级别。深度级别表示直接复制的影片剪辑的堆叠顺序。这种堆叠顺序很像时间轴中图层的堆叠顺序；较低深度级别的影片剪辑隐藏在较高深度级别的剪辑之下。您必须为每个直接复制的影片剪辑指定唯一的深度级别，使其不会覆盖已占用的深度级别上的现有影片剪辑。

说明

函数：在播放 SWF 文件期间创建影片剪辑实例，并返回对被直接复制的影片剪辑的引用。无论播放头在原始（父级）影片剪辑中处于什么位置，直接复制的影片剪辑的播放头始终从第 1 帧开始。父级影片剪辑中的变量不复制到直接复制的影片剪辑中。如果删除父级影片剪辑，则直接复制的影片剪辑也被删除。使用 `removeMovieClip()` 函数或方法可以删除用 `duplicateMovieClip()` 创建的影片剪辑实例。

示例

以下示例在级别为 10 的深度上直接复制一个名为 `originalClip` 的影片剪辑，来创建名为 `newClip` 的新剪辑。新剪辑的 `x` 位置设置为 100 像素。

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

以下示例在一个 `for` 循环中使用 `duplicateMovieClip()`，以一次创建多个新影片剪辑。索引变量会保持跟踪已占用的最高堆叠深度。每个直接复制的影片剪辑的名称都包含一个与其堆叠深度相对应的数字后缀（`clip1`、`clip2`、`clip3`）。

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

另请参见

[removeMovieClip\(\)](#)

eval()

可用性

Flash Lite 1.0。

用法

`eval(expression)`

操作数

expression 一个字符串，它包含要检索的变量、属性、对象或影片剪辑的名称。

说明

函数：按名称访问变量、属性、对象或影片剪辑。如果 *expression* 是变量或属性，则返回其值。如果 *expression* 是对象或影片剪辑，则返回指向它的引用。如果无法找到 *expression* 中指定的元素，则返回 `undefined`。

您可以使用 `eval()` 来模拟数组，或者动态设置和检索变量的值。

示例

以下示例使用 `eval()` 确定表达式 `"piece" + x` 的值。因为该结果是变量名 `piece3`，所以 `eval()` 返回该变量的值，并将其赋给 `y`：

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

以下示例演示如何模拟数组：

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

getProperty()

可用性

Flash Lite 1.0。

用法

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

操作数

my_mc 要检索其属性的影片剪辑的实例名称。

property 影片剪辑的属性。

说明

函数：返回影片剪辑 `my_mc` 的指定属性的值。

示例

以下示例在根影片时间轴中检索影片剪辑 `my_mc` 的水平轴坐标 (`_x`)：

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

另请参见

[setProperty\(\)](#)

getTimer()

可用性

Flash Lite 1.0。

用法

```
getTimer()
```

操作数

无。

说明

函数；返回自 SWF 文件开始播放时起已经过的毫秒数。

示例

以下示例将 `timeElapsed` 变量设置为自 SWF 文件开始播放时起经过的毫秒数：

```
timeElapsed = getTimer();  
trace (timeElapsed);// 输出: 影片已经播放的毫秒数
```

getURL()

可用性

Flash Lite 1.0。

用法

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

操作数

`url` 可从中获取文档的 URL。

`window` 可选参数，它指定应将文档加载到其中的窗口或 HTML 框架。



`window` 参数不是为 Flash Lite 应用程序指定的，因为手机上的浏览器不支持多个窗口。

您可以输入空字符串或特定窗口的名称，或者从下面的保留目标名称中选择：

- `_self` 指定当前窗口中的当前框架。
- `_blank` 指定一个新窗口。
- `_parent` 指定当前框架的父级。
- `_top` 指定当前窗口中的顶级框架。

variables 用于发送变量的 GET 或 POST 方法。如果没有变量，则省略此参数。GET 方法将变量追加到 URL 的末尾，它用于发送少量的变量。POST 方法在单独的 HTTP 标头中发送变量，它用于发送大量的变量。

说明

函数：将来自特定 URL 的文档加载到窗口中，或将变量传递到位于所定义 URL 的另一个应用程序。要测试此函数，请确保要加载的文件位于指定的位置。要使用绝对 URL（例如 <http://www.myserver.com>），则需要网络连接。

Flash Lite 1.0 仅识别 HTTP、HTTPS、mailto 和 tel 协议。Flash Lite 1.1 除了识别这几种协议外，还能识别 file、SMS（短消息服务）和 MMS（多媒体消息服务）协议。

Flash Lite 将调用传递到操作系统，而操作系统会用指定协议的已注册的默认应用程序处理该调用。

每一帧或每个事件处理函数只能处理一个 `getURL()` 函数。

某些手机将 `getURL()` 函数限制为仅处理按键事件。在这种情况下，只有在按键事件处理函数中触发了 `getURL()` 调用时，才会处理它。即使在这种情况下，每个事件处理函数也只能处理一个 `getURL()` 函数。

示例

在下面的 **ActionScript** 中，Flash Lite 播放器会在默认的浏览器中打开 **mobile.macromedia.com**：

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

您也可以使用 GET 或 POST 从当前时间轴中发送变量。以下示例使用 GET 方法向 URL 中追加变量：

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

以下 **ActionScript** 使用 POST 在 HTTP 标头中发送变量：

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


可以分配一个按钮函数来打开一个 address、subject 和 body 文本字段已填充的电子邮件合成窗口。使用以下方法之一可以分配一个按钮函数：方法 1 用于 **Shift-JIS** 或英文字符编码；方法 2 仅用于英文字符编码。

方法 1：为每个需要的参数设置变量，如本例中所示：

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

方法 2：在 getURL() 函数内定义每个参数，如本例所示：

```
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");
}
```

方法 1 生成自动 URL 编码，而方法 2 保留字符串中的空格。例如，使用方法 1 生成的字符串如下所示：

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

与此相对照，方法 2 生成以下字符串：

```
email subject
email body
```

以下示例使用 **tel** 协议：

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

在下面的示例中，getURL() 用于发送 **SMS** 消息：

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

在下面的示例中，getURL() 用于发送 **MMS** 消息：

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

在下面的示例中，`getUrl()` 用于打开存储在本地文件系统上的文本文件：

```
// file 协议示例
filePath = "file://c:/documents/flash/myApp/myvariables.txt";
on(keyPress "7") {
    getUrl(filePath);
}
```

gotoAndPlay()

可用性

Flash Lite 1.0。

用法

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

操作数

scene 一个可选字符串，它指定播放头要转到的场景的名称。

frame 表示将播放头转到的帧编号的数字，或者表示将播放头转到的帧的标签的字符串。

说明

函数：将播放头转到场景中指定的帧并从该帧开始播放。如果未指定场景，则播放头将移动到当前场景中的指定帧。

您只能在根时间轴上使用 *scene* 参数，而不能在文档中的影片剪辑或其它对象的时间轴内使用。

示例

在下面的示例中，当用户单击已为其分配 `gotoAndPlay()` 的按钮时，播放头会移动到当前场景中的第 16 帧并开始播放 SWF 文件：

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

gotoAndStop()

可用性

Flash 1.0。

用法

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

操作数

scene 一个可选字符串，它指定播放头要转到的场景的名称。

frame 表示将播放头转到的帧编号的数字，或者表示将播放头转到的帧的标签的字符串。

说明

函数：将播放头转到场景中指定的帧并停止播放。如果未指定场景，则播放头将转到当前场景中的帧。

您只能在根时间轴上使用 *scene* 参数，而不能在文档中的影片剪辑或其它对象的时间轴内使用。

示例

在下面的示例中，当用户单击已为其分配 gotoAndStop() 的按钮时，播放头将转到当前场景中的第 5 帧并且停止播放 SWF 文件：

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

ifFrameLoaded()

可用性

Flash Lite 1.0。

用法

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

操作数

scene 一个可选字符串，它指定要加载的场景的名称。

frame 在执行下一条语句之前必须加载的帧编号或帧标签。

statement 在加载了指定的帧或场景及帧时要执行的指令。

说明

函数：检查特定帧的内容是否可在本地使用。使用 `ifFrameLoaded` 函数可在 SWF 文件的其余部分向本地计算机下载的同时开始播放简单的动画。您还可以使用 `_framesloaded` 属性来检查外部 SWF 文件的下载进度。使用 `_framesloaded` 和 `ifFrameLoaded` 的区别在于 `_framesloaded` 允许您添加自定义的 `if` 或 `else` 语句。

示例

以下示例使用 `ifFrameLoaded` 函数检查是否已加载 SWF 文件的第 10 帧。如果已加载该帧，则 `trace()` 命令会向“输出”面板中打印“**frame number 10 is loaded**”（已加载帧编号 10）。并且将变量 `output` 赋值为：`frame loaded: 10`。

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

另请参见

[_framesloaded](#)

int()

可用性

Flash Lite 1.0。

用法

`int(value)`

操作数

value 一个要截断为整数的数字或字符串。

说明

函数：将小数截断为整数值。

示例

以下示例将截断 `distance` 和 `myDistance` 变量中的数字：

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

length()

可用性

Flash Lite 1.0。

用法

`length(expression)`

`length(variable)`

操作数

expression 字符串。

variable 变量的名称。

说明

字符串函数；返回指定字符串或变量名的字符数。

示例

以下示例返回字符串 "Hello" 的长度：

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

结果为 5。

以下示例通过检查电子邮件地址是否包含至少六个字符来验证电子邮件地址：

```
email = "someone@macromedia.com";  
if (length(email) > 6) {  
    //trace ("email appears to have enough characters to be valid");  
}
```

loadMovie()

可用性

Flash Lite 1.1。

用法

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

操作数

url 一个字符串，它指定要加载的 SWF 文件的绝对或相对 URL。相对路径必须相对于级别 0 处的 SWF 文件。绝对 URL 必须包括协议引用，如 `http://` 或 `file:///`。

target 对影片剪辑的引用或者表示指向目标影片剪辑的路径的字符串。用已加载的 SWF 文件替换目标影片剪辑。

method 可选字符串参数，指定用于发送变量的 HTTP 方法。该参数必须是字符串 GET 或 POST。如果没有要发送的变量，则省略此参数。GET 方法将变量追加到 URL 的末尾，它用于发送少量的变量。POST 方法在单独的 HTTP 标头中发送变量，它用于发送大量的变量。

说明

函数：在播放原始 SWF 文件时将一个 SWF 文件加载到 Flash Lite 中。

若要将一个 SWF 文件加载到特定级别，请使用 `loadMovieNum()` 函数替代 `loadMovie()`。

如果 SWF 文件加载到目标影片剪辑，则可使用该影片剪辑的目标路径来定位加载的 SWF 文件。加载到目标的 SWF 文件会继承目标影片剪辑的位置、旋转和缩放属性。加载的图像或 SWF 文件的左上角与目标影片剪辑的注册点对齐。但是，如果目标为根时间轴，则该图像或 SWF 文件的左上角与舞台的左上角对齐。

使用 `unloadMovie()` 函数可删除用 `loadMovie()` 加载的 SWF 文件。

示例

以下示例从同一目录中加载 SWF 文件 `circle.swf`，并替换舞台上已存在的名为 `mySquare` 的影片剪辑：

```
loadMovie("circle.swf", "mySquare");  
// 等效语句: loadMovie("circle.swf", _level0.mySquare);
```

另请参见

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

loadMovieNum()

可用性

Flash Lite 1.1。

用法

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

操作数

url 一个字符串，它指定要加载的 SWF 文件的绝对或相对 URL。相对路径必须相对于级别 0 处的 SWF 文件。为了在独立的 Flash Lite 播放器中使用 SWF 文件或在 Flash 创作应用程序的测试模式中使用 SWF 文件，必须将所有的 SWF 文件存储在同一个文件夹中，并且其文件名不能包括文件夹或驱动器说明。

level 一个整数，它指定 Flash Lite 中加载 SWF 文件的级别。

method 可选字符串参数，指定用于发送变量的 HTTP 方法。其值必须为 GET 或 POST。如果没有要发送的变量，则省略此参数。GET 方法将变量追加到 URL 的末尾，它用于发送少量的变量。POST 方法在单独的 HTTP 标头中发送变量，它用于发送大量的变量。

说明

函数：在播放原始加载的 SWF 文件时将一个 SWF 文件加载到 Flash Lite 中的某个级别。

通常，Flash Lite 会显示单个 SWF 文件，然后关闭。使用 loadMovieNum() 函数可以一次显示多个 SWF 文件，并且无需加载另一个 HTML 文档即可在 SWF 文件之间进行切换。

若要指定目标而不是级别，请使用 loadMovie() 函数代替 loadMovieNum()。

Flash Lite 具有从级别 0 开始的级别堆叠顺序。这些级别类似于醋酸纤维层；除了每个级别上的对象之外，它们是透明的。当使用 loadMovieNum() 时，必须指定 SWF 文件将加载到 Flash Lite 中的哪个级别。在 SWF 文件加载到某个级别后，即可使用语法 `_levelN` 来定位 SWF 文件，其中 *N* 为级别号。

在加载 SWF 文件时，可以指定任何级别号。您可以将 SWF 文件加载到其中已加载了一个 SWF 文件的某个级别，并用新的 SWF 文件替换现有文件。如果将 SWF 文件加载到级别 0，则 Flash Lite 中的每个级别均被卸载，并且用新文件替换级别 0。处于级别 0 的 SWF 文件为所有其它加载的 SWF 文件设置帧频、背景色和帧大小。

使用 unloadMovieNum() 可删除用 loadMovieNum() 加载的 SWF 文件或图像。

示例

以下示例将 SWF 文件加载到级别 2：

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

另请参见

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

loadVariables()

可用性

Flash Lite 1.1。

用法

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

操作数

url 一个字符串，它表示变量所处位置的绝对或相对 URL。如果发布此调用的 SWF 文件运行在 Web 浏览器中，则 *url* 必须与该 SWF 文件位于同一个域中。

target 指向接收所加载变量的影片剪辑的目标路径。

variables 可选字符串参数，它指定用于发送变量的 HTTP 方法。该参数必须是字符串 GET 或 POST。如果没有要发送的变量，则省略此参数。GET 方法将变量追加到 URL 的末尾，它用于发送少量的变量。POST 方法在单独的 HTTP 标头中发送变量，它用于发送大量的变量。

说明

函数：从外部文件（如文本文件或由 ColdFusion、CGI、Active Server Pages (ASP)、PHP 或 Perl 脚本生成的文本）中读取数据，并设置目标影片剪辑中变量的值。此函数还可以用新值更新活动 SWF 文件中的变量。

指定 URL 处的文本必须为标准的 MIME 格式 `application/x-www-form-urlencoded`（CGI 脚本所使用的一种标准格式）。可以指定任意数量的变量。例如，下面的语句定义了几个变量：

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

若要将变量加载到特定级别，请使用 `loadVariablesNum()` 函数代替 `loadVariables()` 函数。

示例

以下示例从文本文件和服务器加载变量：

```
// 从本地文件系统上的文本文件加载变量 (Symbian Series 60)
on(release, keyPress "1") {
    filePath = "file:///c:/documents/flash/myApp/myvariables.txt";
    loadVariables(filePath, _root);
}
```

```
// 将变量（来自服务器）加载到影片剪辑
urlPath = "http://www.someserver.com/myvariables.txt";
loadVariables(urlPath, "myClip_mc");
```

另请参见

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

loadVariablesNum()

可用性

Flash Lite 1.1。

用法

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

操作数

url 一个字符串，它表示要加载的变量所处位置的绝对或相对 URL。如果发布此调用的 SWF 文件运行在 Web 浏览器中，则 *url* 必须与该 SWF 文件位于同一个域中；有关更多信息，请参见下面的“说明”部分。

level 一个整数，它指定 Flash Lite 中接收这些变量的级别。

variables 可选字符串参数，它指定用于发送变量的 HTTP 方法。该参数必须是字符串 GET 或 POST。如果没有要发送的变量，则省略此参数。GET 方法将变量追加到 URL 的末尾，它用于发送少量的变量。POST 方法在单独的 HTTP 标头中发送变量，它用于发送大量的变量。

说明

函数：从外部文件（如文本文件或由 ColdFusion、CGI、ASP、PHP 或 Perl 脚本生成的文本）中读取数据，并设置 Flash Lite 级别中变量的值。此函数还可以用新值更新活动 SWF 文件中的变量。

指定 URL 处的文本必须为标准的 MIME 格式 `application/x-www-form-urlencoded`（CGI 脚本所使用的一种标准格式）。可以指定任意数量的变量。下面的示例语句定义了几个变量：

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

通常，Flash Lite 会显示单个 SWF 文件，然后关闭。使用 `loadVariablesNum()` 函数可以一次显示多个 SWF 文件，并且无需加载另一个 HTML 文档即可在 SWF 文件之间进行切换。若要将变量加载到目标影片剪辑，请使用 `loadVariables()` 函数代替 `loadVariablesNum()` 函数。

另请参见

`getURL()`、`loadMovie()`、`loadMovieNum()`、`loadVariables()`

mbchr()

可用性

Flash Lite 1.0。

用法

`mbchr(number)`

操作数

number 要转换为多字节字符的数字。

说明

字符串函数：将 ASCII 码数字转换为多字节字符。

示例

以下示例将 ASCII 码数字转换为其多字节字符等效形式：

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

myString = mbchr(51) + mbchr(49);
trace ("result = " + myString); // 输出: result = 2
```

另请参见

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

mblength()

可用性

Flash Lite 1.0。

用法

`mblength(string)`

操作数

string 字符串。

说明

字符串函数：返回多字节字符串的长度。

示例

以下示例显示 `myString` 变量中字符串的长度：

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

另请参见

[mbchr\(\)](#)、[mbsubstring\(\)](#)

mbord()

可用性

Flash Lite 1.0。

用法

`mbord(character)`

操作数

character 要转换为多字节数字的字符。

说明

字符串函数：将指定的字符转换为多字节数字。

示例

以下示例将 `myString` 变量中的字符转换为多字节数字：

```
myString = "A";  
trace ("ord = " add mbord(myString));// 输出: 65  
  
myString = "$120";  
for (i=1; i<=length(myString); i++)  
    char = substring(myString, i, 1);  
    trace ("char ord = " add mbord(char));// 输出: 36, 49, 50, 48  
}
```

另请参见

[mbchr\(\)](#)、[mbsubstring\(\)](#)

mbsubstring()

可用性

Flash Lite 1.0。

用法

`mbsubstring(value, index, count)`

操作数

value 要从其中提取新的多字节字符串的多字节字符串。

index 要提取的第一个字符的编号。

count 要包含在所提取字符串中的字符的数目，其中不包括 **index** 处的字符。

说明

字符串函数：从多字节字符串中提取一个新的多字节字符串。

示例

以下示例从 `myString` 变量所包含的字符串中提取新的多字节字符串：

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

另请参见

[mbchr\(\)](#)

nextFrame()

可用性

Flash Lite 1.0。

用法

`nextFrame()`

操作数

无。

说明

函数：将播放头转到下一帧并停止。

示例

在下面的示例中，当用户单击按钮时，播放头会移动到下一帧并停止：

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

另请参见

[prevFrame\(\)](#)

nextScene()

可用性

Flash Lite 1.0。

用法

`nextScene()`

操作数

无。

说明

函数：将播放头转到下一场景的第 1 帧并停止。

示例

在下面的示例中，当用户释放按钮时，播放头会移动到下一场景的第 1 帧：

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

另请参见

[prevScene\(\)](#)

Number()

可用性

Flash Lite 1.0。

用法

`Number(expression)`

操作数

expression 要转换为数字的表达式。

说明

函数：将参数 *expression* 转换为数字，并按以下列表中所述返回一个值：

- 如果 *expression* 为数字，则返回值为 *expression*。
- 如果 *expression* 为布尔值，则当它为 `true` 时，返回值为 `1`；当它为 `false` 时返回值为 `0`。
- 如果 *expression* 为字符串，则该函数尝试将它解析为一个带有可选尾随指数的十进制数字（如 `1.57505e-3`）。
- 如果 *expression* 为 `undefined`，则返回值为 `-1`。

示例

以下示例将 `myString` 变量中的字符串转换为数字，将该数字存储在 `myNumber` 变量中，使该数字加 `5`，并将结果存储在变量 `myResult` 中。最后一行显示您用布尔值调用 `Number()` 时返回的结果。

```
myString = "55";  
myNumber = Number(myString);  
myResult = myNumber + 5;  
  
trace (myResult); // 输出: 60  
  
trace (Number(true)); // 输出: 1
```

on()

可用性

Flash Lite 1.0。

用法

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

操作数

statement *event* 发生时要执行的指令。

event 此触发器称为“事件”。当用户事件发生时，会执行该事件后面大括号 ({}) 中的语句。可以为 *event* 参数指定以下任意值：

- `press` 在鼠标指针位于按钮上时按下按钮。
- `release` 在鼠标指针位于按钮上时释放按钮。
- `rollOut` 鼠标指针滑出按钮区域。
- `rollOver` 鼠标指针滑过按钮。
- `keyPress "key"` 按下指定的键。对于此参数的 **key** 部分，需指定键控代码或键常量。

说明

事件处理函数：指定触发函数的用户事件或按键。并非所有事件都受支持。

示例

以下代码会在用户按 8 键时，将 `myText` 字段向下滚动一行，滚动之前会对 `maxscroll` 进行测试：

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

ord()

可用性

Flash Lite 1.0。

用法

`ord(character)`

操作数

character 要转换为 ASCII 码数字的字符。

说明

字符串函数：将字符转换为 ASCII 码数字。

示例

以下示例使用 `ord()` 函数显示字符 A 的 ASCII 码：

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

play()

可用性

Flash Lite 1.0。

用法

`play()`

操作数

无。

说明

函数：在时间轴中向前移动播放头。

示例

以下代码使用 `if` 语句检查用户输入的名称值。如果用户输入 Steve，则调用 `play()` 函数，而且播放头会在时间轴中向前移动。如果用户输入 Steve 以外的任何其它内容，则不播放 SWF 文件，而显示变量名为 `alert` 的文本字段。

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


prevFrame()

可用性

Flash Lite 1.0。

用法

prevFrame()

操作数

无。

说明

函数：将播放头转到前一帧并停止。如果当前帧为第 1 帧，则播放头不移动。

示例

当用户单击附加了以下处理函数的按钮时，播放头会转到上一帧：

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

另请参见

[nextFrame\(\)](#)

prevScene()

可用性

Flash Lite 1.0。

用法

prevScene()

操作数

无。

说明

函数：将播放头移到前一场景的第 1 帧并停止。

示例

在本例中，当用户单击附加了以下处理函数的按钮时，播放头会转到前一场景：

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

另请参见

[nextScene\(\)](#)

random()

可用性

Flash Lite 1.0。

用法

`random(value)`

操作数

value 一个整数。

说明

函数；返回一个随机整数，此整数介于 0 和一个小于 *value* 参数中指定的整数之间。

示例

以下示例基于指定范围的整数生成一个数字：

```
// 在 0 至 5 之间选择一个随机数字  
myNumber = random(5);  
trace (myNumber);/// 输出: 可能是 0、1、2、3、4
```

```
// 在 5 至 10 之间选择一个随机数字  
myNumber = random(5) + 5;  
trace (myNumber);// 输出: 可能是 5、6、7、8、9
```

以下示例生成一个数字，然后将它连接到计算为变量名的字符串的末端。此示例说明了如何使用 **Flash Lite 1.1** 语法模拟数组。

```
// 从列表中选择随机名称  
myNames1 = "Mike";  
myNames2 = "Debbie";  
myNames3 = "Logan";  
  
ran = random(3) + 1;  
ranName = "myNames" + ran;  
trace (eval(ranName));// 输出: 将为 mike、debbie 或 logan
```

removeMovieClip()

可用性

Flash Lite 1.0。

用法

```
removeMovieClip(target)
```

操作数

target 用 `duplicateMovieClip()` 创建的影片剪辑实例的目标路径。

说明

函数：删除原来使用 `duplicateMovieClip()` 创建的指定影片剪辑。

示例

以下示例删除名为 `second_mc` 的直接复制影片剪辑：

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

set()

可用性

Flash Lite 1.0。

用法

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

操作数

variable 保存 *expression* 参数值的标识符。

expression 赋给变量的值。

说明

语句：为变量赋值。**variable** 是保存数据的容器。容器本身始终不变，但内容可以更改。通过在播放 **SWF** 文件时更改变量的值，可以记录和保存有关用户所执行操作的信息、记录播放 **SWF** 文件时更改的值，或者计算某条件是 `true` 还是 `false`。

变量可以保存任何类型的数据（如字符串、数字、布尔值或影片剪辑）。每个 **SWF** 文件和影片剪辑的时间轴都有其自己的变量集，每个变量又都有其自己的独立于其它时间轴上的变量的值。

示例

以下示例设置一个名为 `orig_x_pos` 的变量，该变量存储 `ship` 影片剪辑的 *x* 轴原始位置，以便以后在 SWF 文件中将 **ship** 重置到其起始位置：

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

上面的代码与下面的代码效果相同：

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

setProperty()

可用性

Flash Lite 1.0。

用法

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

操作数

target 指向要设置其属性的影片剪辑实例名称的路径。

property 要设置的属性。

value 属性的新文本值。

expression 计算结果为属性新值的公式。

说明

函数：当影片播放时，更改影片剪辑的属性值。

示例

当用户单击与此事件处理函数相关联的按钮时，以下语句会将 `star` 影片剪辑的 `_alpha` 属性设置为 **30%**：

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

另请参见

[getProperty\(\)](#)

stop()

可用性

Flash Lite 1.0。

用法

stop()

操作数

无。

说明

函数；停止当前正在播放的 SWF 文件。此函数最通常的用法是用按钮控制影片剪辑。

示例

当用户单击与此事件处理函数相关联的按钮时，以下语句会调用 stop() 函数：

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

stopAllSounds()

可用性

Flash Lite 1.0。

用法

stopAllSounds()

操作数

无。

说明

函数；在不停止播放头的情况下停止 SWF 文件中当前正在播放的所有声音。设置到流的聲音在播放头移过它们所在的帧时将恢复播放。

示例

以下代码可以应用到一个按钮，当单击此按钮时，将停止播放 SWF 文件中的所有声音：

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

String()

可用性

Flash Lite 1.0。

用法

`String(expression)`

操作数

expression 要转换为字符串的表达式。

说明

函数：返回指定参数的字符串表示形式，如以下列表所述：

- 如果 *expression* 是数字，则返回的字符串为此数字的文本表示形式。
- 如果 *expression* 为字符串，则返回的字符串就是 *expression*。
- 如果 *expression* 为布尔值，则返回的字符串为 `true` 或 `false`。
- 如果 *expression* 是一个影片剪辑，则返回值是以斜杠 (/) 记号表示的该影片剪辑的目标路径。

示例

以下示例将 `birthYearNum` 设置为 **1976**，并使用 `String()` 函数将其转换为字符串，然后使用 `eq` 运算符将它与字符串 “1976” 进行比较。

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

substring()

可用性

Flash Lite 1.0。

用法

`substring(string, index, count)`

操作数

string 从中提取新字符串的字符串。

index 要提取的第一个字符的编号。

count 要包含在所提取字符串中的字符的数目，其中不包括 **index** 处的字符。

说明

函数：提取部分字符串。此函数是从 1 开始的，而 **String** 类方法是从 0 开始的。

示例

以下示例从字符串 “Hello World” 中提取前五个字符：

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

tellTarget()

可用性

Flash Lite 1.0。

用法

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

操作数

target 一个字符串，它指定要控制的时间轴的目标路径。

statement 当条件的计算结果为 true 时要执行的指令。

说明

函数：将 *statement* 参数中指定的指令应用于 *target* 参数中指定的时间轴。

tellTarget() 函数对导航控制很有帮助。可以将 tellTarget() 分配给在舞台上的其它位置停止或开始影片剪辑的按钮。还可以使影片剪辑转到此剪辑的特定帧。例如，可以将 tellTarget() 分配给舞台上停止或开始影片剪辑的按钮，或提示影片剪辑移动到特定帧的按钮。

示例

在下面的示例中，tellTarget() 控制主时间轴上的 ball 影片剪辑实例。ball 实例的第 1 帧为空白，并具有一个 stop() 函数，因此它在舞台上不可见。当用户按 5 键时，tellTarget() 会指示 ball 中的播放头转到开始播放动画的第 2 帧。

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

toggleHighQuality()

可用性

Flash Lite 1.0。

用法

`toggleHighQuality()`

操作数

无。

说明

函数：在 **Flash Lite** 中打开和关闭消除锯齿功能。消除锯齿可使对象的边缘平滑，但会减缓 SWF 文件的播放速度。此函数会影响 **Flash Lite** 中的所有 SWF 文件。

示例

以下代码可以应用到一个按钮，单击该按钮时会在开启和关闭消除锯齿功能间切换：

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

trace()

可用性

Flash Lite 1.0。

用法

`trace(expression)`

操作数

expression 要计算的表达式。在 **Flash** 创作工具中打开 SWF 文件（使用“测试影片”命令）时，*expression* 参数的值会出现在“输出”面板中。

说明

函数：在测试模式中，计算表达式并在“输出”面板中显示结果。

在测试 SWF 文件时，使用此函数可记录编程注释或者在“输出”面板中显示消息。使用 *expression* 参数可以检查是否存在某个条件，或者在“输出”面板中显示值。`trace()` 函数类似于 JavaScript 中的 `alert` 函数。

可以使用发布设置中的“省略跟踪动作”命令将 `trace()` 函数从导出的 SWF 文件中删除。

示例

以下示例使用 `trace()` 函数来观察 `while` 循环的行为：

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

unloadMovie()

可用性

Flash Lite 1.0。

用法

`unloadMovie(target)`

操作数

target 影片剪辑的目标路径。

说明

函数：从 Flash Lite 中删除通过 `loadMovie()`、`loadMovieNum()` 或 “`duplicateMovieClip()`” 方式加载的影片剪辑。

示例

用户按 **3** 键时，以下代码会做出响应，卸载主时间轴上的 `draggable_mc` 影片剪辑，并将 `movie.swf` 加载到文档堆栈的级别 **4** 中：

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

用户按 **3** 键时，以下示例会卸载已加载到级别 **4** 中的影片：

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

另请参见

[loadMovie\(\)](#)

unloadMovieNum()

可用性

Flash Lite 1.0。

用法

```
unloadMovieNum(level)
```

操作数

level 所加载影片的级别 (`_levelN`)。

说明

函数：从 Flash Lite 中删除通过 `loadMovie()`、`loadMovieNum()` 或 “`duplicateMovieClip()`” 方式加载的影片剪辑。

通常，Flash Lite 会显示单个 SWF 文件，然后关闭。使用 `loadMovieNum()` 函数可以一次显示多个 SWF 文件，并且无需加载另一个 HTML 文档即可在 SWF 文件之间进行切换。

另请参见

[loadMovieNum\(\)](#)

本部分说明 Macromedia Flash Lite 1.x 可以识别的属性。各个条目按字母顺序列出，忽略前导下划线。下表概述了这些属性：

属性	说明
/（正斜杠）	指定或返回对影片主时间轴的引用。
<code>_alpha</code>	返回影片剪辑的 Alpha 透明度值。
<code>_currentframe</code>	返回播放头在时间轴中所处的帧的编号。
<code>_focusrect</code>	指定在具有当前焦点的按钮或文本字段周围是否显示黄色矩形。
<code>_framesloaded</code>	返回从动态加载的 SWF 文件中已经加载的帧数。
<code>_height</code>	指定影片剪辑的高度，以像素为单位。
<code>_highquality</code>	指定应用于当前 SWF 文件的锯齿消除级别。
<code>_level</code>	返回对 <code>_levelN</code> 的根时间轴的引用。在使用 <code>loadMovieNum()</code> 函数将 SWF 文件加载到 Flash Lite 播放器以后，才可使用 <code>_level</code> 属性来定位这些文件。还可使用 <code>_levelN</code> 来定位由 <code>N</code> 所指定级别处的已加载 SWF 文件。
<code>maxscroll</code>	表明可滚动的文本字段中最后一行可见时，该字段中的第一个可见行的行号。
<code>_name</code>	返回影片剪辑的实例名称。该属性仅应用于影片剪辑，而不应用于主时间轴。
<code>_rotation</code>	返回影片剪辑距其原始方向的旋转程度，以度为单位。
<code>scroll</code>	控制与变量相关联的文本字段中的信息的显示。 <code>scroll</code> 属性定义文本字段开始显示内容的位置；设置此属性后，当用户滚动该文本字段时，Flash Lite 将更新此属性。
<code>_target</code>	返回影片剪辑实例的目标路径。
<code>_totalframes</code>	返回影片剪辑中的总帧数。
<code>_visible</code>	表明影片剪辑是否可见。
<code>_width</code>	返回影片剪辑的宽度，以像素为单位。
<code>_x</code>	包含设置影片剪辑 x 坐标的一个整数。
<code>_xscale</code>	设置从影片剪辑的注册点开始应用的该影片剪辑的水平缩放比例（百分比）。

属性	说明
<code>_y</code>	包含设置影片剪辑 <code>y</code> 坐标的一个整数，该坐标相对于父级影片剪辑的本地坐标。
<code>_yscale</code>	设置从影片剪辑注册点开始应用的该影片剪辑的垂直缩放比例（百分比）。

/（正斜杠）

可用性

Flash Lite 1.0

用法

/

/targetPath

/:varName

说明

标识符；指定或返回对影片主时间轴的引用。此属性提供的功能与 **Flash 5** 中由 `_root` 属性所提供的功能相似。

示例

若要在某个时间轴上指定一个变量，请结合使用斜杠语法 (/) 和冒号 (:)。

示例 1：主时间轴上的 `car` 变量：

```
/:car
```

示例 2：位于主时间轴上的影片剪辑实例 **mc1** 中的 `car` 变量：

```
/mc1/:car
```

示例 3：在主时间轴上的影片剪辑实例 **mc1** 中嵌套的影片剪辑实例 **mc2** 中的 `car` 变量：

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

示例 4：位于当前时间轴上的影片剪辑实例 **mc2** 中的 `car` 变量：

```
mc2/:car
```

_alpha

可用性

Flash Lite 1.0。

用法

my_mc._alpha

属性：由 *my_mc* 变量指定的影片剪辑的 **alpha** 透明度值。有效值为 **0**（完全透明）到 **100**（完全不透明），默认值为 **100**。对于 **_alpha** 设置为 **0** 的影片剪辑中的对象，尽管它们不可见，但它们是活动的。例如，您仍可以单击 **_alpha** 属性设置为 **0** 的影片剪辑中的按钮。

示例

当用户单击按钮时，下面的按钮事件处理函数代码会将 *my_mc* 影片剪辑的 **_alpha** 属性设置为 **30%**：

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

_currentframe

可用性

Flash Lite 1.0。

用法

my_mc._currentframe

说明

属性（只读）：返回由 *my_mc* 变量指定的时间轴中播放头所处的帧的编号。

示例

以下示例使用 **_currentframe** 属性和 **gotoAndStop()** 函数指示 *my_mc* 影片剪辑的播放头从其当前位置向前移动五帧：

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

另请参见

[gotoAndStop\(\)](#)

`_focusrect`

可用性

Flash Lite 1.0。

用法

```
_focusrect = Boolean;
```

说明

属性（全局）：指定在具有当前焦点的按钮或文本字段周围是否显示黄色矩形。默认值为 `true`，当用户按电话或移动设备上的上箭头键或下箭头键在 **SWF** 文件中的对象之间导航时，会在当前具有焦点的按钮或文本字段周围显示一个黄色矩形。如果不希望显示黄色矩形，请指定 `false`。

示例

以下示例禁用应用程序中显示黄色焦点矩形：

```
_focusrect = false;
```

`_framesloaded`

可用性

Flash Lite 1.0。

用法

```
my_mc:_framesloaded
```

说明

属性（只读）：从动态加载的 **SWF** 文件中已经加载的帧数。该属性可用于确定特定帧及其前面所有帧的内容是否已经加载，并且是否可在浏览器本地使用。它还可作为大 **SWF** 文件下载时的监视器。例如，在 **SWF** 文件中的指定帧完成加载之前，可能需要向用户显示一条消息以表明 **SWF** 文件正在加载。

示例

以下示例使用 `_framesloaded` 属性在所有帧都已加载后开始播放 **SWF** 文件。如果未加载完所有帧，则会按比例增大影片剪辑实例 `loader` 的 `_xscale` 属性以创建进度栏。

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

_height

可用性

Flash Lite 1.0。

用法

my_mc:_height

说明

属性（只读）：影片剪辑的高度，以像素为单位。

示例

当用户单击鼠标按钮时，下面的事件处理函数代码示例会设置影片剪辑的高度：

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

_highquality

可用性

Flash Lite 1.0。

用法

_highquality

说明

属性（全局）：指定应用于当前 **SWF** 文件的锯齿消除级别。指定 **2** 可获得最佳品质的锯齿消除。指定 **1** 可获得高品质的锯齿消除。指定 **0** 可防止锯齿消除。

示例

以下语句对当前 SWF 文件应用高品质锯齿消除：

```
_highquality = 1;
```

另请参见

[toggleHighQuality\(\)](#)

_level

可用性

Flash Lite 1.0。

用法

`_levelN`

说明

标识符；对 `_levelN` 的根时间轴的引用。在使用 `loadMovieNum()` 函数将 SWF 文件加载到 Flash Lite 播放器以后，才可使用 `_level` 属性来定位这些文件。还可使用 `_levelN` 来定位由 *N* 所指定级别处的已加载 SWF 文件。

加载到 Flash Lite 播放器的实例中的初始 SWF 文件会自动加载到 `_level0`。`_level0` 中的 SWF 文件为随后加载的所有 SWF 文件设置帧频、背景颜色和帧大小。然后，这些 SWF 文件堆叠在处于 `_level0` 的 SWF 文件之上的更高编号级别中。

您必须为每个通过使用 `loadMovieNum()` 函数加载到 Flash Lite 播放器中的 SWF 文件分配一个级别。您可按任意顺序分配级别。如果您分配的级别（包括 `_level0`）中已经包含 SWF 文件，则处于该级别的 SWF 文件将被卸载并替换为新的 SWF 文件。

示例

以下示例将一个 SWF 文件加载到级别 1 中，然后将已加载的 SWF 文件的播放头停在第 6 帧：

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

// 至少之后 1 帧
tellTarget(_level1) {
    gotoAndStop(6);
}
```

另请参见

[loadMovie\(\)](#)

maxscroll

可用性

Flash Lite 1.1

用法

variable_name:maxscroll

说明

属性（只读）：表明可滚动的文本字段中最后一行可见时，该字段中的第一个可见行的行号。maxscroll 属性与 scroll 属性一起使用，来控制文本字段中信息的显示方式。可检索该属性，但不能进行修改。

示例

当用户按 8 键时，以下代码会将 myText 文本字段向下滚动一行，滚动之前会对 maxscroll 进行测试：

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

另请参见

[scroll](#)

_name

可用性

Flash Lite 1.0。

用法

my_mc:_name

说明

属性；由 *my_mc* 指定的影片剪辑的实例名称。该属性仅应用于影片剪辑，而不应用于主时间轴。

示例

以下示例在“输出”面板中以字符串的形式显示 bigRose 影片剪辑的名称：

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

_rotation

可用性

Flash Lite 1.0。

用法

my_mc:_rotation

说明

属性；影片剪辑距其原始方向的旋转程度，以度为单位。从 **0** 到 **180** 的值表示顺时针方向的旋转；从 **0** 到 **-180** 的值表示逆时针方向的旋转。对于此范围之外的值，可以通过加上或减去 **360** 获得该范围内的值。例如，语句 `my_mc:_rotation = 450` 与 `my_mc:_rotation = 90` 等效。

示例

当用户按 **2** 键时，以下示例会将 `my_mc` 影片剪辑顺时针旋转 **15** 度：

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

scroll

可用性

Flash Lite 1.1。

用法

textFieldVariableName:scroll

说明

属性；控制与变量相关联的文本字段中的信息的显示。`scroll` 属性定义文本字段开始显示内容的位置；设置此属性后，当用户滚动该文本字段时，**Flash Lite** 将更新此属性。您可以使用 `scroll` 属性来创建滚动的文本字段或者将用户定向到长章节中的特定段落。

示例

用户每次单击 **2** 键时，以下代码会将 `myText` 文本字段向上滚动一行：

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

另请参见

[maxscroll](#)

_target

可用性

Flash Lite 1.0。

用法

my_mc._target

说明

属性（只读）；返回 *my_mc* 指定的影片剪辑实例的目标路径。

_totalframes

可用性

Flash Lite 1.0。

用法

my_mc._totalframes

说明

属性（只读）；返回 *my_mc* 影片剪辑中的总帧数。

示例

以下代码将 **mySWF.swf** 加载到级别 1，然后在 25 帧后，检查它是否已加载：

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

// 主时间轴中的 25 帧后
if (_level1._framesloaded >= _level1._totalframes) {
    tellTarget("_level1/") {
        gotoAndStop("myLabel");
    }
} else {
    // 循环 ...
}
```

_visible

可用性

Flash Lite 1.0。

用法

my_mc:_visible

说明

属性：一个布尔值，它指示 *my_mc* 指定的影片剪辑是否可见。不可见的影片剪辑（_visible 属性设置为 false）处于禁用状态。例如，不能单击 _visible 设置为 false 的影片剪辑中的按钮。除非通过这种方式显式地使影片剪辑不可见，否则，影片剪辑将可见。

示例

当用户按 3 键时，以下代码禁用 *my_mc* 影片剪辑，并当用户按 4 键时启用该影片剪辑：

```
on(keyPress "3") {  
    my_mc:_visible = 0;  
}  
  
on(keyPress "4") {  
    my_mc:_visible = 1;  
}
```

_width

可用性

Flash Lite 1.0。

用法

my_mc:_width

说明

属性：影片剪辑的宽度，以像素为单位。

示例

当用户按 5 键时，以下示例会设置影片剪辑的宽度属性：

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

_x

可用性

Flash Lite 1.0。

用法

my_mc:_x

说明

属性：一个整数，它设置影片剪辑（此处表示为 *my_mc*）的 **x** 坐标，该坐标相对于父影片剪辑的本地坐标。如果影片剪辑位于主时间轴中，则其坐标系将舞台的左上角视为 (0, 0)。

如果影片剪辑位于另一个具有变形的影片剪辑中，则该影片剪辑位于包含它的影片剪辑的本地坐标系中。例如，如果将影片剪辑逆时针旋转 90 度，则其子级影片剪辑将继承逆时针旋转 90 度的坐标系。影片剪辑的坐标指的是注册点的位置。

示例

当用户按 6 键时，以下示例会更改 *my_mc* 影片剪辑的水平位置：

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

另请参见

[_xscale](#)、[_y](#)、[_yscale](#)

_xscale

可用性

Flash Lite 1.0。

用法

my_mc:_xscale

说明

属性：设置从影片剪辑注册点开始应用的该影片剪辑的水平缩放比例（百分比）。默认注册点为 (0, 0)。

缩放本地坐标系将影响 **_x** 和 **_y** 属性的设置，这两个设置是以像素为单位定义的。例如，如果父影片剪辑缩小到 50%，则设置 **_x** 属性时将移动该影片剪辑中的对象，移动距离为在影片设置为 100% 时其像素数的一半。

示例

当用户按 7 键时，以下示例会更改 my_mc 影片剪辑的水平缩放比例：

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

另请参见

[_x](#)、[_y](#)、[_yscale](#)

_y

可用性

Flash Lite 1.0。

用法

my_mc._y

说明

属性；一个整数，它设置影片剪辑（此处表示为 *my_mc*）的 y 坐标，该坐标相对于父影片剪辑的本地坐标。如果影片剪辑位于主时间轴中，则其坐标系将舞台的左上角视为 (0, 0)。

如果影片剪辑位于另一个具有变形的影片剪辑中，则该影片剪辑位于包含它的影片剪辑的本地坐标系中。例如，如果将影片剪辑逆时针旋转 90 度，则其子级影片剪辑将继承逆时针旋转 90 度的坐标系。影片剪辑的坐标指的是注册点的位置。

示例

当用户按 1 键时，以下代码会将 my_mc 影片剪辑的 y 坐标设置为其父剪辑的坐标 (0, 0) 以下 10 像素：

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

另请参见

[_x](#)、[_xscale](#)、[_yscale](#)

`_yscale`

可用性

Flash Lite 1.0。

用法

`my_mc._yscale`

说明

属性：设置从影片剪辑注册点开始应用的该影片剪辑的垂直缩放比例（百分比）。默认注册点为 (0, 0)。

缩放本地坐标系将影响 `_x` 和 `_y` 属性的设置，这两个设置是以像素为单位定义的。例如，如果父影片剪辑缩小到 50%，则设置 `_y` 属性时将移动影片剪辑中的对象，移动距离为在影片设置为 100% 时其像素数的一半。

示例

当用户按 1 键时，以下示例会将 `my_mc` 影片剪辑的垂直缩放比例更改为 10%：

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

另请参见

[_x](#)、[_xscale](#)、[_y](#)

本部分说明 Macromedia Flash Lite 1.x ActionScript 语句的语法和用法，这些语句是执行或指定动作的语言元素。下表概述了这些语句：

语句	说明
<code>break</code>	指示 Flash Lite 跳过循环体的其余部分，停止循环动作，并执行循环语句后面的语句。
<code>case</code>	定义用于 <code>switch</code> 语句的条件。如果 <code>case</code> 关键字后面的 <i>expression</i> 参数等于 <code>switch</code> 语句的 <i>expression</i> 参数，则执行 <i>statements</i> 参数中的语句。
<code>continue</code>	跳过最里层循环中所有其余的语句并开始循环的下一个迭代，就如同控制正常传递到了循环结尾。
<code>do..while</code>	执行语句，然后只要循环中的条件为 <code>true</code> ，就计算该条件。
<code>else</code>	指定当 <code>if</code> 语句中的条件计算结果为 <code>false</code> 时要运行的语句。
<code>else if</code>	计算某个条件，并指定当初始 <code>if</code> 语句中的该条件返回 <code>false</code> 值时要运行的语句。
<code>for</code>	计算一次 <i>init</i> （初始化）表达式，然后开始一个循环序列。在该循环序列中，只要 <i>condition</i> 计算结果为 <code>true</code> ，就执行 <i>statement</i> ，然后计算下一个表达式。
<code>if</code>	对条件进行计算以确定 SWF 文件中的下一步动作。如果条件为 <code>true</code> ，则 Flash Lite 将运行条件后面大括号 ({}) 内的语句。如果条件为 <code>false</code> ，则 Flash Lite 将跳过大括号内的语句，而运行大括号后面的语句。
<code>switch</code>	与 <code>if</code> 语句相似， <code>switch</code> 语句会测试某个条件，并在条件计算结果为 <code>true</code> 时执行语句。
<code>while</code>	测试一个表达式，只要该表达式为 <code>true</code> ，就会在循环中重复运行一条语句或一系列语句。

break

可用性

Flash Lite 1.0。

用法

break

参数

无。

说明

语句；出现在一个循环（for、do..while 或 while）中，或者出现在与 switch 语句内的特定 case 相关联的语句块中。break 语句可指示 **Flash Lite** 跳过循环体的其余部分，停止循环动作，并执行循环语句后面的语句。在使用 break 语句时，**ActionScript** 解释程序会跳过该 case 块中的其余语句，并跳到包含它的 switch 语句后的第一条语句。使用此语句可中断一系列嵌套的循环。

示例

以下示例使用 break 语句来退出一个循环（如果没有该语句，则该循环为无限循环）：

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

另请参见

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

case

可用性

Flash Lite 1.0。

用法

case *expression*: *statements*

参数

expression 任何表达式。

statements 任何语句。

说明

语句；定义用于 switch 语句的条件。如果 case 关键字后面的 *expression* 参数等于 switch 语句的 *expression* 参数，则执行 *statements* 参数中的语句。

如果在 switch 语句外部使用 case 语句，将产生错误，且不会编译这段代码。

示例

在以下示例中，如果 myNum 参数的计算结果为 1，则执行 case 1 后面的 `trace()` 语句；如果 myNum 参数的计算结果为 2，则执行 case 2 后面的 `trace()` 语句，依此类推。如果任何 case 表达式都不与 number 参数匹配，则执行 default 关键字后面的 `trace()` 语句。

```
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")  
}
```

在以下示例中，由于第一个 case 组中没有出现 `break`，因而，如果数字为 1，则会在“输出”面板中出现 A 和 B：

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

另请参见

[switch](#)

continue

可用性

Flash Lite 1.0。

用法

continue

参数

无。

说明

语句：跳过最内层循环中的所有剩余语句并开始下一个循环迭代，就如同已将控制正常传递到循环体的末尾。在循环外部时无作用。

- 在 while 循环中，continue 可使 **Flash** 解释程序跳过循环体的其余部分，并跳到循环体的顶端（在此进行条件测试）。
- 在 do..while 循环中，continue 可使 **Flash** 解释程序跳过循环体的其余部分，并跳到循环体的底端（在此进行条件测试）。
- 在 for 循环中，continue 可使 **Flash** 解释程序跳过循环体的其余部分，并转而计算 for 循环后的表达式。

示例

在下面的 while 循环中，continue 可使 **Flash Lite** 跳过循环体的其余部分，并跳到循环体的顶端（在此进行条件测试）：

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

在下面的 do..while 循环中，continue 可使 **Flash Lite** 跳过循环体的其余部分，并跳到循环体的底端（在此进行条件测试）：

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

```
    i++;  
} while (i < 10);
```

在 for 循环中，continue 可使 **Flash Lite** 跳过循环体的其余部分。在以下示例中，如果 i 模 3 等于 0，则会跳过 trace(i) 语句：

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

另请参见

[do..while](#)、[for](#)、[while](#)

do..while

可用性

Flash Lite 1.0。

用法

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

参数

statement 只要 *condition* 参数计算结果为 true 就执行的语句。

condition 要计算的条件。

说明

语句；只要循环中的条件为 true，就执行语句，然后计算该条件。

示例

以下示例中，只要索引变量的值小于 10，就会递增该变量：

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

另请参见

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

else

可用性

Flash Lite 1.0。

用法

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

参数

condition 计算结果为 true 或 false 的表达式。

t-statement 当条件的计算结果为 true 时要执行的指令。

f-statement 当条件的计算结果为 false 时要执行的替代指令系列。

说明

语句；指定当 if 语句中的条件计算结果为 false 时要运行的语句。

示例

以下示例说明如何将 else 语句与某一条件一起使用。实际的示例将包括根据该条件采取某一操作的代码。

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

另请参见

[if](#)

else if

可用性

Flash Lite 1.0。

用法

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

参数

condition 计算结果为 true 或 false 的表达式。

statement 当 if 语句中指定的条件为 false 时要运行的一系列语句。

说明

语句；计算条件，并指定当初始 if 语句中的条件返回 false 值时要运行的语句。如果 else if 条件返回 true 值，则 **Flash** 解释程序运行 else if 条件后面大括号 ({}) 中的语句。如果 else if 条件为 false，则 **Flash** 将跳过大括号中的语句，而运行大括号后面的语句。使用 else if 语句可在脚本中创建分支逻辑。

示例

以下示例使用 else if 语句检查对象的每一边是否都在特定的边界内：

```
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");  
}
```

另请参见

[if](#)

for

可用性

Flash Lite 1.0。

用法

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

参数

init 一个在开始循环序列前要计算的表达式，通常为赋值表达式。

condition 计算结果为 true 或 false 的表达式。在每次循环迭代前计算该条件；当条件的计算结果为 false 时退出循环。

next 在每次循环迭代后要计算的表达式；通常为使用递增 (++) 或递减 (--) 运算符的赋值表达式。

statement 要在循环中执行的一个或多个指令。

说明

语句：一种循环构造，它首先计算一次 *init*（初始化）表达式，然后开始一个循环序列。在该循环序列中，只要 *condition* 的计算结果为 true，就执行 *statement*，然后计算下一个表达式。

某些属性无法用 for 或 for..in 语句进行枚举。例如，影片剪辑属性（如 _x 和 _y）就不能枚举。

示例

以下示例使用 for 循环对从 1 到 100 的数字求和：

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

另请参见

[++（递增）](#)、[--（递减）](#)、[do..while](#)、[while](#)

if

可用性

Flash Lite 1.0。

用法

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

参数

condition 计算结果为 true 或 false 的表达式。

statement 当条件的计算结果为 true 时要执行的指令。

说明

语句；对条件进行计算以确定 SWF 文件中的下一步动作。如果条件为 true，则 **Flash Lite** 将运行条件后面大括号 ({}) 内的语句。如果条件为 false，则 **Flash Lite** 将跳过大括号内的语句，而运行大括号后面的语句。使用 if 语句可在脚本中创建分支逻辑。

示例

在以下示例中，括号内的条件对变量 name 进行计算，以查看其是否具有文本值 "Erica"。如果有，则运行 play() 函数。

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

switch

可用性

Flash Lite 1.0。

用法

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

参数

expression 任何数值表达式。

caseClause 一个 **case** 关键字，其后跟有一个表达式、一个冒号和一组语句，如果表达式与 **switch** 的 *expression* 参数相匹配，则执行这组语句。

defaultClause 一个可选的 **default** 关键字，其后跟有一组语句。如果任何一个 **case** 表达式都不与 **switch** 的 *expression* 参数相匹配，则执行这组语句。

说明

语句；创建 **ActionScript** 语句的分支结构。与 **if** 语句相似，**switch** 语句会测试某个条件，并在条件计算结果为 **true** 时执行语句。

Switch 语句包含一个名为 **default** 的代用条件选项。如果其它语句都不为 **true**，则会执行 **default** 语句。

示例

在以下示例中，如果 **myNum** 参数的计算结果为 **1**，则执行 **case 1** 后面的 **trace()** 语句；如果 **myNum** 参数的计算结果为 **2**，则执行 **case 2** 后面的 **trace()** 语句，依此类推。如果任何 **case** 表达式都不与 **number** 参数匹配，则执行 **default** 关键字后面的 **trace()** 语句。

```
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")
}
```

在以下示例中，第一个 **case** 组不包含 **break**。因此，如果数字为 **1**，则会在“输出”面板中显示 **A** 和 **B**：

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

另请参见

[case](#)

while

可用性

Flash Lite 1.0。

用法

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

参数

condition 每次执行 while 语句时都要计算的表达式。

statement 当条件的计算结果为 true 时要执行的指令。

说明

语句；测试一个表达式，只要该表达式为 true，就会在循环中重复运行一条语句或一系列语句。

在运行该语句块之前，首先测试条件；如果测试返回 true，则运行该语句块。如果该条件为 false，则跳过该语句块，并执行 while 语句所在语句块之后的第一条语句。

通常，当计数器变量小于某指定值时，使用循环来执行动作。在每个循环的结尾递增计数器的值，直到达到指定值为止。此时，条件不再为 true，因此循环结束。

while 语句执行下面一系列步骤。步骤 1 至步骤 4 的每次重复，称作循环的一次“迭代”。在每次迭代开始时，都会对条件进行测试：

1. 计算表达式 *condition*。
2. 如果 *condition* 的计算结果为 true，或者是一个可转换为布尔值 true 的值（如非零数字），则转到步骤 3。
否则，while 语句结束，并从 while 循环之后的下一语句继续执行。
3. 运行语句块 *statement*。
4. 转到步骤 1。

示例

只要索引变量 *i* 的值小于 10，以下示例就会执行循环：

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

另请参见

[continue](#)、[do..while](#)、[for](#)

本部分说明 Macromedia Flash Lite 1.x ActionScript 运算符的语法和用法。所有条目均按字母顺序列出。但某些运算符是符号，它们按照其文本说明的字母顺序列出。

下表中总结了本部分中的运算符：

运算符	说明
<code>add</code> （字符串连接）	运算符；连接（合并）两个或多个字符串。
<code>+=</code> （加法赋值）	将 <i>expression1</i> + <i>expression2</i> 的值赋予 <i>expression1</i> 。
<code>and</code>	执行逻辑 AND 运算。
<code>=</code> （赋值）	将 <i>expression2</i> （右侧的操作数）的值赋予 <i>expression1</i> 中的变量或属性。
<code>/*</code> （块注释）	表示一行或多行脚本注释。出现在注释开始标签 (<i>/*</i>) 和注释结束标签 (<i>*/</i>) 之间的任何字符都被 ActionScript 解释程序解释为注释并忽略。
<code>,</code> （逗号）	计算 <i>expression1</i> ，然后计算 <i>expression2</i> ，并返回 <i>expression2</i> 的值。
<code>//</code> （注释）	表示脚本注释的开始。任何出现在注释分隔符 (<i>//</i>) 和行结束符之间的字符都被 ActionScript 解释程序解释为注释并忽略。
<code>?:</code> （条件）	指示 Flash Lite 计算 <i>expression1</i> ，如果它的值为 true，则该运算符返回 <i>expression2</i> 的值；否则，返回 <i>expression3</i> 的值。
<code>--</code> （递减）	从 <i>expression</i> 中减去 1。此运算符的预先递减形式 (<i>--expression</i>) 从 <i>expression</i> 中减去 1，然后返回数字结果。此运算符的滞后递减形式 (<i>expression--</i>) 从 <i>expression</i> 中减去 1，然后返回 <i>expression</i> 的初始值（减去 1 之前的值）。
<code>/</code> （除法）	将 <i>expression1</i> 除以 <i>expression2</i> 。
<code>/=</code> （除法赋值）	将 <i>expression1</i> / <i>expression2</i> 的值赋予 <i>expression1</i> 。
<code>.</code> （点）	用于定位影片剪辑的层次结构，以便访问嵌套的（子级）影片剪辑、变量或属性。
<code>++</code> （递增）	将 <i>expression</i> 加 1。 <i>expression</i> 可以是变量、数组中的元素或对象的属性。此运算符的预先递增形式 (<i>++expression</i>) 将 <i>expression</i> 加 1，然后返回数字结果。此运算符的预先递增形式 (<i>expression++</i>) 将 <i>expression</i> 加 1，然后返回 <i>expression</i> 的初始值（加 1 之前的值）。

运算符	说明
&& (逻辑 AND)	计算 <i>expression1</i> (该运算符左侧的表达式), 当此表达式的计算结果为 false 时返回 false。如果 <i>expression1</i> 的计算结果为 true, 则计算 <i>expression2</i> (运算符右侧的表达式)。如果 <i>expression2</i> 的计算结果为 true, 则最终结果为 true; 否则, 最终结果为 false。
! (逻辑 NOT)	对变量或表达式的布尔值取反。如果 <i>expression</i> 是绝对值或转换值为 true 的变量, 则 ! <i>expression</i> 的值为 false。如果表达式 <i>x</i> && <i>y</i> 的计算结果为 false, 则表达式 !(<i>x</i> && <i>y</i>) 的计算结果为 true。
(逻辑 OR)	计算 <i>expression1</i> 和 <i>expression2</i> 。如果其中任何一个或者两个表达式的计算结果为 true, 则结果为 true; 只有当两个表达式的计算结果都为 false 时, 结果才为 false。逻辑 OR 运算符可与任意多个操作数一起使用; 只要任意一个操作数的计算结果为 true, 结果就为 true。
% (模)	计算 <i>expression1</i> 除以 <i>expression2</i> 的余数。如果 <i>expression</i> 操作数为非数字, 则模运算符会尝试将其转换为数字。
%= (模赋值)	将 <i>expression1</i> % <i>expression2</i> 的值赋予 <i>expression1</i> 。
*= (乘法赋值)	将 <i>expression1</i> * <i>expression2</i> 的值赋予 <i>expression1</i> 。
* (乘法)	将两个数字表达式相乘。
+ (数字加法)	将数字表达式相加。
== (数值等于)	测试是否相等; 如果 <i>expression1</i> 等于 <i>expression2</i> , 则结果为 true。
> (数值大于)	比较两个表达式, 并确定 <i>expression1</i> 是否大于 <i>expression2</i> ; 如果是, 则该运算符返回 true。如果 <i>expression1</i> 小于或等于 <i>expression2</i> , 则该运算符返回 false。
>= (数值大于或等于)	比较两个表达式, 并确定 <i>expression1</i> 是大于等于 <i>expression2</i> (true) 还是小于 <i>expression2</i> (false)。
<> (数值不等于)	测试是否不相等; 如果 <i>expression1</i> 等于 <i>expression2</i> , 则结果为 false。
< (数值小于)	比较两个表达式, 并确定 <i>expression1</i> 是否小于 <i>expression2</i> ; 如果是, 则该运算符返回 true。如果 <i>expression1</i> 大于或等于 <i>expression2</i> , 则该运算符返回 false。
<= (数值小于或等于)	比较两个表达式, 并确定 <i>expression1</i> 是否小于或等于 <i>expression2</i> 。如果是, 则该运算符返回 true; 否则, 该运算符返回 false。
() (括号)	将一个或多个参数分组, 执行表达式的按顺序计算, 或将一个或多个参数用括号括起来, 并将它们作为参数传递给括号外的函数。
" " (字符串分隔符)	用于一系列零个或多个字符之前和之后时, 引号表示这些字符具有文本值, 并被视为“字符串”; 它们不是变量、数值或其它 ActionScript 元素。
eq (字符串相等)	比较两个表达式是否相等, 如果 <i>expression1</i> 的字符串表示形式等于 <i>expression2</i> 的字符串表示形式, 则返回 true; 否则, 该运算符返回 false。

运算符	说明
gt （字符串大于）	将 <i>expression1</i> 的字符串表示形式与 <i>expression2</i> 的字符串表示形式进行比较，如果 <i>expression1</i> 大于 <i>expression2</i> ，则返回 true；否则，返回 false。
ge （字符串大于或等于）	将 <i>expression1</i> 的字符串表示形式与 <i>expression2</i> 的字符串表示形式进行比较，如果 <i>expression1</i> 大于或等于 <i>expression2</i> ，则返回值为 true；否则，返回值为 false。
ne （字符串不等于）	将 <i>expression1</i> 的字符串表示形式与 <i>expression2</i> 的字符串表示形式进行比较，如果 <i>expression1</i> 不等于 <i>expression2</i> ，则返回 true；否则，返回 false。
lt （字符串小于）	将 <i>expression1</i> 的字符串表示形式与 <i>expression2</i> 的字符串表示形式进行比较，如果 <i>expression1</i> 小于 <i>expression2</i> ，则返回值为 true；否则，返回值为 false。
le （字符串小于或等于）	将 <i>expression1</i> 的字符串表示形式与 <i>expression2</i> 的字符串表示形式进行比较，如果 <i>expression1</i> 小于或等于 <i>expression2</i> ，则返回值为 true；否则，返回值为 false。
- （减法）	用于取反或减法。
-= （减法赋值）	将 <i>expression1</i> - <i>expression2</i> 的值赋予 <i>expression1</i> 。

add （字符串连接）

可用性

Flash Lite 1.0。

用法

string1 add *string2*

操作数

string1、*string2* 字符串。

说明

运算符；连接（合并）两个或多个字符串。

示例

以下示例合并两个字符串值以生成字符串 **catalog**。

```
conStr = "cat" add "alog";
trace (conStr);// 输出: catalog
```

另请参见

[+（数字加法）](#)

+=（加法赋值）

可用性

Flash Lite 1.0。

用法

expression1 += *expression2*

操作数

expression1、*expression2* 数字或字符串。

说明

运算符（算术组合赋值）；将 *expression1* + *expression2* 的值赋予 *expression1*。例如，以下两个语句的结果是相同的：

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

加法 (+) 运算符的所有规则适用于加法赋值 (+=) 运算符。

示例

以下示例使用加法赋值 (+=) 运算符将 x 的值增加 y 值：

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

另请参见

[+（数字加法）](#)

and

可用性

Flash Lite 1.0。

用法

condition1 and *condition2*

操作数

condition1、*condition2* 计算结果为 true 或 false 的条件或表达式。

说明

运算符；执行逻辑 AND 运算。

示例

以下示例使用 and 运算符测试游戏者是否在游戏中获胜。在游戏过程中，当轮到游戏者玩或者当游戏者得到计分点时，就会对 turns 和 score 变量进行更新。在 3 轮之内游戏者的得分达到或超过 75 时，下面的脚本就会在“输出”面板中显示“You Win the Game!”（您获得了游戏的胜利！）。

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

另请参见

[&&（逻辑 AND）](#)

=（赋值）

可用性

Flash Lite 1.0。

用法

expression1 = *expression2*

操作数

expression1 一个变量或属性。

expression2 一个值。

说明

运算符；将 *expression2*（位于右侧的操作数）的值赋予 *expression1* 中的变量或属性。

示例

以下示例使用赋值 (=) 运算符将一个数值赋予变量 `weight`：

```
weight = 5;
```

以下示例使用赋值 (=) 运算符将一个字符串值赋予变量 `greeting`：

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

/* (块注释)

可用性

Flash Lite 1.0

用法

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

操作数

`comment` 任何字符。

说明

注释分隔符：表示一行或多行脚本注释。出现在注释开始标签 (`/*`) 和注释结束标签 (`*/`) 之间的任何字符都被 **ActionScript** 解释程序解释为注释并忽略。

使用 `//`（注释分隔符）可以标识单行注释。使用 `/*` 注释分隔符可以在多个连续行中标识注释。在使用注释分隔符的这种格式时，如果不使用结束标签 (`*/`)，就会返回一条错误消息。尝试嵌套注释也会返回错误消息。

使用了注释开始标签 (`/*`) 后，无论在注释开始标签和注释结束标签之间有多少个开始标签 (`/*`)，第一个注释结束标签 (`*/`) 都会结束注释。

另请参见

[// \(注释\)](#)

, (逗号)

可用性

Flash Lite 1.0。

用法

expression1, *expression2*

操作数

expression1、*expression2* 数字或计算结果为数字的表达式。

说明

运算符：先计算 *expression1*，再计算 *expression2*，然后返回 *expression2* 的值。

示例

以下示例使用不带括号 () 运算符的逗号 (,) 运算符，并阐释了逗号运算符只返回第一个不带括号 () 运算符的表达式值：

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

以下示例使用带括号 () 运算符的逗号 (,) 运算符，并阐释了在与括号 () 运算符同时使用时，逗号运算符会返回最后一个表达式的值：

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

以下示例使用不带括号 () 运算符的逗号 (,) 运算符，并阐释了逗号运算符会按顺序计算所有表达式，但只返回第一个表达式的值。计算第二个表达式 *z++*，且 *z* 递增 1。

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

以下示例除了添加括号 () 运算符以外，与前一个示例完全相同，并再次阐释了与括号 () 运算符同时使用时，逗号 (,) 运算符会返回系列中最后一个表达式的值：

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

另请参见

[for、\(\) \(括号\)](#)

// (注释)

可用性

Flash Lite 1.0

用法

// 注释

操作数

comment 任何字符。

说明

注释分隔符；表示脚本注释的开始。任何出现在注释分隔符 (//) 和行结束符之间的字符都被 **ActionScript** 解释程序解释为注释并忽略。

示例

以下示例使用注释分隔符将第一、第三、第五和第七行标识为注释：

```
// 记录 ball 影片剪辑的 X 位置。  
ballX = ball._x;  
// 记录 ball 影片剪辑的 Y 位置。  
ballY = ball._y;  
// 记录 bat 影片剪辑的 X 位置。  
batX = bat._x;  
// 记录 bat 影片剪辑的 Y 位置。  
batY = bat._y;
```

另请参见

[/* \(块注释\)](#)

?: (条件)

可用性

Flash Lite 1.0。

用法

expression1 ? *expression2* : *expression3*

操作数

expression1 计算结果为布尔值的表达式，通常为像 *x* < 5 这样的比较表达式。

expression2、*expression3* 任何类型的值。

说明

运算符；指示 **Flash Lite** 计算 *expression1*，如果它的值为 *true*，则返回 *expression2* 的值；否则，返回 *expression3* 的值。

示例

以下示例因为 *expression1* 的计算结果为 *true*，所以将变量 *x* 的值赋予变量 *z*：

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

-- (递减)

可用性

Flash Lite 1.0。

用法

行 *expression*

expression *行*

操作数

无。

说明

运算符（算术）；从 *expression* 中减去 1 的预先递减和滞后递减的一元运算符。此运算符的预先递减形式 (*--expression*) 从 *expression* 中减去 1，然后返回数字结果。此运算符的滞后递减形式 (*expression--*) 从 *expression* 中减去 1，然后返回 *expression* 的初始值（减去 1 之前的值）。

示例

以下示例显示该运算符的预先递减形式，它将 `aWidth` 递减为 2 (`aWidth - 1 = 2`)，并将结果返回为 `bWidth`：

```
aWidth = 3;  
bWidth = --aWidth;  
// bWidth 值等于 2。
```

下一个示例显示此运算符的滞后递减形式，它将 `aWidth` 递减为 2 (`aWidth - 1 = 2`)，并将 `aWidth` 的初始值返回为结果 `bWidth`：

```
aWidth = 3;  
bWidth = aWidth--;  
// bWidth 值等于 3。
```

/（除法）

可用性

Flash Lite 1.0。

用法

expression1 / expression2

操作数

expression1、*expression2* 数字或计算结果为数字的表达式。

说明

运算符（算术）：将 *expression1* 除以 *expression2*。除法运算的结果为双精度浮点数。

示例

以下语句将浮点数 22.0 除以 7.0，然后在“输出”面板中显示结果：

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

结果为 3.1429，是一个浮点数。

/= （除法赋值）

可用性

Flash Lite 1.0。

用法

expression1 /= expression2

操作数

expression1、*expression2* 数字或计算结果为数字的表达式。

说明

运算符（算术组合赋值）；将 *expression1 / expression2* 的值赋予 *expression1*。例如，下面两个语句是等效的：

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

示例

以下示例将 /= 运算符与变量和数字结合使用：

```
x = 10;  
y = 2;  
x /= y;  
// 表达式 x 现在包含值 5。
```

. （点）

可用性

Flash Lite 1.0。

用法

instancename.variable

instancename.childinstance.variable

操作数

instancename 影片剪辑的实例名称。

childinstance 从属于或嵌套于另一个影片剪辑的影片剪辑实例。

variable 所指定影片剪辑实例名称的时间轴上的变量。

说明

运算符；用于定位影片剪辑的层次结构，以便访问嵌套的（子级）影片剪辑、变量或属性。

示例

以下示例标识影片剪辑 `person_mc` 中变量 `hairColor` 的当前值：

```
person_mc.hairColor
```

它与下面的斜杠标记语法等效：

```
/person_mc:hairColor
```

另请参见

[/（正斜杠）](#)

++（递增）

可用性

Flash Lite 1.0。

用法

```
++expression
```

```
expression++
```

操作数

无。

说明

运算符（算术）：将 *expression* 加 1 的预先递增和预先递增的一元运算符。**expression** 可以是变量、数组中的元素或对象的属性。此运算符的预先递增形式 (`++expression`) 将 *expression* 加 1，然后返回数字结果。此运算符的预先递增形式 (`expression++`) 将 *expression* 加 1，然后返回 *expression* 的初始值（加 1 之前的值）。

示例

以下示例将 `++` 用作预先递增运算符，以使 `while` 循环运行 5 次：

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

以下示例将 `++` 用作预先递增运算符：

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


此脚本在“输出”面板中显示以下结果：

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

以下示例将 ++ 用作预先递增运算符：

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

此脚本在“输出”面板中显示以下结果：

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

&&（逻辑 AND）

可用性

Flash Lite 1.0。

用法

expression1 && *expression2*

操作数

expression1、*expression2* 布尔值或转换为布尔值的表达式。

说明

运算符（逻辑）：对一个或两个表达式的值执行布尔运算。该运算符计算 *expression1*（运算符左侧的表达式），当此表达式的计算结果为 false 时返回 false。如果 *expression1* 的计算结果为 true，则计算 *expression2*（运算符右侧的表达式）。如果 *expression2* 的计算结果为 true，则最终结果为 true；否则，最终结果为 false。

示例

以下示例使用 && 运算符执行一个测试，以确定游戏者是否已经在游戏中获胜。在游戏过程中，当轮到游戏者玩或者当游戏者得到积分点时，就会对 turns 和 score 变量进行更新。在 3 轮之内游戏者的得分达到或超过 75 时，下面的脚本就会在“输出”面板中显示“**You Win the Game!**”（您获得了游戏的胜利！）。

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

```
        trace("Try Again!");  
    }
```

以下示例演示测试过程，以查看假想的 x 位置是否在某个范围之内：

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

！（逻辑 NOT）

可用性

Flash Lite 1.0。

用法

!expression

操作数

无。

说明

运算符（逻辑）；对变量或表达式的布尔值取反。如果 *expression* 是绝对值或转换值为 true 的变量，则 *!expression* 的值为 false。如果表达式 *x && y* 的计算结果为 false，则表达式 *!(x && y)* 的计算结果为 true。

以下表达式显示使用！运算符的结果：

!true 返回 false

!false 返回 true

示例

在以下示例中，将变量 happy 设置为 false。if 条件对 !happy 条件进行计算，如果该条件为 true，则 `trace()` 函数向“输出”面板发送一个字符串。

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

||（逻辑 OR）

可用性

Flash Lite 1.0。

用法

expression1 || *expression2*

操作数

expression1、*expression2* 布尔值或转换为布尔值的表达式。

说明

运算符（逻辑）：计算 *expression1* 和 *expression2*。如果其中任何一个或者两个表达式的计算结果为 true，则结果为 true；只有当两个表达式的计算结果都为 false 时，结果才为 false。逻辑 OR 运算符可与任意多个操作数一起使用；只要任意一个操作数的计算结果为 true，结果就为 true。

对于非布尔表达式，逻辑 OR 运算符会使 Flash Lite 对左侧的表达式进行计算：如果左侧的表达式可以转换为 true，则结果为 true。否则，计算右侧的表达式，而且结果就是该表达式的值。

示例

用法 1：以下示例在 if 语句中使用 || 运算符。第二个表达式的计算结果为 true，因此最终结果为 true：

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

%（模）

Flash Lite 1.0。

用法

expression1 % expression2

操作数

expression1、*expression2* 数字或计算结果为数字的表达式。

说明

运算符（算术）：计算 *expression1* 除以 *expression2* 的余数。如果 *expression* 操作数为非数字，则模运算符会尝试将其转换为数字。**expression** 可以是数字或转换为数值的字符串。

在以 Flash Lite 1.0 或 1.1 为目标时，Flash 编译器会通过使用以下公式将 % 运算符扩展到已发布的 SWF 文件中：

expression1 - int(*expression1*/*expression2*) * *expression2*

这种近似计算可能没有本机支持模运算符的 Flash Player 版本那样高效或精确。

示例

以下代码显示使用模 (%) 运算符的数字示例：

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

%=（模赋值）

可用性

Flash Lite 1.0。

用法

expression1 %= expression2

操作数

expression1、*expression2* 数字或计算结果为数字的表达式。

说明

运算符（算术复合赋值）：将 *expression1 % expression2* 的值赋予 *expression1*。例如，以下两个表达式是等效的：

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

示例

以下示例将值 4 赋予变量 x:

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

另请参见

[% \(模\)](#)

*= (乘法赋值)

可用性

Flash Lite 1.0。

用法

expression1 *= *expression2*

操作数

expression1、*expression2* 数字或计算结果为数字的表达式。

说明

运算符（算术组合赋值）：将 *expression1* * *expression2* 的值赋予 *expression1*。

例如，以下两个表达式是相同的：

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

示例

用法 1：以下示例将值 50 赋予变量 x：

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

用法 2：以下示例的第二行和第三行计算等号 (=) 右侧的表达式，然后将结果赋予 x 和 y：

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

* （乘法）

可用性

Flash Lite 1.0。

用法

expression1 * *expression2*

操作数

expression1、*expression2* 数字表达式。

说明

运算符（算术）：将两个数字表达式相乘。如果两个表达式都是整数，则积为整数。如果其中任何一个或两个表达式是浮点数，则积为浮点数。

示例

用法 1：以下语句将整数 2 与 3 相乘。

2 * 3

结果为 6，是一个整数。

用法 2：以下语句将浮点数 2.0 与 3.1416 相乘：

2.0 * 3.1416

结果为 6.2832，是一个浮点数。

+ （数字加法）

可用性

Flash Lite 1.0。

用法

expression1 + *expression2*

操作数

expression1、*expression2* 数字。

说明

运算符：将数字表达式相加。+ 只是数值运算符；它不能用于字符串连接。

两个表达式都为整数时，和为整数；其中一个或两个表达式为浮点数时，和为浮点数。

示例

以下示例将整数 2 和 3 相加；结果为整数 5，显示在“输出”面板中：

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

以下示例将浮点数 2.5 和 3.25 相加；结果为浮点数 5.75，显示在“输出”面板中：

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

另请参见

[add（字符串连接）](#)

==（数值等于）

可用性

Flash Lite 1.0。

用法

```
expression1 == expression2
```

操作数

expression1、*expression2* 数字、布尔值或变量。

说明

运算符（比较）；测试是否相等；它与 <> 运算符正好相反。如果 *expression1* 等于 *expression2*，则结果为 true。与 <> 运算符一样，“相等”的定义取决于所比较的数据类型：

- 数字和布尔值按值进行比较。
- 变量按引用进行比较。

示例

以下示例显示返回值 true 和 false：

```
trees = 7;
```

```
bushes = "7";
```

```
shrubs = "seven";
```

```
trace (trees == "7");// 输出: 1(true)
```

```
trace (trees == bushes);// 输出: 1(true)
```

```
trace (trees == shrubs);// 输出: 0(false)
```

另请参见

[eq（字符串相等）](#)

> （数值大于）

可用性

Flash Lite 1.0。

用法

expression1 > *expression2*

操作数

expression1、*expression2* 数字或计算结果为数字的表达式。

运算符（比较）：比较两个表达式，并确定 *expression1* 是否大于 *expression2*；如果是，则该运算符返回 true。如果 *expression1* 小于或等于 *expression2*，则该运算符返回 false。

示例

以下示例显示数值比较的结果 true 和 false：

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

另请参见

[gt](#)（字符串大于）

>= （数值大于或等于）

可用性

Flash Lite 1.0。

用法

expression1 >= *expression2*

操作数

expression1、*expression2* 整数或浮点数。

说明

运算符（比较）：比较两个表达式，并确定 *expression1* 是大于或等于 *expression2* (true)，还是 *expression1* 小于 *expression2* (false)。

示例

以下示例显示结果 true 和 false:

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

另请参见

[ge](#) (字符串大于或等于)

<> (数值不等于)

可用性

Flash Lite 1.0。

用法

expression1 <> *expression2*

操作数

expression1、*expression2* 数字、布尔值或变量。

说明

运算符 (比较): 测试是否不等于, 它与等于 == 运算符正好相反。如果 *expression1* 等于 *expression2*, 则结果为 false。与等于 (==) 运算符一样, 相等的定义取决于所比较的数据类型:

- 数字和布尔值按值进行比较。
- 变量按引用进行比较。

示例

以下示例显示返回值 true 和 false:

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

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

另请参见

[ne](#) (字符串不等于)

<（数值小于）

可用性

Flash Lite 1.0。

用法

expression1 < *expression2*

操作数

expression1、*expression2* 数字。

说明

运算符（比较）；比较两个表达式，并确定 *expression1* 是否小于 *expression2*；如果是，则该运算符返回 true。如果 *expression1* 大于或等于 *expression2*，则该运算符返回 false。<（小于）运算符是一个数值运算符。

示例

以下示例显示数值比较和字符串比较的结果 true 和 false：

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

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

另请参见

[lt（字符串小于）](#)

<=（数值小于或等于）

Flash Lite 1.0。

用法

expression1 <= *expression2*

操作数

expression1、*expression2* 数字。

说明

运算符（比较）；比较两个表达式，并确定 *expression1* 是否小于或等于 *expression2*。如果是，则该运算符返回 true；否则，该运算符返回 false。此运算符仅用于数值比较。

示例

以下示例显示数值比较的结果 true 和 false:

```
trace(5 <= 10); // 输出: 1(true)
trace(2 <= 2); // 输出: 1(true)
trace (10 <= 3); // 输出: 0 (false)
```

另请参见

[le](#) (字符串小于或等于)

() (括号)

可用性

Flash Lite 1.0。

用法

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

expression1、*expression2* 数字、字符串、变量或文本。

parameter1、...、*parameterN* 一系列参数，在将其结果作为参数传递给括号外的函数之前执行这些参数。

说明

运算符：对一个或多个参数分组，执行表达式的按顺序计算，或者括住一个或多个参数并将它们作为参数传递给括号外的函数。

用法 1：控制表达式中运算符的执行顺序。括号覆盖正常的优先级顺序，从而导致先计算括号内的表达式。如果括号是嵌套的，则先计算最里面括号中的内容，然后计算较靠外括号中的内容。

用法 2：按顺序计算用逗号分隔的一系列表达式，并返回最终表达式的结果。

示例

用法 1：以下语句显示使用括号来控制表达式的执行顺序（每个表达式的值都显示在“输出”面板中）：

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

用法 1：以下示例显示使用括号来控制表达式的执行顺序（每个表达式的值都会写入到日志文件中）：

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

" "（字符串分隔符）

可用性

Flash Lite 1.0。

用法

`"text"`

操作数

text 零个或多个字符。

说明

字符串分隔符：在零个或多个字符的前后使用字符串分隔符之后，引号指示这些字符具有其文本值，并将被视作一个“字符串”，而不是一个变量、数值或其它 **ActionScript** 元素。

示例

以下示例使用引号来指示变量 `yourGuess` 的值是文本字符串 "Prince Edward Island"，而不是变量名。`province` 的值是一个变量，而不是文本；若要确定 `province` 的值，必须找到 `yourGuess` 的值。

```
yourGuess = "Prince Edward Island";

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

eq（字符串相等）

可用性

Flash Lite 1.0。

用法

expression1 eq *expression2*

操作数

expression1、*expression2* 数字、字符串或变量。

说明

比较运算符：比较两个表达式是否相等。如果 *expression1* 的字符串表示形式与 *expression2* 的字符串表示形式相等，则返回 true；否则，该运算返回 false。

示例

以下示例显示结果 true 和 false：

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

另请参见

[==（数值等于）](#)

gt（字符串大于）

可用性

Flash Lite 1.0。

用法

expression1 gt *expression2*

操作数

expression1、*expression2* 数字、字符串或变量。

说明

运算符（比较）：比较 *expression1* 的字符串表示形式和 *expression2* 的字符串表示形式。如果 *expression1* 大于 *expression2*，则返回值为 true；否则，返回值为 false。字符串按字母顺序进行比较；数字优先于所有字母，所有大写字母优先于小写字母。

示例

以下示例显示结果 true 和 false:

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

另请参见

> (数值大于)

ge (字符串大于或等于)

可用性

Flash Lite 1.0。

用法

expression1 ge *expression2*

操作数

expression1、*expression2* 数字、字符串或变量。

说明

运算符(比较): 比较 *expression1* 的字符串表示形式和 *expression2* 的字符串表示形式。如果 *expression1* 大于或等于 *expression2*, 则返回值为 true; 否则, 返回值为 false。字符串按字母顺序进行比较; 数字优先于所有字母, 所有大写字母优先于小写字母。

示例

以下示例显示结果 true 和 false:

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

另请参见

>= (数值大于或等于)

ne（字符串不等于）

可用性

Flash Lite 1.0。

用法

expression1 ne *expression2*

操作数

expression1、*expression2* 数字、字符串或变量。

说明

运算符（比较）：比较 *expression1* 的字符串表示形式和 *expression2* 的字符串表示形式。如果 *expression1* 不等于 *expression2*，则返回 true；否则，返回 false。

示例

以下示例显示结果 true 和 false：

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

另请参见

[<>（数值不等于）](#)

lt（字符串小于）

可用性

Flash Lite 1.0。

用法

expression1 lt *expression2*

操作数

expression1、*expression2* 数字、字符串或变量。

说明

运算符（比较）；比较 *expression1* 的字符串表示形式和 *expression2* 的字符串表示形式。如果 *expression1* 小于 *expression2*，则返回值为 `true`；否则，返回值为 `false`。字符串按字母顺序进行比较；数字优先于所有字母，所有大写字母优先于小写字母。

示例

以下示例显示各种字符串比较的输出结果。请注意，在最后一行中，当比较字符串和整数时，`lt` 不会返回错误，这是因为 **ActionScript 1.0** 语法会尝试将整数数据类型转换为字符串并返回 `false`。

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

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

另请参见

[<（数值小于）](#)

le（字符串小于或等于）

可用性

Flash Lite 1.0。

用法

expression1 le *expression2*

操作数

expression1、*expression2* 数字、字符串或变量。

说明

运算符（比较）；比较 *expression1* 的字符串表示形式和 *expression2* 的字符串表示形式。如果 *expression1* 小于或等于 *expression2*，则返回值为 `true`；否则，返回值为 `false`。字符串按字母顺序进行比较；数字优先于所有字母，所有大写字母优先于小写字母。

示例

以下示例显示各种字符串比较的输出结果：

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

另请参见

[<=（数值小于或等于）](#)

-（减法）

可用性

Flash Lite 1.0。

用法

（符号反转） *-expression*

（减法） *expression1 - expression2*

操作数

expression1、*expression2* 数字或计算结果为数字的表达式。

说明

运算符（算术）；用于符号反转或减法运算。

用法 1：用于符号反转时，它将数值表达式的符号反转。

用法 2：用于减法时，它对两个数值表达式执行算术减法运算，从 *expression1* 中减去 *expression2*。两个表达式都为整数时，差为整数。其中任何一个或两个表达式为浮点数时，差为浮点数。

示例

用法 1：以下语句将表达式 `2 + 3` 的符号反转。

```
trace(-(2 + 3));  
// 输出: -5.
```

用法 2：下面的语句从整数 5 中减去整数 2。

```
trace(5 - 2);  
// 输出: 3.
```

结果为一个整数 3。

用法 3: 以下语句从浮点数 3.25 中减去浮点数 1.5:

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

结果为浮点数 1.75。

-= （减法赋值）

可用性

Flash Lite 1.0。

用法

```
expression1 -= expression2
```

操作数

expression1、*expression2* 数字或计算结果为数字的表达式。

说明

运算符（算术组合赋值）；将 *expression1* - *expression2* 的值赋予 *expression1*。不返回值。

例如，以下两个语句是相同的：

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

字符串表达式必须转换为数字，否则会返回 -1。

示例

用法 1: 以下示例使用 -= 运算符从 2 中减去 3，然后将结果赋予变量 x：

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

用法 2: 以下示例显示如何将字符串转换为数字：

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

本部分介绍 **Macromedia Flash Lite 1.1** 能够识别的平台功能和变量，以及可以使用 `fscommand()` 和 `fscommand2()` 函数执行的 **Flash Lite** 命令。本部分所说明的功能特定于 **Flash Lite**。

下表概括了本部分的内容：

语言元素	说明
<code>_capCompoundSound</code>	表明 Flash Lite 是否能处理复合声音。
<code>_capEmail</code>	表明 Flash Lite 客户端是否能使用 <code>GetURL()</code> ActionScript 命令发送电子邮件。
<code>_capLoadData</code>	表明主机应用程序是否能通过调用 <code>loadMovie()</code> 、 <code>loadMovieNum()</code> 、 <code>loadVariables()</code> 和 <code>loadVariablesNum()</code> 函数动态加载其它数据。
<code>_capMFi</code>	表明设备是否能播放 i-mode (MFi) 音频格式的声音数据。
<code>_capMIDI</code>	表明设备是否能播放乐器数字接口 (MIDI) 音频格式的声音数据。
<code>_capMMS</code>	表明 Flash Lite 是否能通过使用 <code>GetURL()</code> ActionScript 命令发送多媒体消息服务 (MMS) 消息。
<code>_capMP3</code>	表明设备是否能播放 MPEG 音频层 3 (MP3) 音频格式的声音数据。
<code>_capSMAF</code>	表明设备是否能播放合成音乐移动应用格式 (SMAF) 的多媒体文件。
<code>_capSMS</code>	表明 Flash Lite 是否能通过使用 <code>GetURL()</code> ActionScript 命令发送短消息服务 (SMS) 消息。
<code>_capStreamSound</code>	表明设备是否能播放（同步）声音流。
<code>_cap4WayKeyAS</code>	表明 Flash Lite 是否执行附加到与右箭头键、左箭头键、上箭头键和下箭头键相关联的按键事件处理函数的 ActionScript 表达式。
<code>\$version</code>	包含 Flash Lite 的版本号。
<code>fscommand()</code>	用于执行 Launch 命令的函数（请参见下一项）。
<code>Launch</code>	（唯一支持 <code>fscommand()</code> 的命令）允许 SWF 文件与 Flash Lite 或主机环境（如电话操作系统或设备操作系统）进行通信。
<code>fscommand2()</code>	用于执行此表中除 <code>fscommand()</code> 之外的命令的函数。

语言元素	说明
<code>Escape</code>	将任意字符串编码为可进行安全网络传输的格式。
<code>FullScreen</code>	设置用于呈现的显示区域大小。
<code>GetBatteryLevel</code>	返回当前电池电量。
<code>GetDateDay</code>	以数值形式返回当前日期的日。
<code>GetDateMonth</code>	以数值形式返回当前日期的月份。
<code>GetDateWeekday</code>	以数值形式返回当前日期的日。
<code>GetDateYear</code>	返回表明当前日期的年份的四位数。
<code>GetDevice</code>	设置标识运行 Flash Lite 的设备的参数。
<code>GetDeviceID</code>	设置表示设备的唯一标识符（如序列号）的参数。
<code>GetFreePlayerMemory</code>	返回当前可用于 Flash Lite 的堆内存量（以千字节为单位）。
<code>GetLanguage</code>	设置标识设备当前所用语言的参数。
<code>GetLocaleLongDate</code>	将参数设置为以长形式表示当前日期的字符串，其格式按照当前定义的区域设置。
<code>GetLocaleShortDate</code>	将参数设置为以缩略形式表示当前日期的字符串，其格式按照当前定义的区域设置。
<code>GetLocaleTime</code>	将参数设置为表示当前时间的字符串，其格式按照当前定义的区域设置。
<code>GetMaxBatteryLevel</code>	返回设备的最大电池电量。
<code>GetMaxSignalLevel</code>	返回最大信号强度级别。
<code>GetMaxVolumeLevel</code>	以数值形式返回设备的最大音量级别。
<code>GetNetworkConnectStatus</code>	返回表示当前网络连接状态的值。
<code>GetNetworkName</code>	将参数设置为当前网络的名称。
<code>GetNetworkRequestStatus</code>	返回表示最近 HTTP 请求状态的值。
<code>GetNetworkStatus</code>	返回表示电话的网络状态的值（即是否注册了网络以及电话当前是否在漫游）。
<code>GetPlatform</code>	设置标识当前平台的参数，它详细说明了设备的类别。对于使用开放操作系统的设备而言，此标识符通常是操作系统的名称和版本。
<code>GetPowerSource</code>	返回表示电源当前是由电池提供还是由外接电源提供的值。
<code>GetSignalLevel</code>	以数值形式返回当前信号强度。
<code>GetTimeHours</code>	以数值形式返回当前时间的小时数，使用 24 小时制时钟。
<code>GetTimeMinutes</code>	以数值形式返回当前时间的分钟数。
<code>GetTimeSeconds</code>	以数值形式返回当前时间的秒钟数。

语言元素	说明
<code>GetTimeZoneOffset</code>	将参数设置为本地时区和通用时间 (UTC) 之间相差的分钟数。
<code>GetTotalPlayerMemory</code>	返回分配给 Flash Lite 的总计堆内存量（以千字节为单位）。
<code>GetVolumeLevel</code>	以数值形式返回设备的当前音量级别。
<code>Quit</code>	使 Flash Lite 播放器停止播放并退出。
<code>ResetSoftKeys</code>	将软键重置为其原设置。
<code>SetInputTextType</code>	指定用于打开输入文本字段的模式。
<code>SetQuality</code>	设置动画的呈现品质。
<code>SetSoftKeys</code>	重新映射设备的左软键和右软键，前提是可以访问和重新映射这两个软键。
<code>StartVibrate</code>	启动电话的振动功能。
<code>StopVibrate</code>	停止当前的振动功能（如果有）。
<code>Unescape</code>	将编码为安全网络传输形式的任意字符串解码为其正常、未编码形式。

功能

本部分介绍 Macromedia Flash Lite 1.1 能够识别的平台功能和变量。各个条目按字母顺序列出，忽略前导下划线。

_capCompoundSound

可用性

Flash Lite 1.1。

用法

`_capCompoundSound`

说明

数值变量；表明 Flash Lite 是否能处理复合声音数据。如果能，则定义此变量并具有值 1；如果不能，则不定义此变量。

例如，单一的 Flash 文件可以包含以 MIDI 和 MFi 格式表示的同一个声音。播放器稍后会根据设备所支持的格式以相应格式回放数据。此变量定义 Flash Lite 播放器是否在当前手机上支持此功能。

在以下示例中，在 **Flash Lite 1.1** 中将 `useCompoundSound` 设置为 **1**，但在 **Flash Lite 1.0** 中不进行定义：

```
useCompoundSound = _capCompoundSound;

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

`_capEmail`

可用性

Flash Lite 1.1。

用法

`_capEmail`

说明

数值变量；表明 **Flash Lite** 客户端是否能够通过使用 `GetURL()` **ActionScript** 命令发送电子邮件。如果能，则定义此变量并具有值 **1**；如果不能，则不定义此变量。

示例

如果主机应用程序能够通过使用 `GetURL()` **ActionScript** 命令发送电子邮件，则以下示例会将 `canEmail` 设置为 **1**：

```
canEmail = _capEmail;

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

`_capLoadData`

可用性

Flash Lite 1.1。

用法

`_capLoadData`

说明

数值变量；表明主机应用程序是否能够通过调用 `loadMovie()`、`loadMovieNum()`、`loadVariables()` 和 `loadVariablesNum()` 函数动态加载其它数据。如果能，则定义此变量并具有值 **1**；如果不能，则不定义此变量。

示例

如果主机应用程序能够执行影片和变量的动态加载，则以下示例会将 `iCanLoad` 设置为 **1**：

```
canLoad = _capLoadData;
```

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

_capMFi

可用性

Flash Lite 1.1。

用法

```
_capMFi
```

说明

数值变量；表明设备是否能播放 **i-mode (MFi)** 音频格式的声音数据。如果能，则定义此变量并具有值 **1**；如果不能，则不定义此变量。

示例

如果设备能够播放 **MFi** 声音数据，则以下示例会将 `canMFi` 设置为 **1**：

```
canMFi = _capMFi;
```

```
if (canMFi == 1) {  
    // 将影片剪辑按钮发送到具有触发事件声音按钮的帧  
    tellTarget("buttons") {  
        gotoAndPlay(2);  
    }  
}
```

_capMIDI

可用性

Flash Lite 1.1。

用法

`_capMIDI`

说明

数值变量；表明设备是否能播放乐器数字接口 (MIDI) 音频格式的声音数据。如果能，则定义此变量并具有值 1；如果不能，则不定义此变量。

示例

如果设备能够播放 MIDI 声音数据，则以下示例会将 `canMidi` 设置为 1：

```
canMIDI = _capMIDI;

if (canMIDI == 1) {
    // 将影片剪辑按钮发送到具有触发事件声音按钮的帧
    tellTarget("buttons") {
        gotoAndPlay(2);
    }
}
```

_capMMS

可用性

Flash Lite 1.1。

用法

`_capMMS`

说明

数值变量；表明 **Flash Lite** 是否能通过使用 `GetURL()` **ActionScript** 命令发送多媒体消息服务 (MMS) 消息。如果能，则定义此变量并具有值 1；如果不能，则不定义此变量。

示例

以下示例在 **Flash Lite 1.1** 中将 `canMMS` 设置为 `1`，但在 **Flash Lite 1.0** 中保留为 `undefined`（但由于并不是所有 **Flash Lite 1.1** 电话都能发送 MMS 消息，因此，此代码的使用还取决于电话）：

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

_capMP3

可用性

Flash Lite 1.1。

用法

`_capMP3`

说明

数值变量；表明设备是否能播放 **MPEG 音频层 3 (MP3)** 音频格式的声音数据。如果能，则定义此变量并具有值 `1`；如果不能，则不定义此变量。

示例

如果设备能够播放 **MP3** 声音数据，则以下示例会将 `canMP3` 设置为 `1`：

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

_capSMAF

可用性

Flash Lite 1.1。

用法

_capSMAF

说明

数值变量；表明设备是否能播放合成音乐移动应用程序格式 (SMAF) 的多媒体文件。如果能，则定义此变量并具有值 1；如果不能，则不定义此变量。

示例

以下示例在 Flash Lite 1.1 中将 canSMAF 设置为 1，但在 Flash Lite 1.0 中保留为 **undefined**（但由于并不是所有 Flash Lite 1.1 电话都能发送 SMAF 消息，因此，此代码的使用还取决于电话）：

```
canSMAF = _capSMAF;

if (canSMAF) {
    // 将影片剪辑按钮发送到具有触发事件声音按钮的帧
    tellTarget("buttons") {
        gotoAndPlay(2);
    }
}
```

_capSMS

可用性

Flash Lite 1.1。

用法

_capSMS

说明

数值变量；表明 Flash Lite 是否能通过使用 GetURL() **ActionScript** 命令发送短消息服务 (SMS) 消息。如果能，则定义此变量并具有值 1；如果不能，则不定义此变量。

示例

以下示例在 **Flash Lite 1.1** 中将 `canSMS` 设置为 `1`，但在 **Flash Lite 1.0** 中保留为 `undefined`（但由于并不是所有 **Flash Lite 1.1** 电话都能发送 SMS 消息，因此，此代码的使用还取决于电话）：

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

_capStreamSound

可用性

Flash Lite 1.1。

用法

`_capStreamSound`

说明

数值变量；指示设备是否能播放（同步）声音流。如果能，则定义此变量并具有值 `1`；如果不能，则不定义此变量。

示例

如果启用了 `canStreamSound`，则以下示例播放声音流：

```
on(press) {  
    canStreamSound = _capStreamSound;  
    if (canStreamSound) {  
        // 用此按钮播放影片剪辑中的声音流  
        tellTarget("music") {  
            gotoAndPlay(2);  
        }  
    }  
}
```

_cap4WayKeyAS

可用性

Flash Lite 1.1。

用法

_cap4WayKeyAS

说明

数值变量；表明 **Flash Lite** 是否执行附加到与右箭头键、左箭头键、上箭头键和下箭头键相关联的按键事件处理函数的 **ActionScript** 表达式。只有当主机应用程序使用四向键导航模式在 **Flash** 控件（按钮和输入文本字段）之间移动时，才会定义此变量并具有值 1。否则，不定义此变量。

在按下其中一个四向键时，如果此变量的值为 1，则 **Flash Lite** 首先查找该键的处理函数。如果未找到，则会进行 **Flash** 控件导航。但是，如果找到事件处理函数，则不会执行该键的导航动作。例如，如果找到了下箭头键的按键处理函数，则用户不能进行导航。

示例

以下示例在 **Flash Lite 1.1** 中将 canUse4Way 设置为 1，而在 **Flash Lite 1.0** 中不进行定义（但由于并不是所有 **Flash Lite 1.1** 电话都支持四向键，因此，这段代码的使用还取决于电话）：

```
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

可用性

Flash Lite 1.1。

用法

\$version

说明

字符串变量；包含 **Flash Lite** 的版本号。它包含主版本号、次版本号、生成版本号和内部生成版本号，它在所有已发布的版本中通常为 0。

为所有 Flash Lite 1.x 产品报告的主版本号都为 5。Flash Lite 1.0 的次版本号为 1；Flash Lite 1.1 的次版本号为 2。

示例

在 Flash Lite 1.1 播放器中，以下代码将 myVersion 的值设置为 “5, 2, 12, 0”：

```
myVersion = $version;
```

fscommand()

可用性

Flash Lite 1.1。

用法

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

参数

"Launch" 命令说明符。Launch 命令是可以使用 fscommand() 函数执行的唯一的命令。

"application-path, arg1, arg2,..., argn" 启动的应用程序的名称及其参数，用逗号分隔。

说明

函数：允许 SWF 文件与 Flash Lite 或主机环境（如电话操作系统或设备操作系统）进行通信。

另请参见

[fscommand2\(\)](#)

Launch

可用性

Flash Lite 1.1。

用法

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

参数

"Launch" 命令说明符。在 Flash Lite 中，只能使用 fscommand() 函数来执行 Launch 命令。

"application-path, arg1, arg2,..., argn" 启动的应用程序的名称及其参数，用逗号分隔。

说明

通过 `fscommand()` 函数执行的命令：启动设备上的其它应用程序。被启动应用程序的名称及其参数作为单个参数传入。



此功能依赖于操作系统。

仅当 **Flash Lite** 播放器在独立模式中运行时才支持此命令。播放器运行于其它应用程序环境（如浏览器的插件）中时，不支持此命令。

示例

以下示例将在 **Series 60** 电话的服务 /Web 浏览器上打开 `wap.yahoo.com`：

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

另请参见

[fscommand2\(\)](#)

fscommand2()

可用性

Flash Lite 1.1。

用法

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

参数

command 传递给主机应用程序用于任何用途的一个字符串，或传递给 **Flash Lite** 的一个命令。

parameter1...parameterN 作为参数传递给由 *command* 指定的命令的字符串列表（以逗号分隔）。

说明

函数：允许 SWF 文件与 **Flash Lite** 或主机环境（如电话操作系统或设备操作系统）进行通信。`fscommand2()` 返回的值取决于特定的命令。

`fscommand2()` 函数类似于 `fscommand()` 函数，但有以下区别：

- `fscommand2()` 函数可以使用任意数量的参数。
- **Flash Lite** 可以立即执行 `fscommand2()`，而 `fscommand()` 在被处理的帧结束时执行。
- `fscommand2()` 函数可以返回一个用于报告成功、失败或命令结果的值。

本部分的表中描述了作为命令和参数传递给函数的字符串和表达式。

该表具有以下三列：

- “命令” 列显示标识命令的字符串文本参数。
- “参数” 列说明要为其它参数传递的值（如果有）的类型。
- “返回值” 列说明预期的返回值。

示例

本部分的其余部分介绍了可以使用 `fscommand2()` 函数执行的特定命令的示例。

另请参见

[fscommand\(\)](#)

Escape

可用性

Flash Lite 1.1。

说明

将任意字符串编码为可进行安全网络传输的格式。用十六进制转义序列（多字节字符时为 `%xx` 或 `%xx%xx`）替换每个非字母数字字符。

命令	参数	返回值
"Escape"	<i>original</i> 要编码为安全 URL 的格式的字符串。 <i>encoded</i> 生成的编码字符串。 这些参数可以是变量名称或常数字符串值（如 "Encoded_String"）。	0: 失败。 1: 成功。

示例

以下示例显示示例字符串向其编码形式的转换：

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

另请参见

[Unescape](#)

FullScreen

可用性

Flash Lite 1.1。

说明

设置用于呈现的显示区域大小。大小可以为全屏，也可以小于全屏。

仅当 **Flash Lite** 在独立模式中运行时，才支持此命令。播放器运行于其它应用程序环境（如浏览器的插件）中时，不支持此命令。

命令	参数	返回值
"FullScreen"	<i>size</i> 一个已定义的变量或一个常数字串值，具有以下值之一：true（全屏）或 false（小于全屏）。任何其它值将被视为值 false。	-1: 不支持。 0: 支持。

示例

以下示例尝试将显示区域设置为全屏。如果返回值不是 0，则它会将播放头转到标签为 **smallScreenMode** 的帧：

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

GetBatteryLevel

可用性

Flash Lite 1.1。

说明

返回当前电池电量。它是一个数值，其范围介于 0 到由 **GetMaxBatteryLevel** 变量返回的最大值之间。

命令	参数	返回值
"GetBatteryLevel"	无。	-1: 不支持。 其它数值：当前电池电量。

示例

以下示例将 `battLevel` 变量设置为电池的当前电量:

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

另请参见

[GetMaxBatteryLevel](#)

GetDateDay

可用性

Flash Lite 1.1。

说明

返回当前日期的日。它是一个数值（不带前导 0）。有效日从 1 到 31。

命令	参数	返回值
"GetDateDay"	无。	-1: 不支持。 1 到 31: 月份中的日。

示例

以下示例收集日期信息并构造一个完整的日期字符串:

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

另请参见

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

GetDateMonth

可用性

Flash Lite 1.1。

说明

以数值形式返回当前日期的月份（不带前导 0）。

命令	参数	返回值
"GetDateMonth"	无。	-1: 不支持。 1 到 12: 当前月份的数字。

示例

以下示例收集日期信息并构造一个完整的日期字符串：

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

另请参见

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

GetDateWeekday

可用性

Flash Lite 1.1。

说明

返回表明当前日期的日是星期几的数值。

命令	参数	返回值
"GetDateWeekday"	无。	-1: 不支持。 0: 星期日。 1: 星期一。 2: 星期二。 3: 星期三。 4: 星期四。 5: 星期五。 6: 星期六。

示例

以下示例收集日期信息并构造一个完整的日期字符串：

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

另请参见

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

GetDateYear

返回表明当前日期的年份的四位数值。

命令	参数	返回值
"GetDateYear"	无。	-1: 不支持。 0 到 9999: 当前年份。

可用性

Flash Lite 1.1。

示例

以下示例收集日期信息并构造一个完整的日期字符串：

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

另请参见

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

GetDevice

设置标识运行 **Flash Lite** 的设备的参数。此标识符通常为模型名称。

命令	参数	返回值
"GetDevice"	<i>device</i> 要接收设备标识符的字符串。它可以是变量名，也可以是包含变量名的字符串值。	-1: 不支持。 0: 支持。

可用性

Flash Lite 1.1。

示例

以下代码示例将设备标识符赋予 `statusdevice` 变量，然后用通用设备名称更新文本字段。

下面是一些示例结果及其表示的设备：

D506i Mitsubishi 506i 电话。

DFOMA1 Mitsubishi FOMA1 电话。

F506i Fujitsu 506i 电话。

FFOMA1 Fujitsu FOMA1 电话。

N506i NEC 506i 电话。

NFOMA1 NEC FOMA1 电话。

Nokia3650 Nokia 3650 电话。

p506i Panasonic 506i 电话。

PFOMA1 Panasonic FOMA1 电话。

SH506i Sharp 506i 电话。

SHFOMA1 Sharp FOMA1 电话。

S0506i Sony 506i 电话。

```
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 "SHFOMA1":
    /:myText += "device: Sharp FOMA1" add newline;
    break;
case "S0506i":
    /:myText += "device: Sony 506i" add newline;
    break;
}

```

GetDeviceID

设置表示设备的唯一标识符（如序列号）的参数。

命令	参数	返回值
"GetDeviceID"	<i>id</i> 要接收设备的唯一标识符的字符串。它可以是变量名，也可以是包含变量名的字符串值。	-1: 不支持。 0: 支持。

可用性

Flash Lite 1.1。

示例

以下示例将唯一的标识符赋予 `deviceId` 变量：

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

GetFreePlayerMemory

返回当前可用于 Flash Lite 的堆内存量（以千字节为单位）。

命令	参数	返回值
"GetFreePlayerMemory"	无。	-1: 不支持。 0 或正值：堆内存的可用千字节数。

可用性

Flash Lite 1.1。

示例

以下示例将状态设置为等于可用内存量：

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

另请参见

[GetTotalPlayerMemory](#)

GetLanguage

可用性

Flash Lite 1.1。

设置标识设备当前所用语言的参数。语言以字符串的形式在按名称传递的变量中返回。

命令	参数	返回值
"GetLanguage"	<p><i>language</i> 要接收语言代码的字符串。它可以是变量名，也可以是包含变量名的字符串值。返回值是以下之一：</p> <p>cs: 捷克语。 da: 丹麦语。 de: 德语。 en-UK: 英式英语或国际英语。 en-US: 美式英语。 es: 西班牙语。 fi: 芬兰语。 fr: 法语。 hu: 匈牙利语。 it: 意大利语。 jp: 日语。 ko: 朝鲜语。 nl: 荷兰语。 no: 挪威语。 pl: 波兰语。 pt: 葡萄牙语。 ru: 俄语。 sv: 瑞典语。 tr: 土耳其语。 xu: 未确定的语言。 zh-CN: 简体中文。 zh-TW: 繁体中文。</p>	<p>-1: 不支持。 0: 支持。</p>

提醒	在将日语电话设置为显示英语时，将会为 <i>language</i> 返回 en_US。
----	--

示例

以下示例将语言代码赋予 language 变量，然后用 **Flash Lite** 播放器所识别的语言来更新文本字段：

```
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

可用性

Flash Lite 1.1。

说明

将参数设置为以长形式表示当前日期的字符串，其格式按照当前定义的区域设置。

命令	参数	返回值
"GetLocaleLongDate"	<p><i>longdate</i> 要接收当前日期的长形式值的字符串变量，如 "October 16, 2004" 或 "16 October 2004"。</p> <p>它可以是变量名，也可以是包含变量名的字符串值。</p> <p><i>longdate</i> 中返回的值是一个多字符、变量长度的字符串。实际的格式取决于设备和区域设置。</p>	<p>-1: 不支持。</p> <p>0: 支持。</p>

示例

以下示例尝试将当前日期的长形式赋予 `longDate` 变量。它还设置 `status` 的值以报告是否能执行此操作。

```
status = fscommand2("GetLocaleLongDate", "longdate");
trace (longdate);           // 输出: Tuesday, June 14, 2005
```

另请参见

[GetLocaleShortDate](#)、[GetLocaleTime](#)

GetLocaleShortDate

可用性

Flash Lite 1.1。

说明

将参数设置为以缩略形式表示当前日期的字符串，其格式按照当前定义的区域设置。

命令	参数	返回值
"GetLocaleShortDate"	<i>shortdate</i> 要接收当前日期的短形式值的字符串变量，如 "10/16/2004" 或 "16-10-2004"。 它可以是变量名，也可以是包含变量名的字符串值。 <i>shortdate</i> 中返回的值是一个多字符、变量长度的字符串。实际的格式取决于设备和区域设置。	-1: 不支持。 0: 支持。

示例

以下示例尝试将当前日期的缩略形式赋予 `shortDate` 变量。它还设置 `status` 的值以报告是否能执行此操作。

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

另请参见

[GetLocaleLongDate](#)、[GetLocaleTime](#)

GetLocaleTime

可用性

Flash Lite 1.1。

说明

将参数设置为表示当前时间的字符串，其格式按照当前定义的区域设置。

命令	参数	返回值
"GetLocaleTime"	<i>time</i> 要接收当前时间值的字符串变量，如 "6:10:44 PM" 或 "18:10:44"。 它可以是变量名，也可以是包含变量名的字符串值。 <i>time</i> 中返回的值是一个多字符、变量长度的字符串。实际的格式取决于设备和区域设置。	-1: 不支持。 0: 支持。

示例

以下示例尝试将当前的本地时间赋予 *time* 变量。它还设置 *status* 的值以报告是否能执行此操作。

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

另请参见

[GetLocaleLongDate](#)、[GetLocaleShortDate](#)

GetMaxBatteryLevel

可用性

Flash Lite 1.1。

说明

返回设备的最大电池电量。它是一个大于 0 的数值。

命令	参数	返回值
"GetMaxBatteryLevel"	无。	-1: 不支持。 其它值: 最大电池电量。

示例

以下示例将 *maxBatt* 变量设置为最大电池电量：

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

GetMaxSignalLevel

可用性

Flash Lite 1.1。

说明

返回最大信号强度级别。它是一个大于 0 的数值。

命令	参数	返回值
"GetMaxSignalLevel"	无。	-1: 不支持。 其它数值: 最大信号强度。

示例

以下示例将最大信号强度赋予 sigStrengthMax 变量:

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

GetMaxVolumeLevel

可用性

Flash Lite 1.1。

说明

以数值形式返回设备的最大音量级别。

命令	参数	返回值
"GetMaxVolumeLevel"	无。	-1: 不支持。 其它值: 最大音量级别。

示例

以下示例将 maxvolume 变量设置为设备的最大音量级别:

```
maxvolume = fscommand2("GetMaxVolumeLevel");  
trace (maxvolume);    // 输出: 80
```

另请参见

[GetVolumeLevel](#)

GetNetworkConnectStatus

可用性

Flash Lite 1.1。

说明

返回表示当前网络连接状态的值。

命令	参数	返回值
"GetNetworkConnectStatus"	无。	-1: 不支持。 0: 当前有活动的网络连接。 1: 设备正在尝试连接到网络。 2: 当前没有活动的网络连接。 3: 网络连接已暂停。 4: 无法确定网络连接。

示例

以下示例将网络连接状态赋予 `connectstatus` 变量，然后使用 `switch` 语句用该连接状态更新文本字段：

```
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

可用性

Flash Lite 1.1。

说明

将参数设置为当前网络的名称。

命令	参数	返回值
"GetNetworkName"	<i>networkName</i> 表示网络名称的字符串。它可以是变量名，也可以是包含变量名的字符串值。 如果网络已经注册，并且可以确定其名称，则将 <i>networkname</i> 设置为网络名称；否则，将其设置为空字符串。	-1: 不支持。 0: 网络没有注册。 1: 网络已经注册，但网络名称未知。 2: 网络已经注册，并且网络名称已知。

示例

以下示例将当前网络的名称赋予 `myNetName` 变量，并将状态值赋予 `netNameStatus` 变量：
`netNameStatus = fscommand2("GetNetworkName", myNetName);`

GetNetworkRequestStatus

可用性

Flash Lite 1.1。

说明

返回表示最近 HTTP 请求状态的值。

命令	参数	返回值
"GetNetworkRequestStatus"	无。	<p>-1: 不支持该命令。</p> <p>0: 存在未处理的请求，已经建立了网络连接，已解析了服务器的主机名称并已建立了到服务器的连接。</p> <p>1: 存在未处理的请求，已经建立了网络连接。</p> <p>2: 存在未处理的请求，但未建立网络连接。</p> <p>3: 存在未处理的请求，已经建立了网络连接，正在解析服务器的主机名称。</p> <p>4: 由于网络错误，请求失败。</p> <p>5: 由于连接到服务器失败，请求失败。</p> <p>6: 服务器已返回 HTTP 错误（如 404）。</p> <p>7: 由于访问 DNS 服务器失败或解析服务器名称失败，请求失败。</p> <p>8: 请求已成功完成。</p> <p>9: 由于超时，请求失败。</p> <p>10: 尚未进行请求。</p>

示例

以下示例将最新 HTTP 请求的状态赋予 requeststatus 变量，然后使用 switch 语句用该状态更新文本字段：

```
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

可用性

Flash Lite 1.1。

说明

返回表示电话的网络状态的值（即是否注册了网络和电话当前是否在漫游）。

命令	参数	返回值
"GetNetworkStatus"	无。	-1: 不支持该命令。 0: 未注册网络。 1: 在本地网络中。 2: 在扩展的本地网络中。 3: 漫游（离开本地网络）。

示例

以下示例将网络连接状态赋予 `networkstatus` 变量，然后使用 `switch` 语句用该状态更新文本字段：

```
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

可用性

Flash Lite 1.1。

说明

设置标识当前平台的参数，它详细说明了设备的类别。对于使用开放操作系统的设备而言，此标识符通常是操作系统的名称和版本。

命令	参数	返回值
"GetPlatform"	<i>platform</i> 要接收平台标识符的字符串。它可以是变量名，也可以是包含变量名的字符串值。	-1: 不支持。 0: 支持。

示例

以下示例将平台标识符赋予 `statusplatform` 变量，然后用通用平台名称更新文本字段。

下面是 `myPlatform` 的一些示例结果及其表示的设备类别：

506i **506i** 电话。

FOMA1 **FOMA1** 电话。

Symbian6.1_s60.1 **Symbian 6.1, Series 60** 版本 1 电话。

Symbian7.0 **Symbian 7.0** 电话

```
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

可用性

Flash Lite 1.1。

说明

返回表示电源当前是由电池提供还是由外接电源提供的值。

命令	参数	返回值
"GetPowerSource"	无。	-1: 不支持。 0: 设备使用电池电源进行操作。 1: 设备使用外部电源进行操作。

示例

以下示例设置 myPower 变量以表明电源；如果不能表明电源，则会将其设置为 -1：
myPower = fscommand2("GetPowerSource");

GetSignalLevel

可用性

Flash Lite 1.1。

说明

以数值形式返回当前信号强度。

命令	参数	返回值
"GetSignalLevel"	无。	-1: 不支持。 其它数值: 当前信号强度, 范围介于 0 到由 GetMaxSignalLevel 返回的最大值之间。

示例

以下示例将信号强度值赋予 sigLevel 变量:

```
sigLevel = fscommand2("GetSignalLevel");
```

GetTimeHours

可用性

Flash Lite 1.1。

说明

返回当前时间的小时数, 使用 24 小时制时钟。它是一个数值 (不带前导 0)。

命令	参数	返回值
"GetTimeHours"	无。	-1: 不支持。 0 到 23: 当前小时。

示例

以下示例将 hour 变量设置为日的当前时间的小时部分, 或者设置为 -1:

```
hour = fscommand2("GetTimeHours");  
trace (hour);           // 输出: 14
```

另请参见

[GetTimeMinutes](#)、[GetTimeSeconds](#)、[GetTimeZoneOffset](#)

GetTimeMinutes

可用性

Flash Lite 1.1。

说明

返回当前时间的分钟数。它是一个数值（不带前导 0）。

命令	参数	返回值
"GetTimeMinutes"	无。	-1: 不支持。 0 到 59: 当前分钟。

示例

以下示例将 minutes 变量设置为当前时间的分钟部分，或者设置为 -1：

```
minutes = fscommand2("GetTimeMinutes");  
trace (minutes);           // 输出: 38
```

另请参见

[GetTimeHours](#)、[GetTimeSeconds](#)、[GetTimeZoneOffset](#)

GetTimeSeconds

可用性

Flash Lite 1.1。

说明

返回当前时间的秒钟数。它是一个数值（不带前导 0）。

命令	参数	返回值
"GetTimeSeconds"	无。	-1: 不支持。 0 到 59: 当前秒钟。

示例

以下示例将 seconds 变量设置为当前时间的秒部分，或者设置为 -1：

```
seconds = fscommand2("GetTimeSeconds");  
trace (seconds);           // 输出: 41
```

另请参见

[GetTimeHours](#)、[GetTimeMinutes](#)、[GetTimeZoneOffset](#)

GetTimeZoneOffset

可用性

Flash Lite 1.1。

说明

将参数设置为本地时区和通用时间 (UTC) 之间相差的分钟数。

命令	参数	返回值
"GetTimeZoneOffset"	<i>timezoneOffset</i> 本地时区与 UTC 之间相差的分钟数。它可以是变量名，也可以是包含变量名的字符串值。 返回一个正数值或一个负数值，如： 540：日本标准时间 -420：太平洋夏令时	-1：不支持。 0：支持。

示例

以下示例将与 UTC 偏差的分钟数赋予 `timezoneoffset` 变量并将 `status` 设置为 0 或将 `status` 设置为 -1：

```
status = fscommand2("GetTimeZoneOffset", "timezoneoffset");  
trace (timezoneoffset);// 输出: 300
```

另请参见

[GetTimeHours](#)、[GetTimeMinutes](#)、[GetTimeSeconds](#)

GetTotalPlayerMemory

可用性

Flash Lite 1.1。

说明

返回分配给 Flash Lite 的总计堆内存量（以千字节为单位）。

命令	参数	返回值
"GetTotalPlayerMemory"	无。	-1：不支持。 0 或正值：堆内存总计千字节数。

示例

以下示例将 status 变量设置为总计堆内存量：

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

另请参见

[GetFreePlayerMemory](#)

GetVolumeLevel

可用性

Flash Lite 1.1。

说明

以数值形式返回设备的当前音量级别。

命令	参数	返回值
"GetVolumeLevel"	无。	-1: 不支持。 其它数值：当前音量级别，范围介于 0 到由 fscommand2("GetMaxVolumeLevel") 返回的值之间。

示例

以下示例将当前音量级别赋予 volume 变量：

```
volume = fscommand2("GetVolumeLevel");  
trace (volume);           // 输出: 50
```

另请参见

[GetVolumeLevel](#)

Quit

可用性

Flash Lite 1.1。

说明

使 Flash Lite 播放器停止播放并退出。

仅当 Flash Lite 在独立模式中运行时，才支持此命令。播放器运行于其它应用程序环境（如浏览器的插件）中时，不支持此命令。

命令	参数	返回值
"Quit"	无。	-1: 不支持。

示例

在独立模式中运行时，以下示例使 Flash Lite 停止播放并退出：

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

ResetSoftKeys

可用性

Flash Lite 1.1。

说明

将软键重置为其原设置。

仅当 Flash Lite 在独立模式中运行时，才支持此命令。播放器运行于其它应用程序环境（如浏览器的插件）中时，不支持此命令。

命令	参数	返回值
"ResetSoftKeys"	无。	-1: 不支持。 0: 支持。

示例

以下语句将软键重置为其原始设置：

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

另请参见

[SetSoftKeys](#)

SetInputTextType

可用性
Flash Lite 1.1。

说明
指定用于打开输入文本字段的模式：

命令	参数	返回值
"SetInputTextType"	<i>variableName</i> 输入文本字段的名称。它可以是变量名，也可以是包含变量名的字符串值。 <i>type</i> Numeric、Alpha、Alphanumeric、Latin、NonLatin 或 NoRestriction 值中的一个。	0: 失败。 1: 成功。

Flash Lite 通过让主机应用程序启动特定于设备的通用文本输入界面（通常称为“前端处理器”（FEP））来支持输入文本功能。如果不使用 SetInputTextType 命令，则会在默认模式中打开 FEP。

下表显示每种模式所具有的作用，以及替换哪些模式：

指定的模式	将 FEP 设置为其中一种互相排斥的模式	当前设备上不支持时，在此模式中打开 FEP
Numeric	仅限数字（0 到 9）	Alphanumeric
Alpha	仅限字母字符（A 到 Z、a 到 z）	Alphanumeric
Alphanumeric	仅限字母数字字符（0 到 9、A 到 Z、a 到 z）	Latin
Latin	仅限拉丁语字符（字母数字和标点符号）	NoRestriction
NonLatin	仅限非拉丁语字符（如日文汉字和日文假名）	NoRestriction
NoRestriction	默认模式（在 FEP 上不设置限制）	

提醒

并不是所有移动电话都支持这些输入文本字段类型。因此，您必须验证输入文本数据。

示例
以下一行代码将与 input1 变量相关联的字段的输入文本类型设置为接收数值数据：
status = fscommand2("SetInputTextType", "input1", "Numeric");

SetQuality

可用性

Flash Lite 1.1。

说明

设置动画的呈现品质。

命令	参数	返回值
"SetQuality"	<i>quality</i> 呈现品质；必须为 "high"、 "medium" 或 "low"。	-1: 不支持。 0: 支持。

示例

以下示例将呈现品质设置为 **LOW**：

```
status = fscommand2("SetQuality", "low");
```

SetSoftKeys

可用性

Flash Lite 1.1。

说明

重新映射设备的左软键和右软键，前提是可以访问和重新映射这两个软键。

执行此命令后，按左键会生成一个 PageUp 按键事件，按右键会生成一个 PageDown 按键事件。分别按这两个键时，会执行与 PageUp 和 PageDown 按键事件相关联的 **ActionScript**。

仅当 **Flash Lite** 在独立模式中运行时，才支持此命令。播放器运行于其它应用程序环境（如浏览器的插件）中时，不支持此命令。

命令	参数	返回值
"SetSoftKeys"	<i>left</i> 要为左软键显示的文本。 <i>right</i> 要为右软键显示的文本。 这两个参数或者是变量名，或者是常数字符串值（如 "Previous"）。	-1: 不支持。 0: 支持。

示例

以下示例指示左软键的标签为 **Previous**，右软键的标签为 **Next**：

```
status = fscommand2("SetSoftKeys", "Previous", "Next");
```

另请参见

[ResetSoftKeys](#)

StartVibrate

可用性

Flash Lite 1.1。

说明

启动电话的振动功能。如果已经产生振动，则在启动新振动前，**Flash Lite** 会停止该振动。当 **Flash** 应用程序停止或暂停播放时以及当 **Flash Lite** 播放器退出时，也会停止振动。

命令	参数	返回值
"StartVibrate"	<i>time_on</i> 启动振动功能的时间，以毫秒为单位（最大值为 5 秒）。	-1: 不支持。 0: 已启动振动功能。
	<i>time_off</i> 关闭振动功能的时间，以毫秒为单位（最大值为 5 秒）。	1: 出现错误，不能启动振动功能。
	<i>repeat</i> 重复此振动的次数（最多为 3 次）。	

示例

以下示例尝试启动一个振动序列（振动 2.5 秒，停止 1 秒），重复两次。它会为 `status` 变量赋予一个值，用以表明成功或失败。

```
status = fscommand2("StartVibrate", 2500, 1000, 2);
```

另请参见

[StopVibrate](#)

StopVibrate

可用性

Flash Lite 1.1。

说明

停止当前的振动功能（如果有）。

命令	参数	返回值
"StopVibrate"	无。	-1: 不支持。 0: 振动已停止。

示例

以下示例调用 StopVibrate，然后将调用结果（不支持或振动已停止）保存在 status 变量中：

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

另请参见

[StartVibrate](#)

Unescape

可用性

Flash Lite 1.1。

说明

将编码为安全网络传输形式的任意字符串解码为其正常、未编码形式。十六进制格式（即一个百分号字符 (%) 后面跟随两位十六进制数字）的所有字符都将转换为其解码形式。

命令	参数	返回值
"Unescape"	<i>original</i> 要从安全 URL 格式解码为正常形式的字符串。 <i>decoded</i> 生成的解码字符串。 （此参数可以是变量名称，也可以是包含变量名的字符串值。）	0: 失败。 1: 成功。

示例

以下示例显示一个编码字符串的解码结果：

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

另请参见

[Escape](#)

索引

符号

! (逻辑 NOT) 运算符 82
" " (字符串分隔符) 运算符 92
\$version 变量 108
%= (模赋值) 运算符 84
% (模) 运算符 84
&& (逻辑 AND) 运算符 81
|| (逻辑 OR) 运算符 83
*= (乘法赋值) 运算符 85
* (乘法) 运算符 86
++ (递增) 运算符 80
+= (加法赋值) 运算符 72
+ (数字加法) 运算符 86
, (逗号) 运算符 75
-- (递减) 运算符 77
-= (减法赋值) 运算符 98
- (减法) 运算符 97
. (点) 运算符 79
/* (块注释) 运算符 74
// (注释) 运算符 76
/= (除法赋值) 运算符 79
/ (除法) 运算符 78
/ (正斜杠 - 根时间轴) 属性 44
== (数值等于) 运算符 87
= (赋值) 运算符 73
>= (大于或等于) 运算符 88
> (大于) 运算符 88
? (条件) 运算符 77
_alpha 变量 45
_cap4WayKeyAS 变量 108
_capCompoundSound 变量 101
_capEmail 变量 102
_capLoadData 变量 102
_capMFi 变量 103
_capMIDI 变量 104
_capMMS 变量 104
_capSMAF 变量 106

_capSMS 变量 106
_capStreamSound 变量 107
_currentframe 属性 45
_focusrect 属性 46
_framesloaded 属性 46
_height 属性 47
_highquality 属性 47
_level 属性 48
_name 属性 49
_rotation 属性 50
_scroll 属性 50
_target 属性 51
_visible 属性 52
_width 属性 52
_x 属性 53
_xscale 属性 53
_y 属性 54
_yscale 属性 55

英文

add (字符串连接) 运算符 71
_alpha 变量 45
AND 运算符 81
and 运算符 72
break 语句 58
call 11
_cap4WayKeyAS 变量 108
_capCompoundSound 变量 101
_capEmail 变量 102
_capLoadData 变量 102
_capMFi 变量 104
_capMMS 变量 104
_capSMAF 变量 106
_capSMS 变量 106
_capStreamSound 变量 107
case 语句 58

- chr() 函数 12
- continue 语句 60
- _currentframe 属性 45
- do..while 语句 61
- duplicateMovieClip() 函数 12
- else if 语句 63
- else 语句 62
- eq (字符串相等) 运算符 93
- eval() 函数 13
- _focusrect 属性 46
- for 循环 64
- for 语句 64
- _framesloaded 属性 46
- fscommand() 命令 109
- getProperty() 函数 14
- getTimer() 函数 15
- getURL() 函数 15
- ge (字符串大于或等于) 运算符 94
- gotoAndPlay() 函数 18
- gotoAndStop() 函数 19
- gt (字符串大于) 运算符 93
- _height 属性 47
- _highquality 属性 47
- if 语句 65
- ifFrameLoaded() 函数 19
- int() 函数 20
- length() 函数 21
- _level 属性 48
- le (字符串小于或等于) 运算符 96
- loadMovie() 函数 21
- loadMovieNum() 函数 22
- loadVariables() 函数 23
- loadVariablesNum() 函数 25
- lt (字符串小于) 运算符 95
- maxscroll 属性 49
- mbchr() 函数 26
- mbsubstring() 函数 28
- MFI 声音 103
- MIDI 声音 104
- MMS 消息 104
- _name 属性 49
- nextFrame() 函数 28
- nextScene() 函数 29
- ne (字符串不等于) 运算符 95
- NOT 运算符 82
- Number() 函数 30
- on() 函数 31
- OR 运算符 83
- ord() 函数 32
- play() 函数 32

- prevFrame() 函数 33
- prevScene() 函数 33
- random() 函数 34
- removeMovieClip() 函数 35
- _rotation 属性 50
- scroll 属性 50
- set() 函数 35
- setProperty() 函数 36
- stop() 函数 37
- stopAllSounds() 函数 37
- String() 函数 38
- substring() 函数 38
- switch 语句 65
- _target 属性 51
- tellTarget() 函数 39
- toggleHighQuality() 函数 40
- _totalframes 属性 51
- trace() 函数 40
- unloadMovie() 函数 41
- unloadMovieNum() 函数 42
- _visible 属性 52
- while 循环 61
- while 语句 67
- _width 属性 52
- _x 属性 53
- _xscale 属性 53
- _y 属性 54
- _yscale 属性 55
- <= (数值小于或等于) 运算符 90
- <> (数值不等于) 运算符 89
- < (数值小于) 运算符 90

B

变量

- \$version 108
- _alpha 45
- _cap4WayKeyAS 108
- _capCompoundSound 101
- _capEmail 102
- _capLoadData 102
- _capMFi 103
- _capMIDI 104
- _capMMS 104
- _capSMAF 106
- _capSMS 106
- _capStreamSound 107
- Flash Lite 的版本号 108
- 电子邮件功能 102
- 加载数据功能 102

箭头键导航 108
变量, 声音
 _capCompoundSound 101
 _capMFi 103
 _capMIDI 104
 _capSMAF 106
 _capStreamSound 107
变量, 消息
 _capMMS 104
 _capSMS 106
不等于运算符 89

C

乘法 86
除法 78
除法赋值运算符 79

D

大于或等于运算符 88
大于运算符 88
递增运算符 80
点运算符 79
电子邮件功能变量 102
逗号运算符 75

F

赋值运算符 73

G

根时间轴标识符 44

H

函数
 chr() 12
 duplicateMovieClip() 12
 eval() 13
 fscommand() 109
 getProperty() 14
 getTimer() 15
 getURL() 15
 gotoAndPlay() 18
 gotoAndStop() 19
 iffFrameLoaded() 19
 int() 20

length() 21
loadMovie() 21
loadMovieNum() 22
loadVariables() 23
loadVariablesNum() 25
mbchr() 26
mbsubstring() 28
nextFrame() 28
nextScene() 29
Number() 30
on() 31
ord() 32
play() 32
prevFrame() 33
prevScene() 33
random() 34
removeMovieClip() 35
set() 35
setProperty() 36
stop() 37
stopAllSounds() 37
String() 38
substring() 38
tellTarget() 39
toggleHighQuality() 40
trace() 40
unloadMovie() 41
unloadMovieNum() 42

J

加法赋值运算符 72
减法赋值运算符 98

K

块注释运算符 74

L

连接 71
逻辑 AND 运算符 81
逻辑 NOT 运算符 82
逻辑 OR 运算符 83

M

模赋值 84
模运算符 84

S

声音变量 101, 103, 104, 106, 107

属性

- _alpha 45
- _currentframe 45
- _focusrect 46
- _framesloaded 46
- _height 47
- _highquality 47
- _level 48
- _name 49
- _rotation 50
- _scroll 50
- _target 51
- _visible 52
- _width 52
- _x 53
- _xscale 53
- _y 54
- _yscale 55
- maxscroll 49
- scroll 50
- 正斜杠 44

数字加法 86

T

条件 65

条件运算符 77

X

消息变量 104, 106

小于或等于运算符 90

小于运算符 90

Y

语句

- break 58
- case 58
- continue 60
- do..while 61
- else 62
- else if 63
- for 64
- if 65
- switch 65
- while 67

逻辑 NOT 82

运算符

- and 72
- 乘法 86
- 除法 78
- 除法赋值 79
- 大于 88
- 大于或等于 88
- 递增 80
- 点 79
- 逗号 75
- 赋值 73
- 加法赋值 72
- 减法赋值 98
- 块注释 74
- 逻辑 AND 81
- 逻辑 NOT 82
- 逻辑 OR 83
- 模 84
- 模赋值 84
- 数值不等于 89
- 数值等于 87
- 数值小于 90
- 数值小于或等于 90
- 数字加法 86
- 条件 77
- 注释 76
- 字符串不等于 95
- 字符串大于 93
- 字符串大于或等于 94
- 字符串分隔符 92
- 字符串连接 71
- 字符串相等 93
- 字符串小于 95
- 字符串小于或等于 96

Z

注释

- 块 74
- 一行 76
- 字符串大于或等于 94
- 字符串大于运算符 93
- 字符串分隔符运算符 92
- 字符串相等运算符 93
- 字符串小于或等于 96