



学习 Flash Lite 1.x ActionScript

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上的内容负责。如果您访问本指南述及的第三方网站，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](http://www.nellymoser.com)) 的许可。



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

Opera® 浏览器版权所有 © 1995-2002 Opera Software ASA 及其提供商。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从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：Tim Statler

总编：Rosana Francescato

编辑：Linda Adler、Geta Carlson、Evelyn Eldridge、Mary Kraemer、Lisa Stanziano

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

<b>第 1 章：关于 Flash Lite 1.x ActionScript</b>	<b>5</b>
Flash Lite 1.x ActionScript 概述	5
Flash Lite 1.0 和 Flash Lite 1.1 ActionScript 之间的差异	6
不受 Flash Lite 1.x ActionScript 支持的 Flash 4 ActionScript	6
Flash Lite 1.x ActionScript 中不可用的功能	7
<b>第 2 章：Flash 4 ActionScript 基本知识</b>	<b>9</b>
获取和设置影片剪辑属性	9
控制其它时间轴	10
使用变量	10
模拟数组	11
使用文本和字符串	12
连接字符串	12
滚动文本	12
使用 call() 函数来创建函数	13
使用 eval() 函数	17
<b>第 3 章：常用撰写脚本任务</b>	<b>19</b>
确定设备和平台功能	19
打开网页	20
启动电话	20
启动文本或多媒体消息	21
启动电子邮件	21
加载外部 SWF 文件	22
加载外部数据	22
<b>索引</b>	<b>25</b>



# 关于 Flash Lite 1.x ActionScript

使用 ActionScript 可向 Macromedia Flash Lite 应用程序中添加编程逻辑和交互。Flash Lite 1.0 和 1.1 中的 ActionScript 版本（统称为 Flash Lite 1.x ActionScript）是 Flash 4 ActionScript 和特定于 Flash Lite 播放器的其它命令和属性（如启动电话或文本消息，或获取设备时间和日期信息的功能）的混合。

本章包含以下主题：

Flash Lite 1.x ActionScript 概述 .....	5
Flash Lite 1.0 和 Flash Lite 1.1 ActionScript 之间的差异 .....	6
不受 Flash Lite 1.x ActionScript 支持的 Flash 4 ActionScript .....	6
Flash Lite 1.x ActionScript 中不可用的功能 .....	7

## Flash Lite 1.x ActionScript 概述

Flash Lite 1.x ActionScript 由以下几部分组成：

**Flash Player 4 ActionScript** 其中包括运算符（如比较运算符和赋值运算符）、影片剪辑属性（如 `_height`、`_x` 和 `_y`）、时间轴控制函数（如 `gotoAndPlay()` 或 `stop()`）和网络函数（如 `loadVariables()` 和 `loadMovie()` 函数）（仅限 Flash Lite 1.1）。若要查看不受支持的 Flash 4 ActionScript 的列表，请参见第 6 页的“不受 Flash Lite 1.x ActionScript 支持的 Flash 4 ActionScript”。

**电话集成命令和属性** Flash Lite 提供了一些命令，例如，其中某些命令可用于查询设备的日期和时间信息、启动电话或短消息服务 (SMS) 文本消息，或者启动安装在设备上的外部应用程序。

**平台功能变量（仅限 Flash Lite 1.1）** 这些属性提供了有关设备功能或 Flash Lite 运行时环境的信息。例如，`_capLoadData` 变量指示应用程序是否可以通过网络加载数据。

**fscommand2() 函数** 与 `fscommand()` 函数一样，您可以使用 `fscommand2()` 来与主机环境或系统（此时指移动电话或设备）通信。`fscommand2()` 函数提供了 `fscommand()` 的增强功能，包括传递任意数量的参数和检索即时返回值（而不必像 `fscommand()` 那样必须等到下一帧）的功能。

# Flash Lite 1.0 和 Flash Lite 1.1 ActionScript 之间的差异

以下 Flash Lite 1.1 ActionScript 功能在 Flash Lite 1.0 中不可用：

- 网络访问或网络状态信息。例如，在 Flash Lite 1.0 中，您不能使用 `loadVariables()` 或 `loadMovie()` 函数来加载外部数据或 SWF 文件，也不能使用各种 `fscommand2()` 命令来确定设备的连接信号强度或网络请求的状态。
- 获取设备的时间和日期信息。
- 提供有关 Flash Lite 平台以及设备的功能信息的平台功能变量。
- `fscommand2()` 函数及其相关联的命令，如 `SetSoftKeys` 和 `FullScreen`。
- `scroll` 和 `maxscroll` 文本字段属性。

## 不受 Flash Lite 1.x ActionScript 支持的 Flash 4 ActionScript

以下 Flash 4 ActionScript 功能在 Flash Lite 1.x ActionScript 中不受支持或仅部分受支持：

- `startDrag()` 和 `stopDrag()` 函数。
- Flash Lite 1.x ActionScript 支持 Flash Player 4 中受支持的按钮事件的子集。有关处理按钮事件的更多信息，请参见《开发 Flash Lite 应用程序》中的第 1 章“创建交互和导航”。
- Flash Lite 1.x ActionScript 支持 Flash Player 4 中受支持的按键事件的子集。有关 Flash Lite 中受支持的按键事件的更多信息，请参见《开发 Flash Lite 应用程序》中的第 1 章“创建交互和导航”。
- `_dropTarget` 属性。
- `_soundBufTime` 属性。
- `_url` 属性。
- `String()` 转换函数。

# Flash Lite 1.x ActionScript 中不可用的功能

因为 Flash Lite 播放器基于旧版本的 Flash Player，因此它并不支持更新版本的 Flash Player 或您可能熟悉的其它编程语言中所有可用的编程功能。本节讨论 Flash Lite 1.x ActionScript 中不可用的编程功能，以及可用的替代功能和变通办法。

**用户定义的函数** Flash Lite 1.x 不支持定义和调用自定义函数的功能。但可以使用 `call()` 函数来执行位于时间轴中任一帧上的代码。有关更多信息，请参见第 13 页的“[使用 call\(\) 函数来创建函数](#)”。

**本机数组、对象或其它复杂数据类型** Flash Lite 1.x 不支持本机数组数据结构或其它复杂数据类型。但是，您可以使用伪数组来模拟数组。伪数组是一种需要使用 `eval()` 函数来动态计算相连字符串的技术。有关更多信息，请参见第 11 页的“[模拟数组](#)”。

**在运行时加载外部图像文件或声音文件** 与桌面版本的 Flash Player 不同，Flash Lite 1.x ActionScript 无法加载外部 JPEG 文件或 MP3 文件。在 Flash Lite 1.1 中，可以使用 `loadMovie()` 函数来加载外部 SWF 文件。有关更多信息，请参见第 22 页的“[加载外部 SWF 文件](#)”。





# Flash 4 ActionScript 基本知识

Flash Lite 1.x ActionScript 是基于最先随 Flash Player 4 推出的那个 ActionScript 版本。因此，Flash Player 以后版本（用于桌面系统）中推出的多种编程功能对于 Flash Lite 1.x 应用程序并不可用。

如果您不熟悉 Flash 4 ActionScript 语法和功能，或者如果您已经忘记了以前进行的 Flash 开发工作的细节，本章会提供有关在 Flash Lite 应用程序中使用 Flash 4 ActionScript 的基本知识。

本章包含以下主题：

获取和设置影片剪辑属性.....	9
控制其它时间轴.....	10
使用变量.....	10
模拟数组.....	11
使用文本和字符串.....	12
使用 call() 函数来创建函数.....	13
使用 eval() 函数.....	17

## 获取和设置影片剪辑属性

要获取或设置影片剪辑的属性（如果可以设置），可以使用点语法，也可以使用 `setProperty()` 或 `getProperty()` 函数。还可以使用 `tellTarget()` 函数。

要使用点语法，请指定影片剪辑实例名称，后面跟随一个点 (.)，然后指定属性名称。例如，以下代码获取名为 `cartoonArea` 的影片剪辑的 `x` 屏幕坐标（由 `_x` 影片剪辑属性表示），并将结果分配给名为 `x_pos` 的变量。

```
x_pos = cartoonArea._x;
```

以下示例等同于上一个示例，只不过使用 `getProperty()` 函数来检索影片剪辑的 `x` 位置：

```
x_pos = getProperty(cartoonArea, _x);
```

通过 `setProperty()` 函数可以设置某个影片剪辑实例的属性，如下面示例中所示：

```
setProperty(cartoonArea, _x, 100);
```

以下示例等同于上一个示例，只不过使用的是点语法：

```
cartoonArea._x = 100;
```

您也可以在 `tellTarget()` 语句中获取或设置影片剪辑属性。以下代码等效于前面所示的 `setProperty()` 示例：

```
tellTarget("/cartoonArea") {  
    _x = 100;  
}
```

有关 `tellTarget()` 函数的更多信息，请参见第 10 页的“控制其它时间轴”。

## 控制其它时间轴

若要指定时间轴的路径，请结合使用斜杠语法 (/) 和点 (..) 来生成路径引用。也可以通过 **Flash 5** 记号 `_levelN`、`_root` 或 `_parent` 来分别引用特定的影片级别、应用程序的根时间轴或父时间轴。

例如，假设您的 **SWF** 文件的主时间轴上有一个名为 `box` 的影片剪辑实例，而该 `box` 实例包含另一个名为 `cards` 的影片剪辑实例。以下示例以主时间轴中的影片剪辑 `cards` 作为目标：

```
tellTarget("/box/cards")  
tellTarget("_level0/box/cards")
```

以下示例以影片剪辑 `cards` 中的主时间轴作为目标：

```
tellTarget(".././cards")  
tellTarget("_root")
```

以下示例以父影片剪辑 `cards` 作为目标：

```
tellTarget("../cards")  
tellTarget("_parent/cards")
```

## 使用变量

若要在某个时间轴上指定一个变量，请结合使用斜杠语法 (/) 和点 (..) 以及冒号 (:)。也可以使用点记号。

以下代码引用主时间轴上的 `car` 变量：

```
/:car  
_root.car
```

以下示例在位于主时间轴上的影片剪辑实例中显示 `car` 变量：

```
/mc1/mc2/:car  
_root.mc1.mc2.car
```

以下示例在位于当前时间轴上的影片剪辑实例中显示 car 变量：

```
mc2/:car  
mc2.car
```

## 模拟数组

数组可用于创建和操作诸如变量和值这类信息的排序列表。不过，Flash Lite 1.1 并不支持本机数组数据结构。Flash Lite（和 Flash 4）编程中常用的一种方法是用字符串处理来模拟数组。模拟数组也称为伪数组。伪数组处理的关键是 eval() ActionScript 函数，该函数可以按名称访问变量、属性或影片剪辑。有关更多信息，请参见第 17 页的“使用 eval() 函数”。

伪数组通常由两个或多个变量构成，这些变量共享同一个基名称，后面跟随一个数字后缀。该后缀是每个数组元素的索引。

例如，假设您创建了以下 ActionScript 变量：

```
color_1 = "orange";  
color_2 = "green";  
color_3 = "blue";  
color_4 = "red";
```

则您可以使用以下代码在伪数组中的元素间循环：

```
for (i = 1; i <=4; i++) {  
    trace (eval ("color_" + i));  
}
```

除了可以引用现有的变量外，您还可以在变量赋值语句的左侧使用 eval() 函数，以便在运行时创建变量。例如，假设您想要在用户游戏时保持一份高分列表。用户每结束一轮游戏，您即将其得分添加到该列表中：

```
eval("highScore" + i + scoreIndex) = currentScore;  
scoreIndex++;
```

此代码每次运行时，都会向高分列表中添加一个新项目，然后使 scoreIndex 变量递增，从而确定每个项目在列表中的索引。例如，您最后可能会使用以下变量：

```
highScore1 = 2000  
highScore2 = 1500  
highScore3 = 3000
```

# 使用文本和字符串

Flash Lite 提供了一些基本的 **ActionScript** 命令和属性，用于处理文本。您可以获取和设置文本字段的值；连接字符串、URL 编码字符串或 URL 解码字符串以及创建滚动文本字段。

本部分包含以下主题：

- [第 12 页的“连接字符串”](#)
- [第 12 页的“滚动文本”](#)

## 连接字符串

若要在 **Flash Lite** 中连接字符串，可以使用 `add` 运算符，如以下示例中所示：

```
city = "Boston";  
team = "Red Sox";  
fullName = city add " " add team;  
// 结果：  
// fullName = "Boston Red Sox"
```

## 滚动文本

可以使用动态和输入文本字段的 `scroll` 属性来获取或设置字段的当前滚动位置。也可以使用 `maxscroll` 位置来确定文本字段相对于最大滚动位置的当前滚动位置。若要查看如何创建滚动文本字段的示例，请参见《开发 **Flash Lite** 应用程序》中的“创建滚动文本（仅限 **Flash Professional**）”。

# 使用 call() 函数来创建函数

不能像在 Flash Player 5 及更高版本中那样在 Flash Lite 中定义或调用自定义函数。但可以使用 `call()` **ActionScript** 函数来执行位于时间轴中任一帧上的代码。这种方法可以将常用的代码封装在单一的位置，使其易于维护。

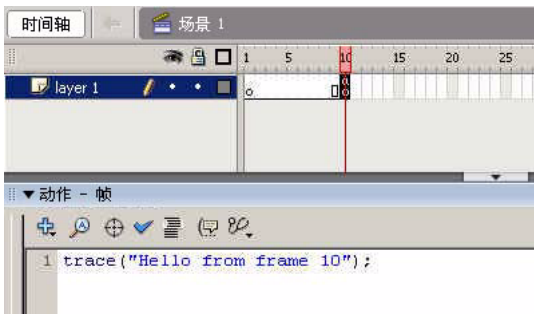
`call()` 函数将帧号或帧标签作为参数。例如，以下 **ActionScript** 调用位于标签为 `moveUp` 的帧上的代码：

```
call("moveUp");
```

`call()` 函数以同步方式操作：所指定帧上的所有 **ActionScript** 都执行完毕之后，`call()` 函数调用之后的任何 **ActionScript** 才会执行。

## 调用另一帧上的 ActionScript:

1. 在一个新建 Flash 文档中，在第 10 帧上插入一个关键帧。



2. 在新建的关键帧处于选中状态下，打开“动作”面板（“窗口” > “动作”），然后键入以下代码：

```
trace("Hello from frame 10");
```

3. 选择第 1 帧上的关键帧，然后在“动作”面板中键入以下代码：

```
stop();  
call(10);
```

这段代码会将播放头停在第 1 帧上，然后调用第 10 帧上的代码。

4. 在模拟器中测试应用程序，并打开“输出”面板（“窗口” > “输出”）。

您应该可以看到“Hello from frame 10”显示在“输出”面板中。

也可以调用位于另一时间轴（如影片剪辑的时间轴）上的代码。要执行代码，请指定影片剪辑实例名称，后面跟随一个冒号，然后指定帧编号或帧标签。例如，以下 **ActionScript** 调用位于名为 `callClip` 的影片剪辑实例中的标签为 `moveUp` 的帧上的代码：

```
call("callClip:moveUp");
```

此方法通常用于创建调用剪辑 或 函数剪辑，即仅用于封装常用代码的影片剪辑。调用剪辑包含用于要创建的每个函数的关键帧。通常需要根据用途对每个关键帧进行标记。**Macromedia** 还建议为每个新建关键帧创建一个新图层，并为每个图层指定与分配给该关键帧的帧标签相同的名称。

下图显示了一个调用剪辑示例的时间轴。调用剪辑的第一个关键帧始终包含一个 `stop()` 动作，这可确保播放头不会连续地在其时间轴中的各帧间循环。随后的关键帧包含每个“函数”的代码。每个函数关键帧都带有标签，以标识所执行的任务。要更容易地编辑和查看调用剪辑，每个函数关键帧通常都插入在一个单独的图层上。



以下过程说明了如何创建和使用调用剪辑。

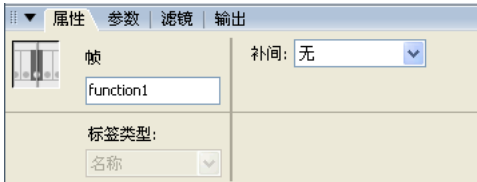
#### 要创建和使用调用剪辑：

1. 在 **Flash Professional 8** 中，通过 **Flash Lite 1.1 Symbian Series 60** 文档模板来创建新文档。
2. 选择“插入” > “新建元件”。
3. 在“创建新元件”对话框的“名称”文本框中，键入 **Call Clip**，然后单击“确定”。  
即会在编辑模式中打开该影片剪辑。
4. 单击“时间轴”窗口上的“添加新图层”按钮两次，以插入两个新图层。  
将顶部图层命名为 **Actions**，将第二个图层命名为 **function1**，将第三个图层命名为 **function2**。

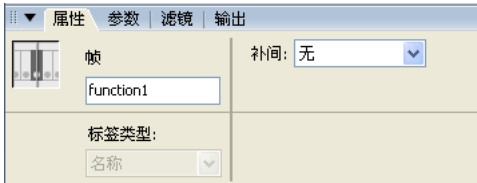
5. 在 `function1` 图层的第 2 帧上插入一个关键帧，并在 `function2` 图层的第 3 帧上插入另一个关键帧，如下图所示：



6. 选择 **Actions** 图层上的关键帧，并打开“动作”面板。
7. 向“动作”面板中添加一个 `stop()` 动作。
8. 选择 `function1` 图层的第 2 帧上的关键帧，并执行以下操作：
- a. 在“属性”检查器中，在“帧标签”文本框中键入 **function1**。

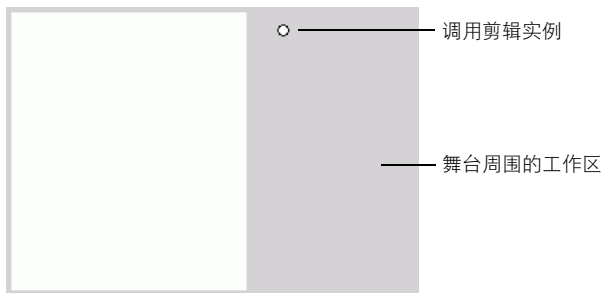


- b. 在“动作”面板（“窗口” > “动作”）中，键入以下代码：
- ```
trace("function1 was called.");
```
9. 选择 `function2` 图层的第 3 帧上的关键帧，并执行以下操作：
- a. 在“属性”检查器中，在“帧标签”文本框中键入 **function2**。



- b. 在“动作”面板（“窗口” > “动作”）中，键入以下代码：
- ```
trace("function2 was called.");
```
10. 按 **Ctrl+E** 组合键 (Windows) 或 **Command+E** 组合键 (Macintosh)，返回到主时间轴。
11. 通过选择“视图” > “工作区”，将文档视图设置为在舞台周围包括工作区。
- 因为调用剪辑不需要对用户可见，因此可以将其放置在工作区中。

12. 打开“库”面板（“窗口”>“库”），并将“调用剪辑”元件拖到舞台周围的工作区中。
- 调用剪辑不包含任何可视元素，因此它在舞台上显示为一个小圆圈，表示影片剪辑的注册点。



提示

要使用调用剪辑在舞台上更易于识别，请向调用剪辑的时间轴中的第一个关键帧添加某些文本或其它可视元素。

13. 在“属性”检查器中，在“实例名称”文本框中键入 **callClip**。

14. 在时间轴中，选择名为 **ActionScript** 的图层上的第 1 帧。

15. 在“动作”面板中，输入以下代码：

```
call("callClip:function1");  
call("callClip:function2");
```

16. 在模拟器中测试应用程序（“控制”>“测试影片”）。

您应在“输出”面板中看到以下文本：

```
function1 was called.  
function2 was called.
```



## 使用 eval() 函数

使用 eval() 函数可以在运行时动态地引用变量和影片剪辑。eval() 函数采用字符串表达式作为参数，并返回由该表达式表示的变量的值，或者返回对影片剪辑的引用。

例如，以下代码计算 name **ActionScript** 变量的值并将结果分配给 nameValue：

```
name = "Jack";
nameValue = eval("name");
// 结果: nameValue = "Jack"
```

由于 **Flash Lite** 不支持本机数组数据结构，因此 eval() 函数通常与 for() 循环和 add（字符串连接）运算符一起使用，来创建由字符串组成的数组。有关更多信息，请参见第 11 页的“模拟数组”。

也可以使用 eval() 来按名称引用影片剪辑实例。例如，假设您有三个影片剪辑，名称分别为 clip1、clip2 和 clip3。以下 for() 循环会使每个剪辑的 x 位置递增 10 像素：

```
for(index = 1; index <= 3; index++) {
    eval("clip" + index)._x += 10
}
```



## 常用撰写脚本任务

本章讨论与用户设备协同工作所需要的常用 **Flash Lite** 撰写脚本任务。例如，这包括获取设备功能信息、启动电话和文本消息以及确定网络状态。

本章包含以下主题：

确定设备和平台功能 .....	19
打开网页 .....	20
启动电话 .....	20
启动文本或多媒体消息 .....	21
启动电子邮件 .....	21
加载外部 <b>SWF</b> 文件 .....	22
加载外部数据 .....	22

### 确定设备和平台功能

**Flash Lite 1.1** 包括了许多 **ActionScript** 变量，这些变量可提供有关特定设备上运行的应用程序可以使用的功能的信息。例如，`_capLoadData` 变量指示设备是否支持加载外部数据，`_capSMS` 变量指示设备是否支持发送 **SMS**（短消息服务）消息。若要查看功能变量的完整列表，请参见 **Flash Lite 1.x ActionScript** 语言参考中的“功能”。

通常，使用功能变量可以在尝试使用某一特定功能前确定设备是否支持该功能。例如，假设您要开发一个使用 `loadVariables()` 函数从 **Web** 服务器下载数据的应用程序。在尝试加载数据前，可以首先检查 `_capLoadData` 变量的值以确定设备是否支持该功能，如下所示：

```
if(_capLoadData == 1) {  
    loadVariables("http://www.macromedia.com/data.txt");  
} else {  
    status_message = "Sorry, unable to load external data."  
}
```

**Flash Lite** 在主 **SWF** 文件的根时间轴上定义功能变量。因此，若要从其它时间轴（例如在影片剪辑的时间轴内部）访问这些变量，需要限定到该变量的路径。例如，以下示例使用斜杠 (/) 来提供到 `_capSMS` 变量的完全限定路径。

```
canSendSMS = /:_capSMS
```

# 打开网页

使用 `getURL()` 命令可以在设备的 **Web** 浏览器中打开一个网页。这与从桌面 **Flash** 应用程序中打开网页的方式相同。例如，以下代码将打开 **Macromedia Web** 页：

```
getURL("http:www.macromedia.com");
```

**Flash Lite** 每帧或每个事件处理函数只处理一个 `getURL()` 动作。某些手持设备将 `getURL()` 动作限制为仅在按钮事件中，在这种情况下，只有在按键事件处理函数内部触发 `getURL()` 调用时，该调用才会得以处理。即使是在这样的情况下，每个按键事件处理函数也只处理一个 `getURL()` 动作。以下代码附加到舞台上的一个按钮实例，当用户在设备上按“选择”按钮时，这段代码会打开一个 **Web** 页：

```
on (keyPress "<Enter>"){  
    getURL("http://www.macromedia.com");  
}
```

# 启动电话

要从 **Flash Lite** 应用程序中启动电话，您可以使用 `getURL()` 函数。通常，使用此函数可以打开一个网页，但在本例中，您要将 `tel:` 指定为协议（代替 `http`），然后提供希望电话拨打的电话号码。调用此函数时，**Flash Lite** 会显示一个确认对话框，询问用户是否想要拨打指定的号码。

以下代码尝试拨打电话 **555-1212**：

```
getURL("tel:555-1212");
```

**Flash Lite** 每帧或每个事件处理函数只处理一个 `getURL()` 动作。某些手持设备将 `getURL()` 动作限制为仅在按钮事件中，在这种情况下，只有在按键事件处理函数内部触发 `getURL()` 调用时，该调用才会得以处理。即使是在这样的情况下，每个按键事件处理函数也只处理一个 `getURL()` 动作。以下示例在用户按设备上的“选择”按钮时启动一个电话：

```
on (keyPress "<Enter>"){  
    getURL("tel:555-1212");  
}
```

## 启动文本或多媒体消息

您可以使用 **Flash Lite** 来启动短消息服务 (SMS) 或多媒体消息服务 (MMS) 消息。要在 **Flash Lite** 应用程序中启动 SMS 或 MMS 消息，您要使用 `getURL()` 命令，为其传递 `sms:` 或 `mms:` 协议代替标准的 `http` 协议，然后传递要向其发送消息的电话号码。

```
getURL("sms:555-1212");
```

您可以选择在 **URL** 查询字符串中指定消息正文，如以下代码所示：

```
getURL("sms:555-1212?body=More info please");
```

若要启动 **MMS** 消息，可以使用 `mms:` 协议代替 `sms:`，如下所示：

```
getURL("mms:555-1212");
```



从 **Flash Lite** 中不可能为 **MMS** 消息指定附件。

**Flash Lite** 每帧或每个事件处理函数只处理一个 `getURL()` 动作。某些手持设备将 `getURL()` 动作限制为仅在按钮事件中，在这种情况下，只有在按键事件处理函数内部触发 `getURL()` 调用时，该调用才会得以处理。即使是在这样的情况下，每个按键事件处理函数也只处理一个 `getURL()` 动作。以下示例在用户按设备上的“选择”按钮时启动 **SMS** 消息：

```
on (keyPress "<Enter>"){  
    getURL("sms:555-1212");  
}
```

## 启动电子邮件

您可以使用 **Flash Lite** 来启动电子邮件消息。要启动电子邮件消息，您要使用 `getURL()` 命令，并为其传递 `mailto:` 协议，后面跟随收件人的电子邮件地址。您可以选择在 **URL** 查询字符串中指定消息的主题和正文，如下所示：

```
getURL("mailto:mobile-developer@macromedia.com?subject=Flash Lite");
```

若要在查询字符串中仅指定消息正文，请使用以下代码：

```
getURL("mailto:mobile-developer@macromedia.com?body=More+info+please");
```

# 加载外部 SWF 文件

使用 `loadMovie()` 函数可以从网络或本地文件加载 SWF 文件。此功能仅在 **Flash Lite 1.1** 及更高版本中可用。以下注意事项在加载外部 SWF 文件时适用：

- **Flash Lite** 可以加载其它 **Flash Lite 1.0** 或 **Flash Lite 1.1** SWF 文件，也可以加载 **Flash 4** 格式或更早版本格式的 SWF 文件。如果尝试加载其它格式的 SWF 文件（如 **Flash Player 6** SWF 文件），**Flash Lite** 将生成一个运行时错误。
- **Flash Lite** 无法直接加载外部图像文件，如 **JPEG** 或 **GIF** 图像。若要加载这些类型的媒体，需要将图像数据转换为 SWF 文件格式。可以通过使用 **Flash** 创作工具，将图像文件导入新文档中，然后将此文档导出为 **Flash Lite** 或 **Flash 4** SWF 文件，来手动完成该转换任务。也可以使用可帮助您自动完成该转换类型的第三方实用程序。

有关加载 SWF 文件的更多信息，请参见《**Flash Lite 1.x** **ActionScript** 语言参考》中的 `loadMovie()`。

# 加载外部数据

若要将外部数据加载到 **Flash Lite** 应用程序中，可以使用 `loadVariables()` 函数。可以通过网络（从某个 **HTTP** 地址）或者从本地文件系统加载数据。此功能仅在 **Flash Lite 1.1** 及更高版本中可用。

本节演示如何使用 `loadVariables()` 函数从外部文件加载数据并在动态文本字段中显示这些数据。首先创建数据文件，这是一个包含五个名称 - 值对（由 **&** 符号分隔）的文本文件。然后创建 **Flash Lite** 应用程序，该应用程序加载并显示文本文件中包含的数据。

此示例假设数据文件和 **SWF** 文件都位于您计算机（当您在模拟器中进行测试时）或设备内存卡（当您在实际设备上进行测试时）上的同一个文件夹中。若要在设备上测试应用程序，必须执行下列操作之一：

- 将该文本文件传输到您的设备上，并将其放在 **SWF** 文件所在的文件夹中。
- 将该文本文件发送到 **Web** 服务器的某个 **URL**（如 `www.your-server.com/data.txt`）上。然后将范例应用程序中的 `loadVariables()` 调用修改为指向该 **URL**，如下所示：  

```
loadVariables("http://www.your-server.com/data.txt", "data_clip");
```

若要查看通过网络加载数据的应用程序的示例，请参见《**Flash** 范例》中的“**Flash Lite** 新闻阅读器”。

若要创建数据文件，请执行以下操作：

1. 使用文本编辑器（如记事本或 SimpleText）创建一个包含以下文本的文件：

```
item_1=Hello&item_2=Bonjour&item_3=Hola&item_4=Buon+giorno&item_5=G'day
```

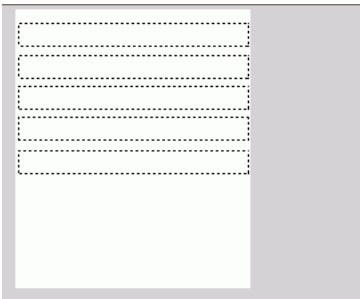
2. 将该文件另存为 **data.txt**。

若要创建加载数据的 Flash Lite 应用程序，请执行以下操作：

1. 利用 Flash Lite 1.1 Symbian Series 60 文档模板创建一个新的文档。

有关使用 Flash Lite 文档模板的更多信息，请参见《Flash Lite 入门》中的“使用 Flash Lite 文档模板（仅限 Flash Professional）”。

2. 将该文件另存为 **dataloading.fla**，并放在刚才创建的文本文件（**data.txt**）所在的文件夹中。
3. 在时间轴中，选择名为 **Content** 的图层的第 1 帧。
4. 使用“文本”工具在舞台上创建五个动态文本字段，如下图所示：



5. 选择第一个（即最上面的）文本字段，然后在属性检查器的“变量”文本框中键入 **item\_1**。  
此变量名称对应于您刚才创建的 **data.txt** 文件中定义的第一个变量的名称（**item\_1=Hello**）。
6. 按照前两个步骤中所描述的相同的方式，为剩余四个文本字段分别赋予变量名称 **item\_2**、**item\_3**、**item\_4** 和 **item\_5**。
7. 按住 **Shift** 键并单击以选择每个文本字段，直到选中所有文本字段，然后选择“修改”>“转换为元件”。
8. 在“转换为元件”对话框中，选择“影片剪辑”作为元件类型，然后单击“确定”。
9. 选择刚才创建的影片剪辑，然后在属性检查器的“实例名称”文本框中键入 **data\_clip**。
10. 在时间轴中，选择“动作”图层的第 1 帧，然后打开“动作”面板（“窗口”>“动作”）。

11. 在“动作”面板中键入以下代码：

```
loadVariables("data.txt", "data_clip");
```

12. 保存您的更改（“文件” > “保存”），然后在模拟器中测试应用程序（“控制” > “测试影片”）。

您应该可以看到每个文本字段都已使用文本文件中的数据进行了填充，如下图所示：





# 索引

## 英文

call() 函数, 使用 13

eval() 函数, 使用 17

Flash Lite 1.x ActionScript

1.0 和 1.1 之间的差异 6

不可用的功能 7

不受支持的 Flash 4 ActionScript 6

概述 5

getURL() 函数

打开 Web 页 20

启动电话 20

启动电子邮件 21

启动多媒体消息 21

启动文本消息 21

loadMovie() 函数, 使用 22

loadVariables() 函数, 使用 22

Web 页, 打开 20

## B

变量

点语法和斜杠语法 10

动态引用 17

引用 10

## D

打开 Web 页 20

电话, 启动 20

电子邮件, 启动 21

多媒体消息, 启动 21

## G

滚动文本, 创建 12

## H

函数, 使用 call() 进行模拟 13

函数剪辑, 创建 13

## J

加载外部 SWF 文件 22

加载外部数据 22

## L

连接字符串 12

## P

平台功能变量, 关于 19

## Q

启动电话 20

启动消息 21

## S

时间轴, 使用 ActionScript 进行控制 10

数组, 使用字符串进行模拟 11

## W

文本消息, 启动 21

## Y

影片剪辑

    动态引用 17

    获取和设置属性 9

## Z

字符串, 连接 12