

ADOBE® FLASH® LITE™ 1.x

Adobe® ActionScript® 언어 참조 설명서

© 2008 Adobe Systems Incorporated. All rights reserved.

Adobe® Flash® Lite™ 1.x ActionScript™ 언어 참조 설명서

본 안내서가 최종 사용자 사용권 계약서가 포함된 소프트웨어와 함께 배포되는 경우, 본 안내서 및 설명된 소프트웨어는 사용권 하에서 제공되며 해당 사용권 약관에 따라 서만 사용하거나 복제할 수 있습니다. 사용권 계약에 의해 허용된 경우를 제외하고는 이 안내서의 어떠한 부분도 Adobe Systems Incorporated의 사전 서면 승인 없이 전자적, 기계적, 기록 또는 그 밖의 다른 형태나 수단으로도 복제하거나, 검색 시스템에 저장하거나, 전송할 수 없습니다. 이 안내서가 최종 사용자 사용권 계약서가 포함된 소프트웨어와 함께 배포되지 않는 경우에도 안내서 내용은 저작권법의 보호를 받습니다.

이 안내서의 내용은 오직 정보 제공만을 위한 것으로 예고 없이 변경될 수 있으며, Adobe Systems Incorporated의 공약으로 해석해서는 안 됩니다. Adobe Systems Incorporated는 이 안내서에 있을 수 있는 정보의 오류나 부정확성에 대해 어떠한 책임이나 의무도 없습니다.

이 안내서에는 Adobe Systems Incorporated가 관리하지 않는 타사 웹 사이트 링크가 포함되어 있지만 Adobe Systems Incorporated는 링크되는 사이트의 내용에 대해 책임이 없습니다. 이 안내서에 언급된 타사 웹 사이트를 방문하는 동안 발생하는 문제는 귀하의 책임입니다. Adobe Systems Incorporated는 이 링크를 오직 사용자 편의를 위해서만 제공하며 이 링크를 포함하는 것이 Adobe Systems Incorporated가 해당 타사 사이트의 내용에 대한 책임을 시인하거나 수용하는 것을 의미하지는 않습니다.

프로젝트에 포함하려는 기존 아트웍이나 이미지는 저작권법에 의해 보호되고 있을 수 있다는 점에 유의하십시오. 이러한 자료를 무단으로 새 작업에 포함시킬 경우 저작권 소유자의 권리를 침해할 수 있습니다. 저작권 소유자로부터 필요한 권한을 부여받으십시오.

예제 템플릿에 인용된 회사명은 데모용으로만 사용되고 실제 조적을 의미하지는 않습니다.

Adobe, the Adobe logo, ColdFusion, and Flash are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Symbian and all Symbian based marks and logos are trademarks of Symbian Limited. All other trademarks are the property of their respective owners.

Sorenson Spark™ video compression and decompression technology licensed from Sorenson Media, Inc.

MPEG Layer-3 audio compression technology licensed by Fraunhofer IIS and Thomson Multimedia (<http://www.iis.fhg.de/amm/>).

Portions licensed from Nellymoser, Inc. (www.nellymoser.com).

Adobe Flash 9.2 video is powered by On2 TrueMotion video technology. © 1992-2005 On2 Technologies, Inc. All Rights Reserved. <http://www.on2.com>.

Updated Information/Additional Third Party Code Information available at <http://www.adobe.com/go/thirdparty/>.

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

Notice to U.S. Government End Users. The Software and Documentation are "Commercial Items," as that term is defined at 48 C.F.R. §2.101, consisting of "Commercial Computer Software" and "Commercial Computer Software Documentation," as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions herein. Unpublished-rights reserved under the copyright laws of the United States. Adobe Systems Incorporated, 345 Park Avenue, San Jose, CA 95110-2704, USA. For U.S. Government End Users, Adobe agrees to comply with all applicable equal opportunity laws including, if appropriate, the provisions of Executive Order 11246, as amended, Section 402 of the Vietnam Era Veterans Readjustment Assistance Act of 1974 (38 USC 4212), and Section 503 of the Rehabilitation Act of 1973, as amended, and the regulations at 41 CFR Parts 60-1 through 60-60, 60-250, and 60-741. The affirmative action clause and regulations contained in the preceding sentence shall be incorporated by reference.

목차

1장: 소개

Samples 폴더	1
인쇄 규칙	1

2장: Flash Lite 전역 함수

call()	3
chr()	4
duplicateMovieClip()	4
eval()	5
getProperty()	6
getTimer()	6
getURL()	7
gotoAndPlay()	9
gotoAndStop()	9
ifFrameLoaded()	10
int()	11
length()	11
loadMovie()	12
loadMovieNum()	13
loadVariables()	13
loadVariablesNum()	14
mbchr()	15
mblength()	16
mbord()	16
mbsubstring()	17
nextFrame()	17
nextScene()	18
Number()	18
on()	19
ord()	20
play()	20
prevFrame()	21
prevScene()	21
random()	22
removeMovieClip()	22
set()	23
setProperty()	23
stop()	24
stopAllSounds()	24

String()25
 substring()26
 tellTarget()26
 toggleHighQuality()27
 trace()27
 unloadMovie()28
 unloadMovieNum()28

3장: Flash Lite 속성

/ (슬래시)31
 _alpha31
 _currentframe32
 _focusrect32
 _framesloaded33
 _height33
 _highquality34
 _level34
 maxscroll35
 _name35
 _rotation36
 scroll36
 _target37
 _totalframes37
 _visible37
 _width38
 _x38
 _xscale39
 _y39
 _yscale40

4장: Flash Lite 명령문

break41
 case42
 continue43
 do..while44
 else45
 else if45
 for46
 if47
 switch47
 while48

5장: Flash Lite 연산자

add(문자열 결합 연산자) 52

+=(더하기 대입) 52

및 53

=(대입) 53

/*(블록 주석 연산자) 54

,(쉼표) 54

// (comment) 55

?:(조건부) 56

-- (감소 연산자) 56

/ (나누기 연산자) 57

/=(나누기 대입) 57

.(도트) 58

++(증가) 59

&&(논리 AND) 59

!(논리 NOT) 60

||(논리 OR) 61

%(모듈러스) 61

%=(모듈러스 대입) 62

*=(곱하기 대입) 62

*(곱하기 연산자) 63

+(숫자 더하기 연산자) 64

==(숫자 항등 연산자) 64

> (보다 큼 숫자 연산자) 65

>=(보다 크거나 같음 숫자 연산자) 65

<>(비항등 숫자 연산자) 66

< (보다 작음 숫자) 66

<=(보다 작거나 같음 숫자 연산자) 67

()(괄호) 68

""(문자열 구분 기호 연산자) 68

eq(문자열 항등 연산자) 69

gt(보다 큼 문자열 연산자) 69

ge(보다 크거나 같음 문자열 연산자) 70

ne(문자열 비항등 연산자) 71

lt(보다 작음 문자열 연산자) 71

le(보다 작거나 같음 문자열 연산자) 72

-(빼기 연산자) 73

=(빼기 대입) 74

6장: Flash Lite 특정 언어 요소

기능 77

_capCompoundSound 77

_capEmail	77
_capLoadData	78
_capMFi	78
_capMIDI	79
_capMMS	79
_capMP3	80
_capSMAF	80
_capSMS	81
_capStreamSound	81
_cap4WayKeyAS	82
\$version	83
fscommand()	83
Launch	83
fscommand2()	84
Escape	85
FullScreen	85
GetBatteryLevel	86
GetDateDay	86
GetDateMonth	87
GetDateWeekday	87
GetDateYear	88
GetDevice	89
GetDeviceID	90
GetFreePlayerMemory	91
GetLanguage	91
GetLocaleLongDate	94
GetLocaleShortDate	94
GetLocaleTime	95
GetMaxBatteryLevel	96
GetMaxSignalLevel	96
GetMaxVolumeLevel	96
GetNetworkConnectStatus	97
GetNetworkName	98
GetNetworkRequestStatus	98
GetNetworkStatus	100
GetPlatform	101
GetPowerSource	102
GetSignalLevel	102
GetTimeHours	103
GetTimeMinutes	103
GetTimeSeconds	104
GetTimeZoneOffset	104

GetTotalPlayerMemory	105
GetVolumeLevel	105
Quit	106
ResetSoftKeys	106
SetInputTextType	107
SetQuality	107
SetSoftKeys	108
StartVibrate	108
StopVibrate	109
Unescape	109
색인	111

1장: 소개

이 설명서에서는 Flash Lite 1.x로 통칭되는 Adobe Macromedia® Flash® Lite™ 1.0 소프트웨어 및 Adobe Macromedia® Flash® Lite™ 1.1 소프트웨어의 응용 프로그램 개발에 사용하는 ActionScript 요소의 구문 및 사용에 대해 설명합니다. Flash Lite 1.x ActionScript는 Macromedia® Flash® 4에서 사용된 ActionScript 버전을 기반으로 합니다. 여기에 나와 있는 예제를 스크립트에 사용하려면 이 설명서의 코드 예제를 복사하여 스크립트 창이나 외부 스크립트 파일에 붙여넣으면 됩니다. 이 설명서에는 연산자, 키워드, 명령문, 명령, 속성, 함수, 클래스 및 메서드 등의 모든 ActionScript 요소가 포함되어 있습니다.

Samples 폴더

작동되는 ActionScript 코드를 가진 전체 Flash Lite 프로젝트의 샘플은 www.adobe.com/go/learn_ftl_samples_and_tutorials의 Flash Lite 샘플 및 자습서 페이지를 참조하십시오. 해당 ActionScript 버전을 찾아 .zip 파일을 다운로드 및 압축 해제한 다음 Samples 폴더로 이동하여 샘플 파일에 액세스하십시오.

인쇄 규칙

이 설명서에는 다음과 같은 인쇄 규칙이 사용됩니다.

- 기울임 글꼴은 대체해야 하는 값을 나타냅니다(예: 폴더 경로).
- 코드 글꼴은 ActionScript 코드를 나타냅니다.
- 코드 기울임 글꼴은 ActionScript 매개 변수를 나타냅니다.
- 짧은 글꼴은 입력 항목을 나타냅니다.
- 코드 예제의 큰따옴표(“)는 구분된 문자열을 나타냅니다. 그러나 프로그래머는 작은따옴표도 사용할 수 있습니다.

2장: Flash Lite 전역 함수

이 단원에서는 Adobe Macromedia Flash Lite 1.1 소프트웨어의 ActionScript 전역 함수 구문과 사용에 대해 설명합니다. 설명할 함수는 다음과 같습니다.

Function	설명
call()	재생 헤드를 호출된 프레임으로 이동시키지 않고 이 프레임의 스크립트를 실행합니다.
chr()	ASCII 코드 번호를 문자로 변환합니다.
duplicateMovieClip()	SWF 파일이 재생되는 동안 동영상 클립의 인스턴스를 작성합니다.
eval()	이름으로 변수, 속성, 객체 또는 동영상 클립에 액세스합니다.
getProperty()	지정된 동영상 클립에 대해 지정된 속성 값을 반환합니다.
getTimer()	SWF 파일이 재생되기 시작한 이후에 경과한 시간을 밀리초 단위로 반환합니다.
getURL()	특정 URL의 문서를 임의의 윈도우로 로드하거나 정의된 URL의 다른 응용 프로그램에 변수를 전달합니다.
gotoAndPlay()	지정된 장면 프레임으로 재생 헤드를 보내어 해당 프레임에서 재생을 시작합니다. 지정된 장면이 없으면 현재 장면의 지정된 프레임으로 재생 헤드가 이동합니다.
gotoAndStop()	지정된 장면 프레임으로 재생 헤드를 보낸 다음 중지합니다. 지정된 장면이 없으면 현재 장면의 프레임으로 재생 헤드가 보내집니다.
iframeLoaded()	특정 프레임의 콘텐츠를 로컬로 사용할 수 있는지 여부를 확인합니다.
int()	10진수를 정수 값으로 자릅니다.
length()	지정된 문자열 또는 변수 이름의 문자 수를 반환합니다.
loadMovie()	원본 SWF 파일이 재생되는 동안 SWF 파일을 Flash Lite로 로드합니다.
loadMovieNum()	처음에 로드된 SWF 파일이 재생되는 동안 SWF 파일을 Flash Lite의 레벨로 로드합니다.
loadVariables()	Adobe ColdFusion®, CGI, ASP, PHP 또는 Perl 스크립트로 만든 텍스트 파일이나 텍스트와 같은 외부 파일에서 데이터를 읽어 들인 다음 Flash Player 레벨에서 변수 값을 설정합니다. 이 함수는 또한 활성 SWF 파일의 변수를 새 값으로 업데이트할 수 있습니다.
loadVariablesNum()	ColdFusion, CGI, ASP, PHP 또는 Perl 스크립트로 만든 텍스트 파일이나 텍스트와 같은 외부 파일에서 데이터를 읽어 들인 다음 Flash Player 레벨에서 변수 값을 설정합니다. 이 함수는 또한 활성 SWF 파일의 변수를 새 값으로 업데이트할 수 있습니다.
mbchr()	ASCII 코드 숫자를 멀티바이트 문자로 변환합니다.
mblength()	멀티바이트 문자열의 길이를 반환합니다.
mbord()	지정된 문자를 멀티바이트 숫자로 변환합니다.
mbsubstring()	멀티바이트 문자열에서 새 멀티바이트 문자열을 추출합니다.
nextFrame()	재생 헤드를 다음 프레임으로 보내고 중지합니다.
nextScene()	재생 헤드를 다음 장면의 프레임 1로 보내고 중지합니다.
Number()	표현식을 숫자로 변환하여 값을 반환합니다.
on()	이벤트를 트리거하는 사용자 이벤트 또는 키 누르기를 지정합니다.
ord()	문자를 ASCII 코드 번호로 변환합니다.

Function	설명
<code>play()</code>	타임라인에서 재생 헤드를 앞쪽으로 이동합니다.
<code>prevFrame()</code>	재생 헤드를 이전 프레임으로 보낸 다음 중지합니다. 현재 프레임이 1이면 재생 헤드가 이동하지 않습니다.
<code>prevScene()</code>	재생 헤드를 이전 장면의 프레임 1로 보내고 중지합니다.
22페이지의 " <code>random()</code> "	난수를 반환합니다.
<code>removeMovieClip()</code>	<code>duplicateMovieClip()</code> 을 사용하여 처음에 만들어진 지정된 동영상 클립을 삭제합니다.
<code>set()</code>	변수에 값을 할당합니다.
<code>setProperty()</code>	동영상이 재생될 때 동영상 클립의 속성 값을 변경합니다.
<code>stop()</code>	현재 재생 중인 SWF 파일을 중지합니다.
<code>stopAllSounds()</code>	재생 헤드를 중지하지 않고 SWF 파일에서 현재 재생되고 있는 모든 사운드를 중지합니다.
<code>String()</code>	지정된 매개 변수의 문자열 표현을 반환합니다.
<code>substring()</code>	문자열의 일부를 추출합니다.
<code>tellTarget()</code>	<i>statement(s)</i> 매개 변수에 지정된 명령어를 <i>target</i> 매개 변수에 지정된 타임라인에 적용합니다.
<code>toggleHighQuality()</code>	Flash Lite에서 엔티앨리어싱을 켜고 끕니다. 엔티앨리어싱은 객체의 가장자리를 매끄럽게 하지만 SWF 파일의 재생을 느리게 합니다.
<code>trace()</code>	표현식을 평가하여 그 결과를 테스트 모드의 출력 패널에 표시합니다.
<code>unloadMovie()</code>	<code>loadMovie()</code> <code>loadMovieNum()</code> 또는 <code>duplicateMovieClip()</code> 을 사용하여 로드한 Flash Lite의 동영상 클립을 제거합니다.
<code>unloadMovieNum()</code>	<code>loadMovie()</code> <code>loadMovieNum()</code> 또는 <code>duplicateMovieClip()</code> 을 사용하여 로드한 동영상을 Flash Lite의 레벨에서 제거합니다.

call()

지원 장치
Flash Lite 1.0.

구문

`call (frame)`

`call (movieClipInstance: frame)`

피연산자

프레임 타임라인에 포함된 프레임의 번호 또는 레이블입니다.

`movieClipInstance` 동영상 클립의 인스턴스 이름입니다.

설명

재생 헤드를 호출된 프레임으로 이동시키지 않고 이 프레임의 스크립트를 실행하는 함수입니다. 스크립트가 실행된 후에는 지역 변수가 존재하지 않습니다. `call()` 함수에는 두 가지 가능한 양식이 있습니다.

- 기본 양식은 재생 헤드를 해당 프레임으로 이동하지 않고 `call()` 함수가 실행되는 동일한 타임라인의 지정된 프레임에서 스크립트를 실행합니다.

- 지정된 클립 인스턴스 양식은 재생 헤드를 해당 프레임으로 이동하지 않고 동영상 클립 인스턴스의 지정된 프레임에서 스크립트를 실행합니다.

참고: call() 함수는 동시에 작동하며 call() 함수 이후의 ActionScript는 지정된 프레임에서 모든 ActionScript가 완료될 때까지 실행되지 않습니다.

예제

다음은 myScript 프레임에서 스크립트를 실행하는 예제입니다.

```
// to execute functions in frame with label "myScript"
thisFrame = "myScript";
trace ("Calling the script in frame: " + thisFrame);

// to execute functions in any other frame on the same timeline
call("myScript");
```

chr()

지원 장치

Flash Lite 1.0.

구문

```
chr(number)
```

피연산자

number ASCII 코드 숫자입니다.

설명

ASCII 코드 숫자를 문자로 변환하는 문자열 함수입니다.

예제

다음은 숫자 65를 문자 A로 변환한 후 변수 myVar에 대입하는 예제입니다.

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

duplicateMovieClip()

지원 장치

Flash Lite 1.0.

구문

```
duplicateMovieClip(target, newname, depth)
```

피연산자

target 복제하려는 동영상 클립의 대상 경로입니다.

newname 복제된 동영상 클립의 고유한 식별자입니다.

심도 복제된 동영상 클립의 고유한 심도 수준입니다. 심도 레벨이란 복제된 동영상 클립이 쌓이는 순서를 말합니다. 심도 레벨은 타임라인에서 레이어가 쌓이는 순서와 매우 비슷합니다. 심도 레벨이 낮은 동영상 클립은 심도 레벨이 높은 클립 아래에 숨겨집니다. 이미 사용 중인 심도 레벨에 기존 동영상 클립을 덮어쓰지 않으려면 복제된 각 동영상 클립에 고유한 심도 레벨을 지정해야 합니다.

설명

SWF 파일이 재생되는 동안 동영상 클립의 인스턴스를 작성하는 함수입니다. 아무 것도 반환하지 않습니다. 복제된 동영상 클립의 재생 헤드는 원본(부모) 동영상 클립에서 재생 헤드의 위치와 상관없이 항상 프레임 1에서 시작합니다. 부모 동영상 클립의 번수는 복제된 동영상 클립으로 복사되지 않습니다. 부모 동영상 클립이 삭제되면 복제된 동영상 클립도 삭제됩니다.

`duplicateMovieClip()`을 사용하여 만든 동영상 클립 인스턴스를 삭제하려면 `removeMovieClip()` 함수 또는 메서드를 사용합니다. **newname** 피연산자 처럼 전달된 문자열을 사용하여 새로운 동영상 클립을 참조합니다.

예제

다음 예제는 `originalClip`이라는 동영상 클립을 복제해서 심도 10의 새로운 동영상 클립인 `newClip`을 만듭니다. 새로운 클립의 `x` 위치는 100 픽셀로 설정됩니다.

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

다음 예제에서는 `for` 루프에서 `duplicateMovieClip()`을 사용하여 여러 개의 새 동영상 클립을 한 번에 만듭니다. 인덱스 변수는 점유된 최상위 쌓기 심도를 추적합니다. 각 복제된 동영상 클립 이름에는 쌓기 심도에 해당하는 숫자 접미어(`clip1`, `clip2`, `clip3`)가 포함됩니다.

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

참고 사항

[removeMovieClip\(\)](#)

eval()

지원 장치

Flash Lite 1.0.

구문

```
eval(expression)
```

피연산자

expression 검색하려는 변수, 속성, 객체 또는 동영상 클립의 이름이 포함된 문자열입니다.

설명

이름으로 변수, 속성, 객체 또는 동영상 클립에 액세스하는 함수입니다. **expression**이 변수 또는 속성인 경우 변수 또는 속성 값이 반환됩니다. **expression**이 객체 또는 동영상 클립인 경우 객체 또는 동영상 클립에 대한 참조가 반환됩니다. **expression**에 지정된 요소를 찾을 수 없는 경우 `undefined`가 반환됩니다.

`eval()`을 사용하여 배열을 시뮬레이션하거나 변수 값을 동적으로 설정하고 검색할 수 있습니다.

예제

다음 예제에서는 eval() 함수를 사용하여 표현식 "piece" + x의 값을 결정합니다. 그 결과가 변수 이름 piece3이 되므로 eval() 함수에서는 변수 값을 반환한 후 이 값을 y에 지정합니다.

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

다음 예제는 배열을 시뮬레이션하는 방법을 보여 줍니다.

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

getProperty()

지원 장치

Flash Lite 1.0.

구문

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

피연산자

my_mc 속성을 검색할 동영상 클립의 인스턴스 이름입니다.

속성 동영상 클립의 속성입니다.

설명

동영상 클립 my_mc의 지정된 속성 값을 반환하는 함수입니다.

예제

다음은 루트 동영상 타임라인의 my_mc 동영상 클립의 수평 축 좌표(_x)를 검색하는 예제입니다.

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

참고 사항

[setProperty\(\)](#)

getTimer()

지원 장치

Flash Lite 1.0.

구문

```
getTimer()
```

피연산자

없음

설명

SWF 파일이 재생된 후 경과한 시간을 밀리초 단위로 반환하는 함수입니다.

예제다음은 SWF 파일이 재생되기 시작한 이후에 경과된 시간을 밀리초 단위로 나타낸 값으로 `timeElapsed` 변수를 설정하는 예제입니다.

```
timeElapsed = getTimer();
trace (timeElapsed); // Output: milliseconds of time movie has been playing
```

getURL()

지원 장치

Flash Lite 1.0.

구문

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

피연산자`url` 문서를 가져올 URL입니다.

창 대신 웹 브라우저의 웹 페이지에서 연결된 PDF를 불러옵니다 해당 문서를 로드할 윈도우 또는 HTML 프레임을 지정하는 매개 변수 선택 사항입니다.

참고: 휴대 전화의 브라우저는 여러 창을 지원하지 않으므로 *window* 매개 변수를 Flash Lite 응용 프로그램에 지정할 수 없습니다.

빈 문자열 또는 특정 윈도우 이름을 직접 입력할 수도 있고 그 다음에 예약된 대상 이름 중에서 선택할 수 있습니다.

- `_self`는 현재 윈도우의 현재 프레임을 지정합니다.
- `_blank`는 새 윈도우를 지정합니다.
- `_parent`는 현재 프레임의 부모를 지정합니다.
- `_top`은 현재 윈도우의 최상위 프레임을 지정합니다.

변수 변수를 전송하는 GET 또는 POST 메서드입니다. 변수가 없으면 이 매개 변수를 생략합니다. GET 메서드는 URL 끝에 변수를 추가하며 변수의 수가 적은 경우 사용됩니다. POST 메서드는 별도의 HTTP 헤더에 있는 변수를 보내며 문자열이 긴 변수를 보내는 데 사용됩니다.

설명특정 URL에 있는 문서를 임의의 윈도우로 로드하거나 정의된 URL의 다른 응용 프로그램에 변수를 전달하는 함수입니다. 이 함수를 테스트하기 위해서는 로드할 파일이 지정된 위치에 있어야 합니다. 절대 URL(예: `http://www.myserver.com`)을 사용하면 네트워크에 연결되어 있어야 합니다.

Flash Lite 1.0은 HTTP, HTTPS, mailto 및 tel 프로토콜만 인식합니다. Flash Lite 1.1은 이러한 프로토콜 외에도 파일, SMS(Short Message Service) 및 MMS(Multimedia Message Service) 프로토콜을 인식합니다.

Flash Lite는 운영 체제에 호출을 전달하고 운영 체제는 지정된 프로토콜에 등록된 기본 응용 프로그램을 사용하여 호출을 처리합니다.

각 프레임 또는 이벤트 핸들러마다 단 하나의 `getURL()` 함수만 처리할 수 있습니다.

특정 핸드셋트는 `getURL()` 함수를 키 이벤트에만 제한하며, 이 경우에는 `getURL()` 호출이 키 이벤트 핸들러에서 트리거되는 경우에만 처리됩니다. 이러한 상황에서도 이벤트 핸들러마다 하나의 `getURL()` 함수만 처리됩니다.

예제

다음 ActionScript에서는 Flash Lite 플레이어가 기본 브라우저에서 `mobile.example.com`을 엽니다.

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

또한 GET 또는 POST를 사용하여 현재 타임라인에서 변수를 보낼 수 있습니다. 다음은 GET 메서드를 사용하여 URL에 변수를 추가하는 예제입니다.

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

다음 ActionScript는 POST를 사용하여 HTTP 헤더에 변수를 보냅니다.

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

`address`, `subject` 및 `body` 텍스트 필드가 이미 채워진 전자 메일 작성 창을 여는 단추 함수를 지정할 수 있습니다. 단추 함수를 지정하려면 다음 메서드 중 하나를 사용합니다. 메서드 1은 Shift-JIS 또는 영어 문자 인코딩을 지원하며, 메서드 2는 영어 문자 인코딩만 지원합니다.

메서드 1: 이 예제에서와 같이 원하는 각 매개 변수에 대해 변수를 설정합니다.

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

메서드 2: 이 예제에서와 같이 `getURL()` 함수 내에서 각 매개 변수를 정의합니다.

```
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");
}
```

메서드 2에서는 문자열의 공백을 보존하는 반면 메서드 1에서는 자동 URL 인코딩으로 처리합니다. 예를 들어, 메서드 1을 사용한 경우 나타나는 문자열은 다음과 같습니다.

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

반면, 메서드 2의 결과는 다음과 같은 문자열로 나타납니다.

```
email subject
email body
```

다음은 tel 프로토콜을 사용하는 예제입니다.

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

다음 예제에서는 `getURL()`을 사용하여 SMS 메시지를 보냅니다.

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

다음 예제에서는 `getURL()`을 사용하여 MMS 메시지를 보냅니다.

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

다음 예제에서는 `getURL()`을 사용하여 로컬 파일 시스템에 저장된 텍스트 파일을 엽니다.

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

gotoAndPlay()

지원 장치

Flash Lite 1.0.

구문

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

피연산자

scene 재생 헤드를 보낼 장면의 이름을 지정하는 문자열입니다. 이 매개 변수는 선택 사항입니다.

프레임 번호를 나타내는 숫자 또는 재생 헤드를 보낼 프레임의 레이블을 나타내는 문자열입니다.

설명

지정된 장면 프레임으로 재생 헤드를 보내어 해당 프레임에서 재생을 시작하는 함수입니다. 지정된 장면이 없으면 현재 장면의 지정된 프레임으로 재생 헤드가 이동합니다.

scene 매개 변수는 동영상 클립 또는 문서의 다른 객체의 타임라인이 아니라 루트 타임라인에서만 사용할 수 있습니다.

예제

다음 예제에서는 `gotoAndPlay()`가 지정된 단추를 클릭하면 재생 헤드가 현재 장면의 프레임 16으로 보내지고 SWF 파일의 재생이 시작됩니다.

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

gotoAndStop()

지원 장치

Flash 1.0

구문

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

피연산자

scene 재생 헤드를 보낼 장면의 이름을 지정하는 문자열입니다. 이 매개 변수는 선택 사항입니다.

프레임 프레임 번호를 나타내는 숫자 또는 재생 헤드를 보낼 프레임의 레이블을 나타내는 문자열입니다.

설명

지정된 장면 프레임으로 재생 헤드를 보낸 다음 중지하는 함수입니다. 지정된 장면이 없으면 현재 장면의 프레임으로 재생 헤드가 보내집니다.

scene 매개 변수는 동영상 클립 또는 문서의 다른 객체의 타임라인이 아니라 루트 타임라인에서만 사용할 수 있습니다.

예제

다음 예제에서는 사용자가 `gotoAndStop()`이 지정된 단추를 클릭하면 재생 헤드가 현재 장면의 프레임 5로 보내지고 SWF 파일이 중지됩니다.

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

ifFrameLoaded()

지원 장치

Flash Lite 1.0.

구문

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

피연산자

scene 로드되어야 할 장면의 이름을 지정하는 문자열입니다. 이 매개 변수는 선택 사항입니다.

프레임 다음 명령문이 실행되기 전에 로드되어야 할 프레임 번호나 프레임 레이블입니다.

statement(s) 지정된 프레임 또는 장면과 프레임이 로드되는 경우 실행할 명령입니다.

설명

특정 프레임의 콘텐츠를 로컬로 사용할 수 있는지 여부를 확인하는 함수입니다. `ifFrameLoaded`를 사용하면 로컬 컴퓨터로 나머지 SWF 파일을 다운로드하는 동안 간단한 애니메이션을 재생할 수 있습니다. `_framesloaded` 속성을 사용하여 외부 SWF 파일의 다운로드 진행률을 확인할 수도 있습니다. `_framesloaded`를 사용하는 경우와 `ifFrameLoaded`를 사용하는 경우의 차이점은 `_framesloaded`를 사용하면 사용자 정의 if 문이나 else 문을 추가할 수 있다는 점입니다.

예제

다음은 `ifFrameLoaded` 함수를 사용하여 SWF 파일의 프레임 10이 로드되는지 확인하는 예제입니다. 프레임이 로드되는 경우 `trace()` 명령은 출력 패널에 "frame number 10 is loaded"를 인쇄합니다. 또한 출력 변수는 `fame loaded: 10`의 변수로 정의됩니다.

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

참고 사항

[_framesloaded](#)

int()

지원 장치

Flash Lite 1.0.

구문

```
int (value)
```

피연산자

value 정수 값으로 잘리는 수 또는 문자열입니다.

설명

10진수를 정수 값으로 자르는 함수입니다.

예제

다음은 **distance** 및 **myDistance** 변수에 저장된 숫자를 자르는 예제입니다.

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

length()

지원 장치

Flash Lite 1.0.

구문

```
length (expression)
```

```
length (variable)
```

피연산자

expression 문자열입니다.

variable 변수의 이름입니다.

설명

지정된 문자열이나 변수 이름의 문자 수를 반환하는 문자열 함수입니다.

예제

다음은 "Hello" 문자열의 길이를 반환하는 예제입니다.

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

반환값은 5입니다.

다음은 최소 6자를 포함하는지 확인하여 전자 메일 주소의 유효성을 검사하는 예제입니다.

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

loadMovie()

지원 장치

Flash Lite 1.1.

구문

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

피연산자

url 로드할 SWF 파일의 절대 또는 상대 URL을 지정하는 문자열입니다. 상대 경로는 수준 0에서 SWF 파일에 상대적이어야 하고 절대 URL에는 `http://` 또는 `file:///`과 같은 프로토콜 참조가 포함되어야 합니다.

target 동영상 클립에 대한 참조이거나 **target** 동영상 클립에 대한 경로를 표시하는 문자열입니다. 대상 동영상 클립은 로드한 SWF 파일로 대체됩니다.

method 변수를 전달하는 데 사용할 HTTP 메서드를 지정하는 선택적 문자열 매개 변수입니다. 매개 변수는 문자열 GET 또는 POST여야 합니다. 전달할 변수가 없으면 이 매개 변수를 생략합니다. GET 메서드는 URL 끝에 변수를 추가하며 변수의 수가 적은 경우 사용됩니다. POST 메서드는 별도의 HTTP 헤더에 있는 변수를 보내며 문자열이 긴 변수에 사용됩니다.

설명

원본 SWF 파일이 재생되는 동안 Flash Lite에 SWF 파일을 로드하는 함수입니다.

특정 레벨로 SWF 파일을 로드하려면 `loadMovie()` 대신 `loadMovieNum()` 함수를 사용합니다.

SWF 파일이 **target** 동영상 클립에 로드되면 해당 동영상 클립의 대상 경로를 사용하여 로드된 SWF 파일의 대상을 지정할 수 있습니다. 대상 동영상 클립에 로드된 SWF 파일은 대상 동영상 클립의 위치, 회전 및 크기 조절 속성을 상속합니다. 로드된 이미지나 SWF 파일의 왼쪽 위 모서리는 대상 동영상 클립의 등록 포인트에 맞춰 정렬됩니다. 그러나 대상이 루트 타임라인인 경우에는 이미지나 SWF 파일의 왼쪽 위 모서리가 스테이지의 왼쪽 위 모서리에 맞춰 정렬됩니다.

`loadMovie()`에 의해 로드된 SWF 파일을 `unloadMovie()`를 사용하여 제거합니다.

예제

다음 예제에서는 같은 디렉토리에 있는 SWF 파일인 `circle.swf`를 로드하여 스테이지에 이미 있는 동영상 클립인 `mySquare`를 대체합니다.

```
loadMovie("circle.swf", "mySquare");
// Equivalent statement: loadMovie("circle.swf", _level0.mySquare);
```

참고 사항

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

loadMovieNum()

지원 장치

Flash Lite 1.1.

구문

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

피연산자

url 로드할 SWF 파일의 절대 또는 상대 URL을 지정하는 문자열입니다. 상대 경로는 레벨 0에 있는 SWF 파일에 상대적이어야 합니다. 독립형 Flash Lite 플레이어에서 사용되거나 Flash 제작 응용 프로그램에서 테스트 모드로 사용하기 위해서는 모든 SWF 파일이 같은 폴더에 있어야 하며 파일 이름에 폴더나 드라이브 이름을 포함할 수 없습니다.

level SWF 파일이 로드되는 Flash Lite의 레벨을 지정하는 정수입니다.

method 변수를 전달하는 데 사용할 HTTP 메서드를 지정하는 선택적 문자열 매개 변수입니다. 이 매개 변수는 값으로 GET 또는 POST를 포함해야 합니다. 전달할 변수가 없으면 이 매개 변수를 생략합니다. GET 메서드는 URL 끝에 변수를 추가하며 변수의 수가 적은 경우 사용됩니다. POST 메서드는 별도의 HTTP 헤더에 있는 변수를 보내며 문자열이 긴 변수에 사용됩니다.

설명

처음에 로드한 SWF 파일이 재생되는 동안 임의의 Flash Lite 레벨에 SWF 파일을 로드하는 함수입니다.

일반적으로 Flash Lite는 단일 SWF 파일을 표시한 다음 종료됩니다. loadMovieNum() 함수를 사용하면 여러 SWF 파일을 동시에 표시할 수 있으며, 다른 HTML 문서를 로드하지 않고도 SWF 파일 사이를 전환할 수 있습니다.

레벨 대신 대상을 지정하려면 loadMovieNum() 대신 loadMovie() 함수를 사용합니다.

Flash Lite에서 레벨은 레벨 0부터 차례로 순서가 매겨지는 계층 구조를 이룹니다. 이 레벨들은 투명 플라스틱으로 만든 층과 같습니다. 즉, 각 객체는 해당 레벨이 아닌 경우에는 모두 투명하게 처리됩니다. loadMovieNum()을 사용할 때에는 SWF 파일이 로드될 임의의 Flash Lite 레벨을 지정해야 합니다. 일단 SWF 파일이 임의의 레벨로 로드되면 _levelN 구문을 사용할 수 있습니다. 여기서 N은 대상 SWF 파일의 레벨 번호를 의미합니다.

SWF 파일을 로드할 때 임의의 레벨 번호를 지정할 수 있습니다. 이미 로드된 SWF 파일이 있는 레벨로 SWF 파일을 로드할 수 있으며 새 SWF 파일을 기존 파일로 바꿀 수 있습니다. 레벨 0으로 SWF 파일을 로드하면 Flash Lite 내의 모든 레벨이 언로드되고 레벨 0이 새 파일로 바뀝니다. 로드된 다른 모든 SWF 파일의 프레임 속도, 배경색 및 프레임 크기는 수준 0에 있는 SWF 파일에 의해 설정됩니다.

unloadMovieNum()에 의해 로드된 SWF 파일이나 이미지를 unloadMovieNum()을 사용하여 제거합니다.

예제

다음은 SWF 파일을 레벨2로 로드하는 예제입니다.

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

참고 사항

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

loadVariables()

지원 장치

Flash Lite 1.1.

구문

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

피연산자

url 변수가 현재 위치해 있는 절대 또는 상대 URL을 나타내는 문자열입니다. 이 호출을 수행하는 SWF 파일이 웹 브라우저에서 실행 중인 경우, **url**은 해당 SWF 파일과 같은 도메인에 속해야 합니다.

target 로드할 변수를 수신하는 동영상 클립의 대상 경로입니다.

변수 변수를 전달하는 데 사용할 HTTP 메서드를 지정하는 선택적 문자열 매개 변수입니다. 매개 변수는 문자열 GET 또는 POST 여야 합니다. 전달할 변수가 없으면 이 매개 변수를 생략합니다. GET 메서드는 URL 끝에 변수를 추가하며 변수의 수가 적은 경우 사용됩니다. POST 메서드는 별도의 HTTP 헤더에 있는 변수를 보내며 문자열이 긴 변수에 사용됩니다.

설명

ColdFusion, CGI, ASP(Active Server Pages), PHP 또는 Perl 스크립트로 만든 텍스트 파일이나 텍스트와 같은 외부 파일에서 데이터를 읽어 들인 다음 대상 동영상 클립에서 변수 값을 설정하는 함수입니다. 이 함수는 또한 활성 SWF 파일의 변수를 새 값으로 업데이트할 수 있습니다.

지정된 URL에 있는 텍스트는 표준 MIME 형식인 **application/x-www-form-urlencoded**(CGI 스크립트에서 사용되는 표준 형식)를 따라야 합니다. 지정할 수 있는 변수의 수에는 제한이 없습니다. 예를 들어, 다음 구문에서는 여러 개의 변수를 정의합니다.

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

변수를 특정 레벨로 로드하려면 `loadVariables()` 함수 대신 `loadVariablesNum()` 함수를 사용합니다.

예제

다음은 텍스트 파일 및 서버에서 변수를 로드하는 예제입니다.

```
// load variables from text file on local file system (Symbian Series 60)
on(release, keyPress "1") {
    filePath = "file:///c:/documents/flash/myApp/myvariables.txt";
    loadVariables(filePath, _root);
}

// load variables (from server) into a movieclip
urlPath = "http://www.someserver.com/myvariables.txt";
loadVariables(urlPath, "myClip_mc");
```

참고 사항

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

loadVariablesNum()

지원 장치

Flash Lite 1.1.

구문

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

피연산자

url 로드할 변수가 위치한 절대 또는 상대 URL을 나타내는 문자열입니다. 이 호출을 수행한 SWF 파일이 웹 브라우저에서 실행되고 있다면 **url**은 SWF 파일과 동일한 도메인에 있어야 합니다. 자세한 내용은 다음 설명 단원을 참조하십시오.

level 변수를 수신할 Flash Lite의 레벨을 지정하는 정수입니다.

변수 변수를 전달하는 데 사용할 HTTP 메서드를 지정하는 선택적 문자열 매개 변수입니다. 매개 변수는 문자열 GET 또는 POST 여야 합니다. 전달할 변수가 없으면 이 매개 변수를 생략합니다. GET 메서드는 URL 끝에 변수를 추가하며 변수의 수가 적은 경우 사용됩니다. POST 메서드는 별도의 HTTP 헤더에 있는 변수를 보내며 문자열이 긴 변수에 사용됩니다.

설명

ColdFusion, CGI, ASP(Active Server Pages), PHP 또는 Perl 스크립트로 만든 텍스트 파일이나 텍스트와 같은 외부 파일에서 데이터를 읽어 들인 다음 Flash Lite 레벨에서 변수 값을 설정하는 함수입니다. 이 함수는 또한 활성 SWF 파일의 변수를 새 값으로 업데이트할 수 있습니다.

지정된 URL에 있는 텍스트는 표준 MIME 형식인 **application/x-www-form-urlencoded**(CGI 스크립트에서 사용되는 표준 형식)를 따라야 합니다. 지정할 수 있는 변수의 수에는 제한이 없습니다. 다음은 여러 개의 변수를 정의하는 예제 코드입니다.

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

일반적으로 Flash Lite는 단일 SWF 파일을 표시한 다음 종료됩니다. loadVariablesNum() 함수를 사용하면 여러 SWF 파일을 동시에 표시할 수 있으며, 다른 HTML 문서를 로드하지 않고도 SWF 파일 사이를 전환할 수 있습니다.

변수를 대상 동영상 클립으로 로드하려면 loadVariablesNum() 함수 대신 loadVariables() 함수를 사용합니다.

참고 사항

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

mbchr()

지원 장치

Flash Lite 1.0.

구문

```
mbchr(number)
```

피연산자

number 멀티바이트 문자로 변환하려는 숫자입니다.

설명

ASCII 코드 숫자를 멀티바이트 문자로 변환하는 함수입니다.

예제

다음은 ASCII 코드 번호를 해당 멀티바이트 문자로 변환하는 예제입니다.

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

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

참고 사항

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

mblength()

지원 장치
Flash Lite 1.0.

구문
`mblength(string)`

피연산자
`string` 문자열입니다.

설명
멀티바이트 문자열의 길이를 반환하는 문자열 함수입니다.

예제
다음은 `myString` 변수의 문자열 길이를 표시하는 예제입니다.

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

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

mbord()

지원 장치
Flash Lite 1.0.

구문
`mbord(character)`

피연산자
문자 멀티바이트 숫자로 변환하려는 문자입니다.

설명
지정된 문자를 멀티바이트 숫자로 변환하는 문자열 함수입니다.

예제
다음은 `myString` 변수의 문자를 멀티바이트 숫자로 변환하는 예제입니다.

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

참고 사항

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

mbsubstring()

지원 장치

Flash Lite 1.0.

구문

```
mbsubstring(value, index, count)
```

피연산자

value 새 멀티바이트 문자열을 추출할 멀티바이트 문자열입니다.

index 추출할 첫 번째 문자의 번호입니다.

count 추출된 문자열에 포함되는 문자 수(인덱스 문자 제외)입니다.

설명

멀티바이트 문자열에서 새 멀티바이트 문자열을 추출하는 문자열 함수입니다.

예제

다음은 myString 변수에 포함된 문자열에서 새 멀티바이트 문자열을 추출하는 예제입니다.

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

참고 사항

[mbchr\(\)](#)

nextFrame()

지원 장치

Flash Lite 1.0.

구문

```
nextFrame()
```

피연산자

없음

설명

재생 헤드를 다음 프레임으로 보내고 중지하는 함수입니다.

예제

다음은 사용자가 단추를 클릭하면 재생 헤드가 다음 프레임으로 이동하여 중지하는 예제입니다.

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

참고 사항

[prevFrame\(\)](#)

nextScene()

지원 장치

Flash Lite 1.0.

구문

```
nextScene()
```

피연산자

없음

설명

재생 헤드를 다음 장면의 프레임 1로 보내고 중지하는 함수입니다.

예제

다음은 사용자가 단추를 놓으면 재생 헤드가 다음 장면의 프레임 1로 이동하는 예제입니다.

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

참고 사항

[prevScene\(\)](#)

Number()

지원 장치

Flash Lite 1.0.

구문

```
Number(expression)
```

피연산자

`expression` 숫자로 변환할 표현식입니다.

설명

매개 변수 `expression`을 숫자로 변환하고 다음 목록에서 설명된 대로 값을 반환하는 함수입니다.

- `expression`이 숫자이면 반환값은 `expression`입니다.
- `expression`이 부울 값인 경우 `expression`이 true이면 1을, `expression`이 false이면 0을 반환합니다.

- **expression**이 문자열이면 함수는 **expression**을 선택적 후행 지수(예: 1.57505e-3)를 가진 10진수로 파싱하려고 합니다.
- **expression**이 **undefined**인 경우 반환값은 -1이 됩니다.

예제

다음은 **myString** 변수의 문자열을 숫자로 변환하고, 그 숫자를 **myNumber** 변수에 저장하고 5를 추가한 다음, 결과를 **myResult** 변수에 저장하는 예제입니다. 마지막 행은 부울 값의 **Number()**를 호출할 때 결과를 표시하도록 합니다.

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

trace (myResult);           // Output: 60

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

on()

지원 장치

Flash Lite 1.0.

구문

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

피연산자

statement(s) **event**가 발생하면 실행되는 명령입니다.

event 이 트리거를 **event**라고 합니다. 사용자 이벤트가 발생하면 중괄호({}) 내에서 그 다음의 명령문이 실행됩니다. **event** 매개 변수에는 다음과 같은 값을 지정할 수 있습니다.

- **press** 포인터가 단추 위에 있는 상태에서 단추가 눌러진 상황을 나타냅니다.
- **release** 포인터가 단추 위에 있는 상태에서 눌렀던 단추를 놓은 상황을 나타냅니다.
- **rollOut**: 포인터가 단추 영역 외부로 벗어납니다.
- **rollOver** 포인터가 단추 위로 들어 온 상황을 나타냅니다.
- **keyPress "key"** 지정된 키가 눌러진 상황을 나타냅니다. 이 매개 변수의 **key** 부분에 키 코드 또는 키 상수를 지정합니다.

설명

함수를 트리거하는 사용자 이벤트 또는 키 누르기를 지정하는 이벤트 핸들러입니다. 일부 이벤트에는 지원되지 않습니다.

예제

다음 코드는 사용자가 8 키를 누르면 **maxscroll**을 테스트 한 후 **myText** 필드를 한 행 아래로 스크롤합니다.

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

ord()

지원 장치
Flash Lite 1.0.

구문
ord(character)

피연산자
문자 ASCII 코드 숫자로 변환할 문자입니다.

설명
문자를 ASCII 코드 번호로 변환하는 문자열 함수입니다.

예제
다음은 ord() 함수를 사용하여 문자 **A**에 대한 ASCII 코드를 표시하는 예제입니다.

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

play()

지원 장치
Flash Lite 1.0.

구문
play()

피연산자
없음

설명
타입라인에서 재생 헤드를 앞쪽으로 이동하는 함수입니다.

예제
다음은 if 문을 사용하여 사용자가 입력하는 이름 값을 확인하는 예제입니다. 사용자가 Steve를 입력하면 play() 함수가 호출되고 타입라인에서 재생 헤드가 앞으로 이동합니다. 사용자가 Steve 외에 다른 것을 입력하면 SWF 파일은 재생되지 않고 alert라는 변수 이름을 가진 텍스트 필드가 표시됩니다.

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

prevFrame()

지원 장치
Flash Lite 1.0.

구문
prevFrame()

피연산자
없음

설명
이전 프레임으로 재생 헤드를 보낸 후 중지하는 함수입니다. 현재 프레임이 1이면 재생 헤드가 이동하지 않습니다.

예제
사용자가 다음 핸들러가 첨부된 단추를 클릭하면 재생 헤드는 이전 프레임으로 보내집니다.

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

참고 사항
[nextFrame\(\)](#)

prevScene()

지원 장치
Flash Lite 1.0.

구문
prevScene()

피연산자
없음

설명
재생 헤드를 이전 장면의 프레임 1로 보내고 중지하는 함수입니다.

예제
이 예제에서는 사용자가 다음 핸들러가 첨부된 단추를 클릭하면 재생 헤드는 이전 장면으로 보내집니다.

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

참고 사항
[nextScene\(\)](#)

random()

지원 장치

Flash Lite 1.0.

구문

```
random(value)
```

피연산자

value 정수입니다.

설명

value 매개 변수에 지정된 정수보다 작은 값과 0 사이의 임의의 정수를 반환하는 함수입니다.

예제

다음은 범위를 지정하는 정수 기반의 숫자를 생성하는 예제입니다.

```
//pick random number between 0 and 5  
myNumber = random(5);  
trace (myNumber);           // Output: could be 0,1,2,3,4
```

```
//pick random number between 5 and 10  
myNumber = random(5) + 5;  
trace (myNumber);           // Output: could be 5,6,7,8,9
```

다음은 숫자를 생성한 다음 변수 이름으로 평가되는 문자열의 끝에 이를 연결하는 예제입니다. 이 예제는 Flash Lite 1.1 구문을 사용하여 배열을 시뮬레이션하는 방법을 보여줍니다.

```
// select random name from list  
myNames1 = "Mike";  
myNames2 = "Debbie";  
myNames3 = "Logan";  
  
ran = random(3) + 1;  
ranName = "myNames" + ran;  
trace (eval(ranName)); // Output: will be mike, debbie, or logan
```

removeMovieClip()

지원 장치

Flash Lite 1.0.

구문

```
removeMovieClip(target)
```

피연산자

target duplicateMovieClip()을 사용하여 만드는 동영상 클립 인스턴스의 대상 경로입니다.

설명

duplicateMovieClip()을 사용하여 처음에 만들어진 지정된 동영상 클립을 삭제하는 함수입니다.

예제

다음은 second_mc라는 복제 동영상 클립을 삭제하는 예제입니다.

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

참고 사항

[duplicateMovieClip\(\)](#)

set()

지원 장치

Flash Lite 1.0.

구문

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

피연산자

variable expression 매개 변수의 값을 가지고 있는 식별자입니다.

expression 변수에 지정된 값입니다.

설명

변수에 값을 지정하는 명령문입니다. 변수는 데이터를 가지고 있는 컨테이너입니다. 컨테이너는 항상 동일하지만 컨테이너에 포함되는 콘텐츠는 바뀔 수 있습니다. SWF 파일이 재생되는 동안 변수 값을 변경함으로써 사용자가 수행한 작업에 대한 정보를 기록 및 저장하거나, SWF 파일이 재생되는 동안 변경되는 값을 기록하거나, 조건이 true 또는 false인지 평가할 수 있습니다.

변수에는 String, Number, Boolean, MovieClip 등 모든 데이터 유형을 지정할 수 있습니다. 각 SWF 파일과 동영상 클립의 타임라인은 각각 일련의 변수를 가지고 있으며 각 변수는 다른 타임라인의 변수와 독립된 값을 가지고 있습니다.

예제

다음은 나중에 SWF 파일의 시작 위치로 배를 재설정할 수 있도록 ship 동영상 클립의 원래 x축 위치를 저장하는 변수 orig_x_pos를 설정하는 예제입니다.

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

이전 코드는 다음 코드와 같은 결과를 보여줍니다.

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

setProperty()

지원 장치

Flash Lite 1.0.

구문

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

피연산자

target 속성을 설정할 동영상 클립의 인스턴스 이름에 대한 경로입니다.

속성 설정할 속성입니다.

value 속성의 새 리터럴 값입니다.

expression 속성의 새 값으로 평가하는 방정식입니다.

설명

동영상이 재생될 때 동영상 클립의 속성값을 변경하는 함수입니다.

예제

다음 명령문은 사용자가 이 이벤트 핸들러와 관련된 단추를 클릭할 때 **star** 동영상 클립의 **_alpha** 속성을 30 퍼센트로 설정합니다.

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

참고 사항

[getProperty\(\)](#)

stop()

지원 장치

Flash Lite 1.0.

구문

```
stop()
```

피연산자

없음

설명

현재 재생 중인 SWF 파일을 중지하는 함수입니다. 이 함수는 단추으로 동영상 클립을 조절할 때 가장 많이 사용됩니다.

예제

다음 명령문은 사용자가 이 이벤트 핸들러와 관련된 단추를 클릭할 때 **stop()** 함수를 호출합니다.

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

stopAllSounds()

지원 장치

Flash Lite 1.0.

구문

```
stopAllSounds();
```

피연산자

없음

설명

재생 헤드를 중지하지 않고 SWF 파일에서 현재 재생되고 있는 모든 사운드를 중지하는 함수입니다. 스트리밍으로 설정된 사운드는 사운드가 있는 프레임 위로 재생 헤드가 이동할 때 재생을 계속합니다.

예제

다음 코드는 클릭할 때 SWF 파일의 모든 사운드를 중지하는 단추에 적용될 수 있습니다.

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

String()

지원 장치

Flash Lite 1.0.

구문

```
String(expression)
```

피연산자

`expression` 문자열로 변환할 표현식입니다.

설명

다음 목록에 설명된 대로 지정된 매개 변수의 문자열 표현을 반환하는 함수입니다.

- `expression`이 숫자이면 반환 문자열은 숫자의 텍스트 표현입니다.
- `expression`이 문자열이면 반환 문자열은 `expression`입니다.
- `expression`이 부울 값이면 반환 문자열은 `true` 또는 `false`입니다.
- `expression`이 동영상 클립이면 반환값은 슬래시(/) 표기법으로 표현된 동영상 클립의 대상 경로입니다.

예제

다음은 `birthYearNum`을 1976로 설정하고 `String()` 함수를 사용하여 문자열로 변환한 다음 `eq` 연산자를 사용하여 문자열 "1976"과 비교하는 예제입니다.

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

substring()

지원 장치

Flash Lite 1.0.

구문

```
substring(string, index, count)
```

피연산자

string 새 문자열을 추출할 문자열입니다.

index 추출할 첫 번째 문자의 번호입니다.

count 추출된 문자열에 포함되는 문자 수(인덱스 문자 제외)입니다.

설명

문자열의 일부를 추출하는 함수입니다. 이 함수는 1부터 시작하는 반면 **String** 클래스 메서드는 0부터 시작합니다.

예제

다음은 문자열 "Hello World"에서 첫 5자를 추출하는 예제입니다.

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

tellTarget()

지원 장치

Flash Lite 1.0.

구문

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

피연산자

target 제어할 타임라인의 대상 경로를 지정하는 문자열입니다.

statement(s) 명령문은 조건이 **true**로 평가될 경우에 실행됩니다.

설명

statement(s) 매개 변수에 지정된 명령어를 **target** 매개 변수에 지정된 타임라인에 적용하는 함수입니다. **tellTarget()** 함수는 탐색 컨트롤에 유용합니다. 스테이지의 다른 위치에서 동영상 클립을 중지하거나 시작하는 단추에 **tellTarget()**을 지정합니다. 또한 동영상 클립을 해당 클립의 특정 프레임으로 이동할 수도 있습니다. 예를 들어, 스테이지에서 동영상 클립을 중지 또는 시작하는 단추에 **tellTarget()**을 지정하거나 동영상 클립을 특정 프레임으로 이동하는 프롬프트를 표시할 수도 있습니다.

예제

다음 예제에서 **tellTarget()**은 기본 타임라인의 **ball** 동영상 클립 인스턴스를 제어합니다. **ball** 인스턴스의 프레임 1은 비어 있으며 스테이지에 표시되지 않도록 **stop()** 함수를 갖고 있습니다. 5 키를 누르면 **tellTarget()**은 애니메이션이 시작하는 프레임 2로 이동하도록 **ball**의 재생 헤드에 지시합니다.

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

toggleHighQuality()

지원 장치
Flash Lite 1.0.

구문
toggleHighQuality()

피연산자
없음

설명
Flash Lite에서 엔티앨리어싱을 켜고 끄는 함수입니다. 엔티앨리어싱은 객체의 가장자리를 매끄럽게 하지만 SWF 파일의 재생을 느리게 합니다. 이 함수는 Flash Lite의 모든 SWF 파일에 영향을 줍니다.

예제
엔티앨리어싱을 설정/해제하는 단추에 다음 코드를 적용할 수 있습니다.

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

trace()

지원 장치
Flash Lite 1.0.

구문
trace(expression)

피연산자
expression 평가할 표현식입니다. Flash 제작 도구에서 동영상 테스트 명령으로 SWF 파일을 열면 **expression** 매개 변수 값이 출력 패널에 표시됩니다.

설명
표현식을 평가하여 그 결과를 테스트 모드의 출력 패널에 표시하는 함수입니다.

이 함수를 사용하여 SWF 파일을 테스트하는 동안 프로그래밍 노트를 기록하거나 출력 패널에 메시지를 표시합니다. **expression** 매개 변수를 사용하여 조건이 있는지 확인하거나 출력 창에 값을 표시합니다. trace() 함수는 JavaScript의 alert 함수와 비슷합니다.

제작 설정의 Trace 액션 생략 명령을 사용하여 내보내진 SWF 파일에서 trace() 함수를 제거할 수 있습니다.

예제

다음은 `trace()` 함수를 사용하여 `while` 루프의 비헤이비어를 관찰하는 예제입니다.

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

unloadMovie()

지원 장치

Flash Lite 1.0.

구문

```
unloadMovie(target)
```

피연산자

`target` 동영상 클립의 대상 경로입니다.

설명

[loadMovie\(\)](#), [loadMovieNum\(\)](#) 또는 [duplicateMovieClip\(\)](#)을 사용하여 로드한 Flash Lite의 동영상 클립을 제거하는 함수입니다.

예제

사용자가 3 키를 누르면 다음 코드가 기본 타임라인의 `draggable_mc` 동영상 클립을 언로드하고 `movie.swf`를 문서 스택의 레벨 4로 로드합니다..

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

다음은 사용자가 3 키를 누르면 레벨 4로 로드된 동영상을 언로드하는 예제입니다.

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

참고 사항

[loadMovie\(\)](#)

unloadMovieNum()

지원 장치

Flash Lite 1.0.

구문

```
unloadMovieNum(level)
```

피연산자

`level` 로드된 동영상의 수준(`_levelN`)입니다.

설명

[loadMovie\(\)](#), [loadMovieNum\(\)](#) 또는 [duplicateMovieClip\(\)](#)을 사용하여 로드한 Flash Lite의 동영상 클립을 제거하는 함수입니다.

일반적으로 Flash Lite는 단일 SWF 파일을 표시한 다음 종료됩니다. [loadMovieNum\(\)](#) 함수를 사용하면 여러 SWF 파일에 동시에 영향을 줄 수 있으며, 다른 HTML 문서를 로드하지 않고도 SWF 파일 사이를 전환할 수 있습니다.

참고 사항

[loadMovieNum\(\)](#)

3장: Flash Lite 속성

이 단원에서는 Adobe Macromedia Flash Lite 1.x 소프트웨어에서 인식하는 속성에 대해 설명합니다. 항목은 처음에 오는 밑줄을 무시하고 알파벳 순서로 나열되어 있습니다. 다음 표에 이러한 속성에 대한 설명이 간략하게 나와 있습니다.

속성	설명
/ (슬래시)	기본 동영상 타임라인에 대한 참조를 지정하거나 반환합니다.
<code>_alpha</code>	동영상 클립의 알파 투명도 값을 반환합니다.
<code>_currentframe</code>	타임 라인에 재생 헤드가 있는 프레임 수를 반환합니다.
<code>_focusrect</code>	현재 포커스가 있는 단추 또는 텍스트 필드 주위에 노란 사각형을 표시할지 여부를 지정합니다.
<code>_framesloaded</code>	동적으로 로드된 SWF 파일에서 로드된 프레임 수를 반환합니다.
<code>_height</code>	동영상 클립 높이를 픽셀 단위로 지정합니다.
<code>_highquality</code>	현재의 SWF 파일에 적용된 엔티앨리어싱 수준을 지정합니다.
<code>_level</code>	<code>_levelN</code> 의 루트 동영상 타임라인에 대한 참조를 반환합니다. 반드시 <code>loadMovieNum()</code> 함수를 사용하여 SWF 파일을 Flash Lite Player로 로드한 다음 해당 SWF 파일의 <code>_level</code> 속성을 사용해야 합니다. <code>_levelN</code> 을 사용하면 N에서 지정한 레벨에 로드된 SWF 파일을 대상으로 지정할 수 있습니다..
<code>maxscroll</code>	필드에 최하위 행이 함께 나타나는 경우 스크롤 가능 텍스트 필드의 최상위에 나타나는 행의 행 번호를 나타냅니다.
<code>_name</code>	동영상 클립의 인스턴스 이름을 반환합니다. 동영상 클립에만 적용되며 기본 타임라인에는 적용되지 않습니다.
<code>_rotation</code>	동영상 클립의 원점으로부터의 회전 각도를 반환합니다.
<code>scroll</code>	변수와 관련된 텍스트 필드에서 정보를 표시하는 방법을 제어합니다. <code>scroll</code> 속성은 텍스트 필드가 콘텐츠를 표시하기 시작하는 위치를 정의합니다. 이 위치를 설정한 후에는 사용자가 텍스트 필드를 스크롤하는 대로 위치가 업데이트됩니다.
<code>_target</code>	동영상 클립 인스턴스의 대상 경로를 반환합니다.
<code>_totalframes</code>	동영상 클립의 전체 프레임 수를 반환합니다.
<code>_visible</code>	동영상 클립이 표시되는지 여부를 나타냅니다.
<code>_width</code>	동영상 클립의 너비를 픽셀 단위로 반환합니다.
<code>_x</code>	동영상 클립의 x 좌표를 설정하는 정수를 포함합니다.
<code>_xscale</code>	동영상 클립의 등록 포인트에 적용된 동영상 클립의 가로 크기(<i>percentage</i>)를 설정합니다.
<code>_y</code>	부모 동영상 클립의 로컬 좌표에 대한 동영상 클립의 상대적인 y 좌표를 설정하는 정수를 포함합니다.
<code>_yscale</code>	동영상 클립의 등록 포인트에 적용된 동영상 클립의 세로 크기(<i>percentage</i>)를 설정합니다.

/ (슬래시)

지원 장치

Flash Lite 1.0

구문

```
/
/targetPath
/:varName
```

설명

기본 동영상 타임라인에 대한 참조를 지정하거나 반환하는 식별자입니다. 이 속성이 제공하는 기능은 Macromedia Flash 5의 `_root` 속성이 제공하는 기능과 유사합니다.

예제

타임라인의 변수를 지정하려면 콜론(:)과 결합된 슬래시 구문(/)을 사용합니다.

예제 1: 기본 타임라인의 `car` 변수:

```
/:car
```

예제 2: 기본 타임라인에 있는 동영상 클립 인스턴스 `mc1`의 `car` 변수:

```
/mc1/:car
```

예제 3: 기본 타임라인에 있는 동영상 클립 인스턴스 `mc1`에 중첩된 동영상 클립 인스턴스 `mc2`의 `car` 변수:

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

예제 4: 현재 타임라인에 있는 동영상 클립 인스턴스 `mc2`의 `car` 변수:

```
mc2/:car
```

_alpha

지원 장치

Flash Lite 1.0.

구문

```
my_mc:alpha
```

설명

`my_mc` 변수에 의해 지정된 동영상 클립의 알파 투명도 값을 나타내는 속성입니다. 유효 값은 0(완전 투명)부터 100(완전 불투명)까지입니다. 기본 값은 100입니다. `_alpha`가 0으로 설정된 동영상 클립의 객체는 표시되지 않더라도 활성 상태입니다. 예를 들어, 동영상 클립의 `_alpha` 속성이 0으로 설정되었더라도 동영상 클립에서 단추를 클릭할 수 있습니다.

예제

단추 이벤트 핸들러에 대한 다음 코드는 사용자가 단추를 클릭할 때 `my_mc` 동영상 클립의 `_alpha` 속성을 30%로 설정합니다.

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

_currentframe

지원 장치
Flash Lite 1.0.

구문
my_mc:_currentframe

설명
my_mc 변수에 의해 지정된 타임라인에서 재생 헤드가 위치하고 있는 프레임의 번호를 반환하는 읽기 전용 속성입니다.

예제
다음 예제에서는 **_currentframe** 속성 및 **gotoAndStop()** 함수를 사용하여 동영상 클립 **my_mc**의 재생 헤드를 현재 위치보다 다섯 프레임 앞으로 이동합니다.

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

참고 사항
[gotoAndStop\(\)](#)

_focusrect

지원 장치
Flash Lite 1.0.

구문
_focusrect = Boolean;

설명
현재 포커스가 활성화된 단추거나 텍스트 필드 주위를 노란 사각형으로 둘러싸게 할지 여부를 지정하는 전역 속성입니다. 기본 값 **true**는 사용자가 전화나 휴대 장치의 위 또는 아래 화살표 키를 눌러 SWF 파일의 객체를 탐색할 때 현재 포커스를 가진 단추 또는 텍스트 필드 주위에 노란 사각형을 표시합니다. **false**로 지정하면 노란색 사각형이 표시되지 않습니다.

예제
다음은 응용 프로그램에 나타나는 노란 포커스 사각형을 비활성화하는 예제입니다.

```
_focusrect = false;
```

_framesloaded

지원 장치

Flash Lite 1.0.

구문

```
my_mc:_framesloaded
```

설명

동적으로 로드된 SWF 파일에서 로드된 프레임 수를 나타내는 읽기 전용 속성입니다. 이 속성은 특정 프레임의 내용과 이 프레임 이전의 모든 프레임을 로드하여 브라우저에서 로컬로 사용할 수 있는지 여부를 파악하는 데 유용합니다. 또한 대용량 SWF 파일의 다운로드 과정을 모니터링하는 데 유용합니다. 예를 들어, SWF 파일에서 지정된 프레임의 로드 작업이 완료될 때까지 SWF 파일을 로드하고 있다는 메시지를 사용자에게 표시할 수 있습니다.

예제

다음 예제에서는 모든 프레임이 로드된 경우 `_framesloaded` 속성을 사용하여 SWF 파일을 시작합니다. 모든 프레임이 로드되지 않은 경우 동영상 클립 인스턴스 `loader`의 `_xscale` 속성이 로드된 프레임의 수와 비례적으로 증가하여 진행률 막대가 만들어집니다.

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

_height

지원 장치

Flash Lite 1.0.

구문

```
my_mc:_height
```

설명

동영상 클립 높이를 픽셀 단위로 나타내는 읽기 전용 속성입니다.

예제

이벤트 핸들러 코드에 대한 다음 예제는 사용자가 마우스 단추를 클릭할 때 동영상 클립 높이를 설정합니다.

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

`_highquality`

지원 장치
Flash Lite 1.0.

구문
`_highquality`

설명
현재 SWF 파일에 적용되는 엔티앨리어싱의 레벨을 지정하는 전역 속성입니다. 최고 품질의 엔티앨리어싱을 적용하려면 2를 지정합니다. 고품질의 엔티앨리어싱을 적용하려면 1을 지정합니다. 엔티앨리어싱을 적용하지 않으려면 0을 지정합니다.

예제
다음 명령문은 현재 SWF 파일에 고품질 엔티앨리어싱을 적용합니다.

```
_highquality = 1;
```

참고 사항
[toggleHighQuality\(\)](#)

`_level`

지원 장치
Flash Lite 1.0.

구문
`_levelN`

설명
`_levelN`의 루트 동영상 타임라인에 대한 참조를 나타내는 식별자입니다. 반드시 `loadMovieNum()` 함수를 사용하여 SWF 파일을 Flash Lite Player로 로드한 다음 해당 SWF 파일의 `_level` 속성을 사용해야 합니다. `_levelN`을 사용하면 **N**으로 할당된 수준에 로드된 SWF 파일을 대상으로 지정할 수 있습니다.

Flash Lite Player의 인스턴스에 로드된 초기 SWF 파일이 자동으로 `_level0`으로 로드됩니다. `_level0`에 있는 SWF 파일이 순차적으로 로드된 모든 SWF 파일의 프레임 속도, 배경색 및 프레임 크기를 설정합니다. 그런 다음, SWF 파일은 `_level0`에 있는 SWF 파일보다 상위 번호의 수준에 쌓입니다.

`loadMovieNum()` 함수를 사용하여 Flash Lite Player에 로드하는 각각의 SWF 파일에는 하나의 레벨을 지정해야 합니다. 이때 수준 순서에는 제한이 없습니다. 이미 SWF 파일이 포함되어 있는 수준(`_level0` 포함)을 할당하면 해당 수준에 있는 SWF 파일이 언로드되어 새 SWF 파일로 대체됩니다.

예제
다음은 SWF 파일을 레벨 1로 로드한 다음 프레임 6에 로드된 SWF 파일의 재생 헤드를 중지하는 예제입니다.

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

// at least 1 frame later
tellTarget(_level1) {
    gotoAndStop(6);
}
```

참고 사항

[loadMovie\(\)](#)

maxscroll

지원 장치

Flash Lite 1.1

구문

```
variable_name:maxscroll
```

설명

텍스트 필드에 마지막 행이 함께 나타나는 경우 스크롤할 수 있는 텍스트 필드에 나타나는 첫 번째 행의 행 번호를 나타내는 읽기 전용 속성입니다. `maxscroll` 속성은 `scroll` 속성과 연계하여 텍스트 필드에서 정보를 표시하는 방법을 제어합니다. 이 속성은 가져올 수는 있지만 변경할 수는 없습니다.

예제

다음 코드는 사용자가 8 키를 누르면 `maxscroll`을 테스트 한 후 `myText` 필드를 한 행 아래로 스크롤합니다.

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

참고 사항

[scroll](#)

_name

지원 장치

Flash Lite 1.0.

구문

```
my_mc:_name
```

설명

`my_mc`에 지정된 동영상 클립의 인스턴스 이름을 나타내는 속성입니다. 동영상 클립에만 적용되며 기본 타임라인에는 적용되지 않습니다.

예제

다음은 출력 패널의 `bigRose` 동영상 클립 이름을 문자열로 표시하는 예제입니다.

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

_rotation

지원 장치

Flash Lite 1.0.

구문

```
my_mc:_rotation
```

설명

동영상 클립의 원점으로부터의 회전 각도를 나타내는 속성입니다. 0~180의 값은 시계 방향 회전을 나타내고 0~180의 값은 시계 반대 방향 회전을 나타냅니다. 이 범위를 벗어나는 값은 360과 더하거나 빼서 범위 내의 값을 구합니다. 예를 들어, 명령문 `my_mc:_rotation = 450`은 `my_mc:_rotation = 90`과 같습니다.

예제

다음은 사용자가 2 키를 누르면 `my_mc` 동영상 클립이 15도 시계 방향으로 회전하는 예제입니다.

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

scroll

지원 장치

Flash Lite 1.1.

구문

```
textFieldVariableName:scroll
```

설명

변수와 관련된 텍스트 필드의 정보 표시를 조절하는 속성입니다. `scroll` 속성은 텍스트 필드가 콘텐츠를 표시하기 시작하는 위치를 정의합니다. 이 위치를 설정한 후에는 사용자가 텍스트 필드를 스크롤하는 대로 위치가 업데이트됩니다. `scroll` 속성을 사용하여 스크롤 텍스트 필드를 만들거나 사용자를 특정 단락으로 이동할 수 있습니다.

예제

다음 코드는 사용자가 2 키를 클릭할 때마다 `myText` 텍스트 필드를 한 행 위로 스크롤합니다.

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

참고 사항

[maxscroll](#)

`_target`

지원 장치

Flash Lite 1.0.

구문

```
my_mc:_target
```

설명

`my_mc`에 지정한 동영상 클립 인스턴스의 대상 경로를 반환하는 읽기 전용 속성입니다.

`_totalframes`

지원 장치

Flash Lite 1.0.

구문

```
my_mc:_totalframes
```

설명

`my_mc` 동영상 클립의 전체 프레임 수를 반환하는 읽기 전용 속성입니다.

예제

다음 코드는 `mySWF.swf`를 레벨 1로 로드한 다음 나중에 25개의 프레임을 로드한 후 파일이 로드되었는지 확인합니다.

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

// 25 frames later in the main timeline
if (_level1._framesloaded >= _level1._totalframes) {
    tellTarget("_level1/") {
        gotoAndStop("myLabel");
    }
} else {
    // loop...
}
```

`_visible`

지원 장치

Flash Lite 1.0.

구문

```
my_mc:_visible
```

설명

my_mc에 지정한 동영상 클립을 보이도록 할지 여부를 지정하는 부울 값 속성입니다. 표시되지 않은 동영상 클립(**_visible** 속성이 **false**로 설정된 동영상 클립)은 사용할 수 없습니다. 예를 들어, **_visible** 속성이 **false**로 설정된 동영상 클립의 단추는 클릭할 수 없습니다. 동영상 클립은 이러한 방법으로 명시적으로 숨기지 않는 한 표시됩니다.

예제

다음 코드는 사용자가 3 키를 누르면 **my_mc** 동영상 클립을 비활성화하고 4 키를 누르면 활성화합니다.

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

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

_width

지원 장치

Flash Lite 1.0.

구문

```
my_mc._width
```

설명

동영상 클립의 폭을 픽셀 단위로 나타내는 속성입니다.

예제

다음은 사용자가 5 키를 누르면 동영상 클립의 폭 속성을 설정하는 예제입니다.

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

_x

지원 장치

Flash Lite 1.0.

구문

```
my_mc._x
```

설명

동영상 클립의 **x** 좌표를 부모 동영상 클립(**my_mc**로 표현됨)의 로컬 좌표에 대한 상대 좌표로 설정하는 정수입니다. 동영상 클립이 기본 타임라인에 있는 경우 좌표계는 스테이지의 왼쪽 위 모서리를 (0, 0)으로 간주합니다.

변형을 포함하는 다른 동영상 클립 내에 동영상 클립이 있는 경우, 이 동영상 클립은 포함하는 동영상 클립의 로컬 좌표계에 있습니다. 예를 들어, 동영상 클립을 시계 반대 방향으로 90도 회전시킨 경우 자식 동영상 클립은 시계 반대 방향으로 90도 회전된 좌표계를 상속합니다. 동영상 클립의 좌표는 등록 포인트 위치를 참조합니다.

예제

다음은 사용자가 6 키를 누르면 my_mc 동영상 클립의 수평 위치를 변경하는 예제입니다.

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

참고 사항

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

_xscale

지원 장치

Flash Lite 1.0.

구문

```
my_mc._xscale
```

설명

동영상 클립의 등록 포인트에 적용된 동영상 클립의 가로 크기(**percentage**)를 설정하는 속성입니다. 기본 등록 포인트는 (0, 0)입니다.

로컬 좌표계의 크기를 조절하게 되면 픽셀 단위로 정의되는 `_x` 및 `_y` 속성 설정에 영향을 줍니다. 예를 들어, 부모 동영상 클립의 크기를 50%로 조절하는 경우 `_x` 속성을 설정하면 동영상이 100%일 경우 픽셀 수의 절반만큼 동영상 클립에서 객체를 이동합니다.

예제

다음은 사용자가 7 키를 누르면 my_mc 동영상 클립의 가로 크기를 변경하는 예제입니다.

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

참고 사항

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

_y

지원 장치

Flash Lite 1.0.

구문

```
my_mc._y
```

설명

동영상 클립의 **y** 좌표를 부모 동영상 클립(**my_mc**로 표현됨)의 로컬 좌표에 대한 상대 좌표로 설정하는 정수입니다. 동영상 클립이 기본 타임라인에 있는 경우 좌표계는 스테이지의 왼쪽 위 모서리를 (0, 0)으로 간주합니다.

변형을 포함하는 다른 동영상 클립 내에 동영상 클립이 있는 경우, 이 동영상 클립은 포함하는 동영상 클립의 로컬 좌표계에 있습니다. 예를 들어, 동영상 클립을 시계 반대 방향으로 90도 회전시킨 경우 자식 동영상 클립은 시계 반대 방향으로 90도 회전된 좌표계를 상속합니다. 동영상 클립의 좌표는 등록 포인트 위치를 참조합니다.

예제

다음 코드는 사용자가 1키를 누를 때 my_mc 동영상 클립의 y 좌표를 부모 클립의 (0, 0) 좌표 아래로 10픽셀 이동합니다.

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

참고 사항

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

_yscale

지원 장치

Flash Lite 1.0.

구문

```
my_mc._yscale
```

설명

동영상 클립의 등록 포인트에 적용된 동영상 클립의 세로 크기(percentage)를 설정하는 속성입니다. 기본 등록 포인트는 (0, 0)입니다.

로컬 좌표계의 크기를 조절하게 되면 픽셀 단위로 정의되는 `_x` 및 `_y` 속성 설정에 영향을 줍니다. 예를 들어, 부모 동영상 클립의 크기를 50%로 조절하는 경우 `_y` 속성을 설정하면 동영상이 100%일 경우 픽셀 수의 절반만큼 동영상 클립에서 객체를 이동합니다.

예제

다음은 사용자가 1 키를 누르면 my_mc 동영상 클립의 가로 크기를 10%로 변경하는 예제입니다.

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

참고 사항

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

4장: Flash Lite 명령문

이 단원에서는 Adobe Macromedia Flash Lite 1.x의 ActionScript 명령문의 구문과 사용에 대해 설명합니다. 명령문은 액션을 수행하거나 지정하는 언어 요소입니다. 다음 표에 이러한 명령문에 대한 설명이 간략하게 나와 있습니다.

명령문	설명
<code>break</code>	Flash Lite에서 나머지 루프 본문을 건너뛰고 반복 액션을 중단한 다음, 루프 명령문 뒤에 나오는 명령문을 실행합니다.
<code>case</code>	<code>switch</code> 문의 조건을 정의합니다. <code>case</code> 키워드 다음에 나오는 <i>expression</i> 매개 변수가 <code>switch</code> 명령문의 <i>expression</i> 매개 변수와 동일한 경우 <i>statements</i> 매개 변수에 있는 명령문이 실행됩니다.
<code>continue</code>	가장 안쪽의 루프에 남아 있는 명령문을 모두 전달하고 제어가 루프의 끝까지 정상적으로 전달된 것처럼 루프의 다음 반복을 시작합니다.
<code>do..while</code>	명령문을 실행한 다음 조건이 <code>true</code> 로 평가되는 동안 루프에서 조건을 평가합니다.
<code>else</code>	<code>if</code> 문의 조건이 <code>false</code> 로 결과가 나오는 경우 실행할 명령문을 지정합니다.
<code>else if</code>	조건을 평가한 후 초기 <code>if</code> 문의 조건이 <code>false</code> 를 반환하는 경우 실행할 명령문을 지정합니다.
<code>for</code>	<i>init</i> 초기화 표현식을 한 번 평가한 다음 <i>condition</i> 이 <code>true</code> 이면 <i>statement</i> 가 실행되고 다음 표현식이 평가되는 식으로 루핑 시퀀스를 시작합니다.
<code>if</code>	SWF 파일의 다음 액션을 결정할 조건을 평가합니다. 조건이 <code>true</code> 이면 중괄호({}) 내에서 조건 뒤에 있는 명령문이 실행됩니다. 조건이 <code>false</code> 이면 중괄호로 둘러싸인 명령문을 건너뛰고 그 다음 명령문이 실행됩니다.
<code>switch</code>	<code>if</code> 명령문과 유사하게 <code>switch</code> 명령문은 조건을 확인하여 조건이 <code>true</code> 로 평가되는 경우 명령문을 실행합니다.
<code>while</code>	표현식을 평가하고 표현식이 <code>true</code> 인 동안 루프에서 일련의 명령문을 반복적으로 실행합니다.

break

지원 장치
Flash Lite 1.0.

구문
`break`

매개 변수
없음

설명

루프(`for`, `do..while` 또는 `while`) 내에 나타나거나 `switch` 문 내의 특정 `case`와 연관된 명령문 블록 내에 나타납니다. `break` 명령문에 의해 Flash Lite는 나머지 루프 본문을 건너뛰고 반복 액션을 중단한 다음, 루프 문 뒤에 나오는 명령문을 실행합니다. `break` 명령문을 사용하는 경우 ActionScript 인터프리터는 `case` 블록에서 나머지 명령문을 건너뛰고 `switch` 명령문 뒤에 나오는 첫 번째 명령문으로 이동합니다. 일련의 중첩 루프에서 빠져 나오려면 이 명령문을 사용합니다.

예제

다음 예제에서는 `break` 문을 사용하여 무한 루프에서 빠져 나옵니다.

```

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

```

참고 사항

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

case

지원 장치

Flash Lite 1.0.

구문

```
case expression: statements
```

매개 변수

`expression` 모든 표현식입니다.

명령문 모든 명령문입니다.

설명

`switch` 명령문에 대한 조건을 정의하는 명령문입니다. `case` 키워드 다음에 나오는 **expression** 매개 변수가 `switch` 명령문의 **expression** 매개 변수와 동일한 경우 **statements** 매개 변수에 있는 명령문이 실행됩니다.

`switch` 명령문 밖에서 `case` 명령문을 사용하면 오류가 발생하고 코드가 컴파일되지 않습니다.

예제

다음 예제에서는 `myNum` 매개 변수가 1로 평가될 경우 `case 1` 뒤에 나오는 `trace()` 명령문이 실행되고, `myNum` 매개 변수가 2로 평가될 경우 `case 2` 뒤에 나오는 `trace()` 명령문이 실행됩니다. `number` 매개 변수와 일치하는 `case` 표현식이 없을 경우 `default` 키워드 뒤에 나오는 `trace()` 명령문이 실행됩니다.

```

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")
}

```

다음 예제에서는 첫 번째 `case` 그룹에서 `break`가 나타나지 않으므로 숫자가 1인 경우 A 및 B 모두 출력 패널에 나타납니다.

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

참고 사항

[switch](#)

continue

지원 장치

Flash Lite 1.0.

구문

```
continue
```

매개 변수

없음

설명

가장 안쪽의 루프에 남아 있는 명령문을 모두 전달하고 제어기가 루프의 끝까지 정상적으로 전달된 것처럼 루프의 다음 반복을 시작하는 명령문입니다. 이 명령문은 루프 외부에는 적용되지 않습니다.

- while 루프에서 continue를 사용하면 Flash 인터프리터는 나머지 루프 부분을 건너뛴 다음, 조건을 확인하는 루프 맨 위로 이동합니다.
- do..while 루프에서 continue를 사용하면 Flash 인터프리터는 루프 본문의 나머지를 건너뛴 다음, 조건이 테스트되는 루프의 맨 아래로 이동합니다.
- for 루프에서 continue를 사용하면 Flash 인터프리터는 나머지 루프 부분을 건너뛴 다음, for 루프의 그 다음 표현식을 평가하는 위치로 이동합니다.

예제

다음 while 루프에서 continue를 사용하면 Flash Lite는 나머지 루프 부분을 건너뛴 다음, 조건을 확인하는 루프 맨 위로 이동합니다.

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

다음 do..while 루프에서 continue를 사용하면 Flash Lite는 루프 본문의 나머지를 건너뛴 다음, 조건이 테스트되는 루프의 맨 아래로 이동합니다.

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

for 루프에서 continue를 사용하면 Flash Lite는 루프 본문의 나머지를 건너뛵니다. 다음 예제에서는 i 모듈러스 3이 0과 같으면 trace(i) 명령문을 건너뛵니다.

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

참고 사항

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

do..while

지원 장치

Flash Lite 1.0.

구문

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

매개 변수

statement(s) **condition** 매개 변수가 true로 평가되는 동안 실행되는 명령문입니다.

condition 평가할 조건입니다.

설명

명령문을 실행한 다음 조건이 true로 평가되는 동안 루프에서 조건을 평가하는 명령문입니다.

예제

다음은 변수의 값이 10보다 작을 경우 인덱스 변수를 증가시키는 예제입니다.

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

참고 사항

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

else

지원 장치

Flash Lite 1.0.

구문

```

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

```

매개 변수

condition true 또는 false로 평가되는 표현식입니다.

t-statement(s) 명령문은 조건이 true로 평가될 경우에 실행됩니다.

f-statement(s) 조건이 false로 평가될 경우 실행할 대체 명령어입니다.

설명

if 문의 조건이 false로 평가될 경우 실행할 명령문을 지정하는 명령문입니다.

예제

다음 예제에서는 조건과 함께 else 문 사용 방법을 보여줍니다. 실제 예제에서는 이 조건에 따라 액션을 취할 코드가 포함될 것입니다.

```

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

```

참고 사항

[if](#)

else if

지원 장치

Flash Lite 1.0.

구문

```

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

```

매개 변수

condition true 또는 false로 평가되는 표현식입니다.

statement(s) if 명령문에 지정된 조건이 false인 경우 실행할 일련의 명령문입니다.

설명

조건을 평가하고 초기 if 명령문에서 false 값이 반환되는 경우 실행할 명령문을 지정하는 명령문입니다. else if 조건에서 true 값이 반환되는 경우 Flash 인터프리터는 중괄호({}) 내에서 else if 조건 뒤의 명령문을 실행합니다. else if 조건이 false이면 중괄호로 둘러싸인 명령문을 건너뛰고 그 다음 명령문을 실행합니다. 스크립트에서 분기 논리를 만들려면 elseif 명령문을 사용합니다.

예제

다음 예제에서는 else if 명령문을 사용하여 객체의 각 면이 지정된 경계 내에 존재하는지 여부를 확인합니다.

```
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");
}
```

참고 사항

[if](#)

for

지원 장치

Flash Lite 1.0.

구문

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

매개 변수

init 루프 시퀀스를 시작하기 전에 평가할 표현식, 즉 일반적으로 대입 표현식입니다.

condition true 또는 false로 평가되는 표현식입니다. 조건은 각 루프가 반복되기 전에 평가됩니다. 조건이 false로 평가되면 해당 루프가 종료됩니다.

next 루프가 반복될 때마다 평가하는 표현식입니다. 일반적으로 증가(++) 또는 감소(--) 연산자를 사용하는 대입 표현식입니다.

statement(s) 루프 내에서 실행될 하나 이상의 구문입니다.

설명

init 초기화 표현식을 한 번 평가한 다음 **condition**이 true이면 **statement**가 실행되고 다음 표현식이 평가되는 식으로 루프 시퀀스를 시작하는 루프 구조를 나타내는 명령문입니다.

일부 속성은 for 또는 for..in 문을 사용하여 열거할 수 없습니다. 예를 들어, _x 및 _y와 같은 동영상 클립 속성은 열거되지 않습니다.

예제

다음은 for 루프를 사용해 1부터 100까지의 수를 더하는 예제입니다.

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

참고 사항

++(증가), --(감소 연산자)--, do..while, while

if

지원 장치

Flash Lite 1.0.

구문

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

매개 변수

condition true 또는 false로 평가되는 표현식입니다.

statement(s) 명령문은 조건이 true로 평가될 경우에 실행됩니다.

설명

SWF 파일의 다음 액션을 결정할 조건을 평가하는 명령문입니다. 조건이 true이면 중괄호({}) 내에서 조건 뒤에 있는 명령문이 실행됩니다. 조건이 false이면 중괄호로 둘러싸인 명령문을 건너뛰고 그 다음 명령문이 실행됩니다. 스크립트에 분기 논리를 만들려면 if 명령문을 사용합니다.

예제

다음 예제에서는 괄호 안의 조건으로 name 변수 값이 "Erica"라는 리터럴 값과 일치하는지 평가합니다. 조건에 일치하면 play() 함수가 실행됩니다.

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

switch

지원 장치

Flash Lite 1.0.

구문

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

매개 변수

expression 임의의 숫자 표현식입니다.

caseClause 표현식이 스위치 **expression** 매개 변수와 일치할 때 실행할 표현식, 콜론, 명령문 그룹이 **case** 키워드 다음에 나옵니다.

defaultClause **case** 표현식 중에서 스위치 **expression** 매개 변수와 일치하는 것이 없을 경우 실행할 명령문이 선택적 **default** 키워드 다음에 나옵니다.

설명

ActionScript 문의 분기 구조를 만드는 명령문입니다. if 명령문과 유사하게 switch 명령문은 조건을 확인하여 조건이 true로 평가되는 경우 명령문을 실행합니다.

Switch 명령문에는 초기값이라는 대안 옵션이 포함됩니다. true인 명령문이 없는 경우 기본 명령문이 실행됩니다.

예제

다음 예제에서는 myNum 매개 변수가 1로 평가될 경우 case 1 뒤에 나오는 **trace()** 명령문이 실행되고, myNum 매개 변수가 2로 평가될 경우 case 2 뒤에 나오는 **trace()** 명령문이 실행됩니다. number 매개 변수와 일치하는 case 표현식이 없을 경우 default 키워드 뒤에 나오는 **trace()** 명령문이 실행됩니다.

```
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")
}
```

다음 예제에서는 첫 번째 case 그룹에 break가 포함되지 않으므로 숫자가 1인 경우 A 및 B 모두 출력 패널에 나타납니다.

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

참고 사항

[case](#)

while

지원 장치

Flash Lite 1.0.

구문

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

매개 변수

condition while 명령문이 실행될 때마다 평가되는 표현식입니다.

statement(s) 명령문은 조건이 true로 평가될 경우에 실행됩니다.

설명

표현식을 평가하고 표현식이 true인 동안 루프에서 일련의 명령문을 반복적으로 실행하는 명령문입니다.

명령문 블록이 실행되기 전에 조건이 평가되며 평가 결과가 true면 명령문 블록이 실행됩니다. 조건이 false면 명령문 블록을 건너뛰고 while 명령문의 명령문 블록 다음의 첫 번째 명령문이 실행됩니다.

일반적으로 루프는 카운터 변수가 지정된 값보다 작을 때 액션을 수행하기 위해 사용됩니다. 각 루프의 끝부분에서 지정된 값이 될 때까지 카운터가 증가합니다. 이때 **condition**은 더 이상 true가 아니며 루프가 끝납니다.

while 문은 다음과 같은 일련의 단계를 수행합니다. 1단계~4단계까지의 각 반복을 루프 반복이라고 합니다. 각 반복이 시작될 때마다 조건이 평가됩니다.

- 1 **condition** 표현식이 평가됩니다.
- 2 **condition**이 true로 평가되거나 0이 아닌 숫자 값이 부울 값 true로 변환되는 값으로 평가되면 3단계로 이동합니다. 그렇지 않으면 while 명령문이 완료되고 while 루프 후 다음 명령문에서 실행이 계속됩니다.
- 3 명령문 블록 **statement(s)**를 실행합니다.
- 4 1단계로 이동합니다.

예제

다음은 인덱스 변수 i의 값이 10보다 작은 동안 루프를 실행하는 예제입니다.

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

참고 사항

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

5장: Flash Lite 연산자

이 단원에서는 Adobe Macromedia Flash Lite 1.x의 ActionScript 연산자의 구문과 사용에 대해 설명합니다. 모든 항목은 알파벳 순서로 나열됩니다. 그러나 심볼로 이루어진 일부 연산자는 텍스트 설명의 알파벳 순서대로 나열됩니다.

다음 표에 이 단원에서 다루는 연산자에 대한 설명이 간략하게 나와 있습니다.

연산자	설명
add(문자열 결합 연산자)	둘 이상의 문자열을 연결하거나 결합합니다.
+=(더하기 대입)	<i>expression1</i> 에 <i>expression1 + expression2</i> 의 값을 대입합니다.
및	논리 AND 연산을 수행합니다.
=(대입)	<i>expression2</i> 의 값(오른쪽 피연산자)을 <i>expression1</i> 의 변수 또는 속성에 대입합니다.
/*(블록 주석 연산자)	한 행 이상의 스크립트 주석을 나타냅니다. 열기 주석 태그(/*)와 닫기 주석 태그(*/) 사이에 존재하는 모든 문자는 주석으로 해석되며 ActionScript 인터프리터에서 무시됩니다.
,(쉼표)	<i>expression1</i> 과 <i>expression2</i> 를 차례로 평가한 후 <i>expression2</i> 의 결과를 반환합니다.
//(comment)	스크립트 주석의 시작을 나타냅니다. 주석 기호(//)와 행의 끝 문자 사이에 있는 모든 문자는 주석으로 해석되며 ActionScript 인터프리터에서 무시됩니다.
?(조건부)	<i>expression1</i> 을 평가한 후 <i>expression1</i> 값이 true이면 <i>expression2</i> 의 값을 반환하고, 그렇지 않으면 <i>expression3</i> 의 값을 반환합니다.
--(감소 연산자)--	<i>expression</i> 에서 1을 뺍니다. 선행 감소 형식의 연산자(-- <i>expression</i>)는 <i>expression</i> 에서 1을 뺀 후 그 결과를 반환합니다. 후행 감소 형식의 연산자(<i>expression</i> --)는 <i>expression</i> 에서 1을 뺀 후 <i>expression</i> 의 초기값(빼기 이전의 값)을 반환합니다.
/(나누기 연산자)	<i>expression1</i> 을 <i>expression2</i> 로 나눕니다.
/(나누기 대입)	<i>expression1</i> 에 <i>expression1 expression2</i> 의 값을 대입합니다.
.(도트)	중첩된 자식 동영상 클립, 변수 또는 속성에 액세스하기 위해 동영상 클립 계층을 탐색하는 데 사용됩니다.
++(증가)	<i>expression</i> 에 1을 더합니다. <i>expression</i> 은 변수, 배열 요소 또는 객체의 속성이 될 수 있습니다. 선행 증가 형식의 연산자(++ <i>expression</i>)는 <i>expression</i> 에 1을 더한 후 그 결과를 숫자로 반환합니다. 후행 증가 형식의 연산자(<i>expression</i> ++)는 <i>expression</i> 에 1을 더한 후 <i>expression</i> 의 초기값(더하기 이전의 값)을 반환합니다.
&&(논리 AND)	연산자의 왼쪽 표현식인 <i>expression1</i> 을 평가한 후 이 표현식이 false로 평가되면 false를 반환합니다. <i>expression1</i> 이 true로 평가되면 <i>expression2</i> (연산자 오른쪽의 표현식)가 평가됩니다. <i>expression2</i> 가 true로 평가되면 최종 결과는 true이고, 그렇지 않으면 false입니다.
!(논리 NOT)	변수 또는 표현식의 부울 값을 반대로 합니다. <i>expression</i> 이 절대값을 포함한 변수이거나 변환 값이 true인 경우 ! <i>expression</i> 값은 false입니다. 표현식 <i>x && y</i> 가 false로 평가되면 표현식 !(<i>x && y</i>)는 true로 평가됩니다.
 (논리 OR)	<i>expression1</i> 과 <i>expression2</i> 를 평가합니다. 표현식 중 하나 또는 모두가 true로 평가되면 그 결과는 true이며 표현식 모두 false로 평가되는 경우에만 그 결과가 false입니다. 논리 OR 연산자와 다른 피연산자를 함께 사용할 수 있으며 피연산자가 true로 평가되면 그 결과는 true입니다.
%(모듈러스)	<i>expression1</i> 을 <i>expression2</i> 로 나눈 나머지를 계산합니다. <i>expression</i> 피연산자가 숫자가 아닌 경우 모듈러스 연산자는 피연산자를 숫자로 변환하려 합니다.

연산자	설명
%=(모듈러스 대입)	<i>expression1</i> 에 <i>expression1 % expression2</i> 의 값을 대입합니다.
*=(곱하기 대입)	<i>expression1</i> 에 <i>expression1 * expression2</i> 의 값을 대입합니다.
*(곱하기 연산자)	두 개의 숫자 표현식을 곱합니다.
+(숫자 더하기 연산자)	숫자 표현식을 추가합니다.
==(숫자 항등 연산자)	항등 여부를 테스트하는 것으로 <i>expression1</i> 이 <i>expression2</i> 와 동일한 경우 결과가 true입니다.
>(보다 큼 숫자 연산자)	두 표현식을 비교하여 <i>expression1</i> 이 <i>expression2</i> 보다 큰지 여부를 확인합니다. 큰 경우 이 연산자는 true를 반환합니다. <i>expression1</i> 이 <i>expression2</i> 보다 작거나 같은 경우 이 연산자는 false를 반환합니다.
>=(보다 크거나 같음 숫자 연산자)	두 표현식을 비교하여 <i>expression1</i> 이 <i>expression2</i> 보다 크거나 같은지(true) 또는 <i>expression1</i> 이 <i>expression2</i> 보다 작은지(false)를 확인합니다.
<>(비항등 숫자 연산자)	항등 여부를 테스트하는 것으로 <i>expression1</i> 이 <i>expression2</i> 와 동일한 경우 결과가 true입니다.
<(보다 작음 숫자)	두 표현식을 비교하여 <i>expression1</i> 이 <i>expression2</i> 보다 작은지 여부를 확인합니다. 작은 경우 이 연산자는 true를 반환합니다. <i>expression1</i> 이 <i>expression2</i> 보다 크거나 같은 경우 이 연산자는 false를 반환합니다..
<=(보다 작거나 같음 숫자 연산자)	두 표현식을 비교하고 <i>expression1</i> 이 <i>expression2</i> 보다 작거나 같은지 결정합니다. 작거나 같은 경우 연산자는 true를 반환하며, 그렇지 않은 경우 false를 반환합니다.
()(괄호)	하나 이상의 매개 변수를 그룹화하거나 연속적으로 표현식 평가를 수행하거나 또는 하나 이상의 매개 변수를 묶은 다음 괄호 밖의 함수에 매개 변수로서 전달합니다.
""(문자열 구분 기호 연산자)	0개 이상의 문자 앞과 뒤에 큰 따옴표가 있으면 이는 문자에 리터럴 값이 있으며 이 문자는 변수나 숫자값 또는 다른 ActionScript 요소가 아닌 문자열로 인식된다는 의미를 갖습니다.
eq(문자열 항등 연산자)	두 표현식의 항등 여부를 비교한 후 <i>expression1</i> 의 문자열 표현이 <i>expression2</i> 의 문자열 표현과 동일하면 true를 반환하고, 그렇지 않으면 false를 반환하는 비교합니다.
gt(보다 큼 문자열 연산자)	<i>expression1</i> 의 문자열 표현과 <i>expression2</i> 의 문자열 표현을 비교하여 <i>expression1</i> 이 <i>expression2</i> 보다 크면 true를 반환하고, 그렇지 않으면 false를 반환합니다.
ge(보다 크거나 같음 문자열 연산자)	<i>expression1</i> 의 문자열 표현을 <i>expression2</i> 의 문자열 표현과 비교하여 <i>expression1</i> 이 <i>expression2</i> 보다 크거나 같으면 true 값을 반환하고, 그렇지 않으면 false 값을 반환하는 비교 연산자입니다.
ne(문자열 비항등 연산자)	<i>expression1</i> 의 문자열 표현을 <i>expression2</i> 와 비교하여 <i>expression1</i> 이 <i>expression2</i> 와 같지 않으면 true를 반환하고, 같으면 false를 반환하는 비교 연산자입니다.
lt(보다 작음 문자열 연산자)	<i>expression1</i> 의 문자열 표현을 <i>expression2</i> 의 문자열 표현과 비교하여 <i>expression1</i> 이 <i>expression2</i> 보다 작으면 true 값을 반환하고, 그렇지 않으면 false 값을 반환하는 비교 연산자입니다.
le(보다 작거나 같음 문자열 연산자)	<i>expression1</i> 의 문자열 표현을 <i>expression2</i> 의 문자열 표현과 비교하여 <i>expression1</i> 이 <i>expression2</i> 보다 작거나 같으면 true 값을 반환하고, 그렇지 않으면 false 값을 반환하는 비교 연산자입니다.
-(빼기 연산자)-	부정하거나 빼는 데 사용됩니다.
==(빼기 대입)	<i>expression1</i> 에 <i>expression1 - expression2</i> 의 값을 대입합니다.

add(문자열 결합 연산자)

지원 장치
Flash Lite 1.0.

구문
`string1 add string2`

피연산자
`string1, string2` 문자열.

설명
둘 이상의 문자열을 연결하거나 결합하는 연산자입니다.

예제
다음 예제에서는 두 문자열 값을 결합하여 문자열 **catalog**를 생성합니다.

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

참고 사항
[+\(숫자 더하기 연산자\)](#)

+=(더하기 대입)

지원 장치
Flash Lite 1.0.

구문
`expression1 += expression2`

피연산자
`expression1, expression2` 숫자 또는 문자열.

설명
`expression1 + expression2`의 값을 `expression1`에 대입하는 연산자(산술 복합 대입)입니다. 예를 들어, 다음 두 명령문의 결과는 동일합니다.

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

더하기(+) 연산자의 모든 규칙은 더하기 대입(+=) 연산자에 적용됩니다.

예제
다음 예제에서는 선행 증가 대입 연산자(+=)를 사용하여 y 값에 따라 x 값을 증가시킵니다.

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

참고 사항

[+\(숫자 더하기 연산자\)](#)

및

지원 장치

Flash Lite 1.0.

구문

```
condition1 and condition2
```

피연산자

condition1, condition2 true 또는 false로 평가되는 조건이나 표현식입니다.

설명

논리 AND 연산을 수행하는 연산자입니다.

예제

다음은 and 연산자를 사용하여 플레이어가 게임에서 이겼는지 테스트하는 예제입니다. 게임을 진행하는 동안 플레이어 차례가 되거나 플레이어가 점수를 올리는 경우 turns 변수 및 score 변수가 업데이트됩니다. 다음 스크립트는 플레이어가 3번의 기회 이내에 75점 이상을 올리면 "You Win the Game!"을 출력 패널에 표시합니다.

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

참고 사항

[&&\(논리 AND\)](#)

=(대입)

지원 장치

Flash Lite 1.0.

구문

```
expression1 = expression2
```

피연산자

expression1 변수 또는 속성입니다.

expression2 값입니다.

설명

expression2의 값(오른쪽 피연산자)을 **expression1**의 변수 또는 속성에 대입하는 연산자입니다.

예제

다음은 대입 연산자(=)를 사용하여 숫자 값을 변수 **weight**에 대입하는 예제입니다.

```
weight = 5;
```

다음은 대입 연산자(=)를 사용하여 문자열 값을 변수 **greeting**에 대입하는 예제입니다.

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

/*(블록 주석 연산자)

지원 장치

Flash Lite 1.0

구문

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

피연산자

comment 모든 문자입니다.

설명

한 행 이상의 스크립트 주석을 나타내는 주석 구분 기호입니다. 열기 주석 태그(/*)와 닫기 주석 태그(*/) 사이에 존재하는 모든 문자는 주석으로 해석되며 ActionScript 인터프리터에서 무시됩니다.

단일 행 주석을 식별하려면 // (주석 구분 기호)를 사용합니다. 연속되는 여러 행에서 주석을 식별하려면 /* 주석 구분 기호를 사용합니다. 이러한 형식의 주석 구분 기호를 사용하는 경우 닫기 태그(*/)를 생략하면 오류 메시지가 반환됩니다. 주석을 중첩하는 경우에도 오류 메시지가 반환됩니다.

열기 주석 태그(/*)를 사용한 후에 첫 번째 닫기 주석 태그(*/)에 의해 주석이 종료됩니다. 이들 사이에 사용된 열기 주석 태그(*/)의 수와는 관계가 없습니다.

참고 사항

[// \(comment\)](#)

,(침표)

지원 장치

Flash Lite 1.0.

구문

```
expression1, expression2
```

피연산자

expression1, expression2 숫자 또는 숫자로 평가되는 표현식입니다.

설명

expression1과 **expression2**를 차례로 평가한 후 **expression2**의 결과를 반환하는 연산자입니다.

예제

다음 예제에서는 괄호() 연산자 없이 쉼표 연산자(,)를 사용하고 쉼표 연산자가 괄호() 연산자 없이 첫 번째 표현식 값만 반환하는 과정을 보여 줍니다.

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

다음 예제에서는 괄호() 연산자와 쉼표 연산자(,)를 함께 사용하고 쉼표 연산자가 괄호() 연산자와 함께 사용될 때 마지막 표현식의 값을 반환하는 과정을 보여 줍니다.

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

다음 예제에서는 괄호() 연산자 없이 쉼표 연산자(,)를 사용하고 쉼표 연산자가 모든 표현식을 순차적으로 평가하지만 첫 번째 표현식의 값을 반환하는 과정을 보여 줍니다. 두 번째 표현식 z++는 평가되며 z는 1씩 증가합니다.

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

다음 예제에서는 괄호() 연산자의 추가를 제외하고 이전 예제와 동일하며 괄호() 연산자와 함께 사용될 때 쉼표 연산자(,)가 연속된 마지막 표현식의 값을 반환하는 과정을 다시 보여 줍니다.

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

참고 사항

[for, \(\)\(괄호\)](#)

// (comment)

지원 장치

Flash Lite 1.0

구문

```
// comment
```

피연산자

comment 모든 문자입니다.

설명

스크립트 주석의 시작을 나타내는 주석 구분 기호입니다. 주석 기호(//)와 행의 끝 문자 사이에 있는 모든 문자는 주석으로 해석되며 ActionScript 인터프리터에서 무시됩니다.

예제

다음은 주석 구분 기호를 사용하여 첫 번째, 세 번째, 다섯 번째 및 일곱 번째 행을 주석으로 나타내는 예제입니다.

```
// Record the X position of the ball movie clip.
ballX = ball._x;
// Record the Y position of the ball movie clip.
ballY = ball._y;
// Record the X position of the bat movie clip.
batX = bat._x;
// Record the Y position of the bat movie clip.
batY = bat._y;
```

참고 사항

/*(블록 주석 연산자)

?:(조건부)

지원 장치

Flash Lite 1.0.

구문

```
expression1 ? expression2 : expression3
```

피연산자

expression1 부울 값으로 평가되는 표현식이며 주로 $x < 5$ 와 같은 비교 표현식입니다.

expression2, expression3 모든 유형의 값입니다.

설명

expression1을 평가한 후 expression1의 값이 true이면 expression2의 값을 반환하고, 그렇지 않으면 expression3의 값을 반환하는 연산자입니다.

예제

다음 예제에서는 expression1이 true로 평가되므로 변수 x의 값을 변수 z에 대입합니다.

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

-- (감소 연산자)

지원 장치

Flash Lite 1.0.

구문

```
--expression
```

```
expression--
```

피연산자

없음

설명

expression에서 1을 빼는 선행 감소 및 후행 감소 단항 산술 연산자입니다. 선행 감소 형식의 연산자(**--expression**)는 **expression**에서 1을 뺀 후 그 결과를 반환합니다. 후행 감소 형식의 연산자(**expression--**)는 **expression**에서 1을 뺀 후 **expression**의 초기값(빼기 이전의 값)을 반환합니다.

예제

다음 예제에서 선행 감소 형식의 연산자는 **aWidth**를 2로 감소시키고(**aWidth - 1 = 2**) 그 결과값을 **bWidth**로 반환합니다.

```
aWidth = 3;
bWidth = --aWidth;
// The bWidth value is equal to 2.
```

다음 예제에서 후행 감소 형식의 연산자는 **aWidth**를 2로 감소시키고(**aWidth - 1 = 2**) **aWidth**의 초기값을 결과값 **bWidth**로 반환합니다.

```
aWidth = 3;
bWidth = aWidth--;
// The bWidth value is equal to 3.
```

/(나누기 연산자)

지원 장치

Flash Lite 1.0.

구문

```
expression1 / expression2
```

피연산자

expression1, **expression2** 숫자 또는 숫자로 평가되는 표현식입니다.

설명

expression1을 **expression2**로 나누는 산술 연산자입니다. 나누기 연산의 결과는 배정밀도 부동 소수점 숫자입니다.

예제

다음 명령문은 부동 소수점 숫자 22.0을 7.0으로 나눈 다음 그 결과를 출력 패널에 표시합니다.

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

결과는 부동 소수점 숫자 3.1429입니다.

/=(나누기 대입)

지원 장치

Flash Lite 1.0.

구문

```
expression1 /= expression2
```

피연산자

expression1, expression2 숫자 또는 숫자로 평가되는 표현식입니다.

설명

expression1/expression2의 값을 **expression1**에 대입하는 산술 복합 대입 연산자입니다. 예를 들어, 다음 두 명령문은 동일합니다.

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

예제

다음은 /= 연산자에 변수와 숫자를 사용하는 예제입니다.

```
x = 10;
y = 2;
x /= y;
// The expression x now contains the value 5.
```

. (도트)

지원 장치

Flash Lite 1.0.

구문

```
instancename.variable
```

```
instancename.childinstance.variable
```

피연산자

instancename 동영상 클립의 인스턴스 이름입니다.

childinstance 다른 동영상 클립의 자식이거나 중첩된 동영상 클립의 인스턴스입니다.

variable 지정된 동영상 클립 인스턴스 이름의 타임라인 변수입니다.

설명

중첩된 자식 동영상 클립, 변수 또는 속성에 액세스하기 위해 동영상 클립 계층을 탐색하는 데 사용되는 연산자입니다.

예제

다음 예제에서는 동영상 클립 person_mc에서 변수 hairColor의 현재 값을 식별합니다.

```
person_mc.hairColor
```

이 명령문은 다음 슬래시 표기법 구문과 동일합니다.

```
/person_mc:hairColor
```

참고 사항

[/\(슬래시\)](#)

++(증가)

지원 장치
Flash Lite 1.0.

구문
`++expression`
`expression++`

피연산자
없음

설명
`expression`에 1을 더하는 선행 증가 및 후행 증가 단항 산술 연산자입니다. `expression`은 변수, 배열 요소 또는 객체의 속성이 될 수 있습니다. 선행 증가 형식의 연산자(`++expression`)는 `expression`에 1을 더한 후 그 결과를 숫자로 반환합니다. 후행 증가 형식의 연산자(`expression++`)는 `expression`에 1을 더한 후 `expression`의 초기값(더하기 이전의 값)을 반환합니다.

예제
다음 예제에서는 ++가 후행 증가 연산자로 사용되어 while 루프를 다섯 번 실행합니다.

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

다음 예제에서는 ++가 선행 증가 연산자로 사용됩니다.

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

이 스크립트는 출력 패널에 결과를 표시합니다.

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

다음 예제에서는 ++가 후행 증가 연산자로 사용됩니다.

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

이 스크립트는 출력 패널에 결과를 표시합니다.

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

&&(논리 AND)

지원 장치
Flash Lite 1.0.

구문

```
expression1 && expression2
```

피연산자

expression1, expression2 부울 값 또는 부울 값으로 변환되는 표현식입니다.

설명

두 표현식 또는 한 표현식의 값에 대해 부울 연산을 수행하는 논리 연산자입니다. 연산자의 왼쪽 표현식인 **expression1**을 평가한 후 이 표현식이 false이면 false를 반환합니다. **expression1**이 true로 평가되면 **expression2**(연산자 오른쪽의 표현식)가 평가됩니다. **expression2**가 true로 평가되면 최종 결과는 true이고, 그렇지 않으면 false입니다.

예제

다음은 플레이어가 게임을 이겼는지 결정하기 위해 && 연산자를 사용하여 테스트하는 예제입니다. 게임을 진행하는 동안 플레이어 차례가 되거나 플레이어가 점수를 올리는 경우 turns 변수 및 score 변수가 업데이트됩니다. 다음 스크립트는 플레이어가 3번의 기회 이내에 75점 이상을 올리면 "You Win the Game!"을 출력 패널에 표시합니다.

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

다음 예제에서는 가상의 x 위치가 범위 내에 있는지 확인하기 위한 테스트 방법을 보여 줍니다.

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

!(논리 NOT)

지원 장치

Flash Lite 1.0.

구문

```
!expression
```

피연산자

없음

설명

변수 또는 표현식의 부울 값을 반전시키는 논리 연산자입니다. **expression**이 절대값을 포함한 변수이거나 변환값이 true인 경우 **!expression**값은 false입니다. 표현식 **x && y**가 false로 평가되면 표현식 **!(x && y)**는 true로 평가됩니다.

다음 표현식은 !연산자의 사용 결과를 보여 줍니다. 연산자:

!true는 false를 반환합니다.

!false는 true를 반환합니다.

예제

다음 예제에서 변수 `happy`는 `false`로 설정됩니다. `if` 조건은 조건 `!happy`를 평가하며, 이 조건이 `true`인 경우 `trace()` 함수는 문자열을 출력 패널에 전달합니다.

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

||(논리 OR)

지원 장치

Flash Lite 1.0.

구문

```
expression1 || expression2
```

피연산자

`expression1`, `expression2` 부울 값 또는 부울 값으로 변환되는 표현식입니다.

설명

`expression1`과 `expression2`를 평가하는 논리 연산자입니다. 표현식 중 하나 또는 모두가 `true`로 평가되면 그 결과는 `true`이며 표현식 모두 `false`로 평가되는 경우에만 그 결과가 `false`입니다. 논리 OR 연산자와 다른 피연산자를 함께 사용할 수 있으며 피연산자가 `true`로 평가되면 그 결과는 `true`입니다.

부울이 아닌 표현식의 경우 Flash는 논리 OR 연산자로 인해 왼쪽의 표현식을 평가합니다. 표현식이 `true`로 변환될 수 있으면 그 결과는 `true`입니다. 그렇지 않으면 Flash는 오른쪽의 표현식을 평가하며 그 결과는 표현식의 값이 됩니다.

예제

다음 예제에서는 `if` 문에 `||` 연산자를 사용합니다. 두 번째 표현식이 `true`로 평가되므로 최종 결과는 `true`입니다.

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

%(모듈러스)

지원 장치

Flash Lite 1.0.

구문

```
expression1 % expression2
```

피연산자

`expression1`, `expression2` 숫자 또는 숫자로 평가되는 표현식입니다.

설명

expression1을 **expression2**로 나눈 나머지를 계산하는 산술 연산자입니다. **expression** 피연산자가 숫자가 아닌 경우 모듈러스 연산자는 피연산자를 숫자로 변환하려 합니다. **expression**은 숫자 또는 숫자 값으로 변환되는 문자열이 될 수 있습니다.

Flash Lite 1.0 또는 1.1을 대상으로 할 때 Flash 컴파일러는 다음 수식을 사용하여 제작된 SWF 파일에 % 연산자를 확장합니다.

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

이 유사 항목의 성능은 기본적으로 모듈러스 연산자를 지원하는 Flash Player의 버전 만큼 빠르거나 정확하지 않을 수 있습니다.

예제

다음 코드는 모듈러스(%) 연산자를 사용하는 숫자의 예제를 보여 줍니다.

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

%=(모듈러스 대입)

지원 장치

Flash Lite 1.0.

구문

```
expression1 %= expression2
```

피연산자

expression1, **expression2** 숫자 또는 숫자로 평가되는 표현식입니다.

설명

expression1% expression2의 값을 **expression1**에 대입하는 산술 복합 대입 연산자입니다. 예를 들어, 다음 두 표현식은 동일합니다.

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

예제

다음 예제에서는 변수 **x**에 4을 대입합니다.

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

참고 사항

[%\(모듈러스\)](#)

*=(곱하기 대입)

지원 장치

Flash Lite 1.0.

구문

```
expression1 *= expression2
```

피연산자

expression1, expression2 숫자 또는 숫자로 평가되는 표현식입니다.

설명

expression1 * expression2의 값을 expression1에 대입하는 산술 복합 대입 연산자입니다.

예를 들어, 다음 두 표현식은 동일합니다.

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

예제

구문 1: 다음은 값 50을 변수 x에 대입하는 예제입니다.

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

구문 2: 다음 예제의 두 번째 행과 세 번째 행에서는 항등 부호(=) 오른쪽의 표현식을 계산한 후 그 결과를 x와 y에 대입합니다.

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

*(곱하기 연산자)

지원 장치

Flash Lite 1.0.

구문

```
expression1 * expression2
```

피연산자

expression1, expression2 숫자 표현식

설명

두 숫자 표현식을 곱하는 산술 연산자입니다. 두 표현식이 모두 정수인 경우에는 곱한 값도 정수이고 두 표현식 중 하나라도 부동 소수점 숫자인 경우에는 곱한 값도 부동 소수점 숫자입니다.

예제

구문 1: 다음 명령문은 정수 2와 3을 곱합니다.

```
2 * 3
```

결과는 정수 6입니다.

구문 2: 다음 명령문은 부동 소수점 숫자 2.0과 3.1416을 곱합니다.

```
2.0 * 3.1416
```

결과는 부동 소수점 숫자 6.2832입니다.

+(숫자 더하기 연산자)

지원 장치
Flash Lite 1.0.

구문
`expression1 + expression2`

피연산자
`expression1`, `expression2` 숫자.

설명
숫자 표현식을 더하는 연산자입니다. +는 숫자 연산자로만 사용되며 문자열 결합에는 사용될 수 없습니다.
두 표현식이 정수인 경우 그 합도 정수이며, 한 표현식 또는 두 표현식이 모두 부동 소수점 숫자인 경우 그 합도 부동 소수점 숫자입니다.

예제
다음은 정수 2와 3을 더하고 그 결과인 정수 5를 출력 패널에 나타내는 예제입니다.

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

다음은 부동 소수점 숫자 2.5와 3.25를 더하고 그 결과인 부동 소수점 숫자 5.75를 출력 패널에 나타내는 예제입니다.

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

참고 사항
[add\(문자열 결합 연산자\)](#)

==(숫자 항등 연산자)

지원 장치
Flash Lite 1.0.

구문
`expression1 == expression2`

피연산자
`expression1`, `expression2` 숫자, 부울 값 또는 변수입니다.

설명
<> 연산자의 정반대 상황인 항등 여부를 테스트하는 비교 연산자입니다. `expression1`과 `expression2`가 동일한 경우 그 결과는 `true`입니다. <> 연산자에서와 마찬가지로 항등에 대한 정의는 비교되는 데이터 유형에 따라 다릅니다.

- 숫자 및 부울 값은 값을 기준으로 비교됩니다.
- 변수는 참조를 기준으로 비교됩니다.

예제
다음 예제에서는 `true` 및 `false` 반환값을 보여 줍니다.

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

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

참고 사항

[eq\(문자열 항등 연산자\)](#)

> (보다 큼 숫자 연산자)

지원 장치

Flash Lite 1.0.

구문

```
expression1 > expression2
```

피연산자

expression1, expression2 숫자 또는 숫자로 평가되는 표현식입니다.

설명

두 표현식을 비교하여 **expression1**이 **expression2**보다 큰지 여부를 결정하는 비교 연산자입니다. 큰 경우 이 연산자는 **true**를 반환합니다. **expression1**이 **expression2**보다 작거나 같은 경우 이 연산자는 **false**를 반환합니다.

예제

다음 예제에서는 숫자 비교에 대한 **true** 및 **false** 결과를 보여 줍니다.

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

참고 사항

[gt\(보다 큼 문자열 연산자\)](#)

>=(보다 크거나 같음 숫자 연산자)

지원 장치

Flash Lite 1.0.

구문

```
expression1 >= expression2
```

피연산자

expression1, expression2 정수 또는 부동 소수점 숫자입니다.

설명

두 표현식을 비교하여 **expression1**이 **expression2**보다 크거나 같은 경우 true를 반환하며, **expression1**이 **expression2**보다 작은 경우 false를 반환하는 비교 연산자입니다.

예제

다음 예제에서는 true 및 false 결과를 보여 줍니다.

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

참고 사항

[ge\(보다 크거나 같음 문자열 연산자\)](#)

<>(비항등 숫자 연산자)

지원 장치

Flash Lite 1.0.

구문

```
expression1 <> expression2
```

피연산자

expression1, **expression2** 숫자, 부울 값 또는 변수입니다.

설명

항등(==) 연산자의 정반대 상황인 비항등 여부를 테스트하는 비교 연산자입니다. **expression1**이 **expression2**와 같으면 결과는 false입니다. 항등(==) 연산자에서와 마찬가지로 항등에 대한 정의는 비교되는 데이터 유형에 따라 다릅니다.

- 숫자 및 부울 값은 값을 기준으로 비교됩니다.
- 변수는 참조를 기준으로 비교됩니다.

예제

다음 예제에서는 true 및 false 반환을 보여 줍니다.

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

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

참고 사항

[ne\(문자열 비항등 연산자\)](#)

< (보다 작음 숫자)

지원 장치

Flash Lite 1.0.

구문`expression1 < expression2`**피연산자**

expression1, expression2 숫자.

설명

두 표현식을 비교하여 **expression1**이 **expression2**보다 작은지 여부를 결정하는 비교 연산자입니다. 작은 경우 연산자는 **true**를 반환합니다. **expression1**이 **expression2**보다 크거나 같은 경우 이 연산자는 **false**를 반환합니다. <(보다 작음) 연산자는 숫자 연산자입니다.

예제

다음 예제에서는 숫자 비교 및 문자열 비교에 대한 **true** 및 **false** 결과를 보여줍니다.

```
trace (3 < 10); // output: 1(true)
trace (10 < 3); // output: 0(false)
```

참고 사항

[lt\(보다 작음 문자열 연산자\)](#)

<=(보다 작거나 같음 숫자 연산자)

지원 장치

Flash Lite 1.0.

구문`expression1 <= expression2`**피연산자**

expression1, expression2 숫자.

설명

두 표현식을 비교하여 **expression1**이 **expression2**보다 작거나 같은지 여부를 확인하는 비교 연산자입니다. 작거나 같은 경우 연산자는 **true**를 반환하며, 그렇지 않은 경우 **false**를 반환합니다. 이 연산자는 숫자 비교에만 사용됩니다.

예제

다음 예제에서는 숫자 비교에 대한 **true** 및 **false** 결과를 보여 줍니다.

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

참고 사항

[le\(보다 작거나 같음 문자열 연산자\)](#)

()(괄호)

지원 장치
Flash Lite 1.0.

구문

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

expression1, expression2 숫자, 문자열, 변수 또는 텍스트입니다.

피연산자

parameter1,..., parameterN 매개 변수를 사용하여 연산 결과를 괄호 밖의 함수로 전달하기 전에 실행되는 일련의 매개 변수입니다.

설명

하나 이상의 매개 변수를 그룹화하고, 연속적으로 표현식 평가를 수행하거나 또는 하나 이상의 매개 변수를 묶은 다음 괄호 밖의 함수에 매개 변수로서 전달하는 연산자입니다.

구문 1: 표현식에서 연산자가 실행되는 순서를 제어합니다. 괄호를 사용하면 일반적인 우선 순위가 무시되고 괄호 안의 표현식이 먼저 평가됩니다. 괄호가 중첩된 경우에는 가장 안쪽 괄호의 콘텐츠가 바깥쪽 괄호의 콘텐츠보다 먼저 평가됩니다.

구문 2: 쉼표로 구분된 일련의 표현식을 순서대로 평가하고 최종 표현식의 결과를 반환합니다.

예제

구문 1: 다음 명령문은 괄호를 사용하여 표현식의 실행 순서를 제어하는 방법을 보여줍니다. 각 표현식의 값은 출력 패널에 표시됩니다.

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

구문 1: 다음 명령문은 표현식이 실행되는(각 표현식의 값은 로그 파일에 작성됨) 순서를 제어하기 위해 괄호를 사용하는 방법을 보여 줍니다.

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

" "(문자열 구분 기호 연산자)

지원 장치
Flash Lite 1.0.

구문

```
"text"
```

피연산자

text 0개 이상의 문자입니다.

설명

문자열 구분 기호입니다. 0개 이상의 문자 앞과 뒤에 큰 따옴표가 있으면 이는 문자에 리터럴 값이 있으며 이 문자는 변수나 숫자값 또는 다른 ActionScript 요소가 아닌 문자열로 인식된다는 의미를 갖습니다.

예제

이 예제에서는 큰 따옴표를 사용하여 `yourGuess` 변수의 값이 리터럴 문자열 `Prince Edward Island`이며 변수의 이름이 아니라는 것을 나타냅니다. `province`의 값은 리터럴이 아니라 변수입니다. `province`의 값을 확인하려면 `yourGuess`의 값을 찾아야 합니다.

```
yourGuess = "Prince Edward Island";

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

eq(문자열 항등 연산자)

지원 장치

Flash Lite 1.0.

구문

```
expression1 eq expression2
```

피연산자

`expression1`, `expression2` 숫자, 문자열 또는 변수입니다.

설명

두 표현식의 항등 여부를 비교한 후 `expression1`의 문자열 표현이 `expression2`의 문자열 표현과 동일하면 `true`를 반환하고, 그렇지 않으면 `false`를 반환하는 연산자(비교)입니다.

예제

다음 예제에서는 `true` 및 `false` 결과를 보여 줍니다.

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

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

참고 사항

[==\(숫자 항등 연산자\)](#)

gt(보다 큼 문자열 연산자)

지원 장치

Flash Lite 1.0.

구문

```
expression1 gt expression2
```

피연산자

expression1, expression2 숫자, 문자열 또는 변수입니다.

설명

expression1의 문자열 표현을 expression2의 문자열 표현과 비교하여 expression1이 expression2보다 크면 true 값을 반환하고, 그렇지 않으면 false 값을 반환하는 비교 연산자입니다. 문자열은 알파벳 순서로 비교되며, 문자보다 숫자가 앞에 오고 소문자보다 대문자가 앞에 옵니다.

예제

다음 예제에서는 true 및 false 결과를 보여 줍니다.

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

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

참고 사항

> [\(보다 큼 숫자 연산자\)](#)

ge(보다 크거나 같음 문자열 연산자)

지원 장치

Flash Lite 1.0.

구문

```
expression1 ge expression2
```

피연산자

expression1, expression2 숫자, 문자열 또는 변수입니다.

설명

expression1의 문자열 표현을 expression2의 문자열 표현과 비교하여 expression1이 expression2보다 크거나 같으면 true 값을 반환하고, 그렇지 않으면 false 값을 반환하는 비교 연산자입니다. 문자열은 알파벳 순서로 비교되며, 문자보다 숫자가 앞에 오고 소문자보다 대문자가 앞에 옵니다.

예제

다음 예제에서는 true 및 false 결과를 보여 줍니다.

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

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

참고 사항

>=(보다 크거나 같음 숫자 연산자)

ne(문자열 비항등 연산자)

지원 장치

Flash Lite 1.0.

구문

```
expression1 ne expression2
```

피연산자

expression1, expression2 숫자, 문자열 또는 변수입니다.

설명

expression1의 문자열 표현을 expression2와 비교하여 expression1이 expression2와 같지 않으면 true를 반환하고, 같으면 false를 반환하는 비교 연산자입니다.

예제

다음 예제에서는 true 및 false 결과를 보여 줍니다.

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

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

참고 사항

<>(비항등 숫자 연산자)

lt(보다 작음 문자열 연산자)

지원 장치

Flash Lite 1.0.

구문

```
expression1 lt expression2
```

피연산자

expression1, **expression2** 숫자, 문자열 또는 변수입니다.

설명

expression1의 문자열 표현을 **expression2**의 문자열 표현과 비교하여 **expression1**이 **expression2**보다 작으면 **true** 값을 반환하고, 그렇지 않으면 **false** 값을 반환하는 비교 연산자입니다. 문자열은 알파벳 순서로 비교되며, 문자보다 숫자가 앞에 오고 소문자보다 대문자가 앞에 옵니다.

예제

다음 예제에서는 다양한 문자열 비교 결과를 보여 줍니다. ActionScript 1.0 구문은 문자열과 정수를 비교할 때 정수 데이터 유형을 문자열로 변환하고 **false**를 반환하기 때문에 문자열을 정수와 비교하는 마지막 행에서 **lt**는 오류를 반환하지 않습니다.

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

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

참고 사항

[<\(보다 작음 숫자\)](#)

le(보다 작거나 같음 문자열 연산자)

지원 장치

Flash Lite 1.0.

구문

```
expression1 le expression2
```

피연산자

expression1, **expression2** 숫자, 문자열 또는 변수입니다.

설명

expression1의 문자열 표현을 **expression2**의 문자열 표현과 비교하여 **expression1**이 **expression2**보다 작거나 같으면 **true** 값을 반환하고, 그렇지 않으면 **false** 값을 반환하는 비교 연산자입니다. 문자열은 알파벳 순서로 비교되며, 문자보다 숫자가 앞에 오고 소문자보다 대문자가 앞에 옵니다.

예제

다음 예제에서는 다양한 문자열 비교 결과를 보여 줍니다.

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

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

참고 사항

<=(보다 작거나 같은 숫자 연산자)

- (빼기 연산자)

지원 장치

Flash Lite 1.0.

구문

(부정) *-expression*

(빼기) *expression1 - expression2*

피연산자

expression1, *expression2* 숫자 또는 숫자로 평가되는 표현식입니다.

설명

부정하거나 빼는 데 사용되는 산술 연산자입니다.

구문 1: 이 구문이 부정에 사용되는 경우 숫자 *expression*의 부호를 반전시킵니다.

구문 2: 빼기에 사용되면 두 개의 숫자 표현식에서 산술 빼기를 수행하여 *expression1*에서 *expression2*를 뺍니다. 두 표현식이 모두 정수인 경우에는 빼기 결과도 정수이고 두 표현식 중 하나라도 부동 소수점 숫자인 경우에는 빼기 결과도 부동 소수점 숫자입니다.

예제

구문 1: 다음 명령문은 2 + 3 표현식의 부호를 반대로 합니다.

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

구문 2: 다음 명령문은 정수 5에서 정수 2를 뺍니다.

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

결과는 정수 3입니다.

구문 3: 다음 명령문은 부동 소수점 숫자 3.25에서 부동 소수점 숫자 1.5를 뺍니다.

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

결과는 부동 소수점 숫자 1.75입니다.

-=(빼기 대입)

지원 장치
Flash Lite 1.0.

구문
`expression1 -= expression2`

피연산자
`expression1`, `expression2` 숫자 또는 숫자로 평가되는 표현식입니다.

설명
`expression1-expression2`의 값을 `expression1` 에 대입하는 산술 복합 대입 연산자입니다. 반환되는 값이 없습니다.

예를 들어, 다음 두 명령문은 동일합니다.

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

문자열 표현식은 숫자로 변환되어야 하며, 그렇지 않은 경우 -1이 반환됩니다.

예제

구문 1: 다음 예제에서는 -= 연산자를 사용하여 5에서 10을 뺀 후 그 결과를 변수 x에 대입합니다.

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

구문 2: 다음 예제에서는 문자열을 숫자로 변환하는 방법을 보여줍니다.

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

6장: Flash Lite 특정 언어 요소

이 단원에서는 Adobe의 Macromedia Flash Lite 1.1 소프트웨어가 인식하는 플랫폼 기능과 변수, 그리고 `fscommand()` 및 `fscommand2()` 함수를 사용해서 실행할 수 있는 Flash Lite 명령을 설명합니다. 이 단원에서 설명된 기능은 Flash Lite에 한정된 것입니다.

이 단원의 내용은 다음 표에 요약되어 있습니다.

언어 요소	설명
<code>_capCompoundSound</code>	Flash Lite가 복합 사운드를 처리할 수 있는지 여부를 나타냅니다.
<code>_capEmail</code>	Flash Lite 클라이언트가 <code>GetURL()</code> ActionScript 명령을 사용하여 전자 메일 메시지를 보낼 수 있는지 여부를 나타냅니다.
<code>_capLoadData</code>	호스트 응용 프로그램에서 <code>loadMovie()</code> , <code>loadMovieNum()</code> , <code>loadVariables()</code> 및 <code>loadVariablesNum()</code> 함수 호출을 통해 추가 데이터를 동적으로 로드할 수 있는지 여부를 나타냅니다.
<code>_capMFi</code>	장치에서 MFi(Melody Format for i-mode) 오디오 형식으로 사운드 데이터를 재생할 수 있는지 여부를 나타냅니다.
<code>_capMIDI</code>	장치에서 MIDI(Musical Instrument Digital Interface) 오디오 형식으로 사운드 데이터를 재생할 수 있는지 여부를 나타냅니다.
<code>_capMMS</code>	Flash Lite가 <code>GetURL()</code> ActionScript 명령을 사용하여 MMS(Multimedia Messaging Service) 메시지를 보낼 수 있는지 여부를 나타냅니다.
<code>_capMP3</code>	장치에서 MPEGAudio Layer 3(MP3) 오디오 형식으로 사운드 데이터를 재생할 수 있는지 여부를 나타냅니다.
<code>_capSMAF</code>	장치에서 SMAF(Synthetic music Mobile Application Format)로 멀티미디어 파일을 재생할 수 있는지 여부를 나타냅니다.
<code>_capSMS</code>	Flash Lite가 <code>GetURL()</code> ActionScript 명령을 사용하여 SMS(Short Message Service) 메시지를 보낼 수 있는지 여부를 나타냅니다.
<code>_capStreamSound</code>	장치에서 스트리밍(동기식) 사운드를 재생할 수 있는지 여부를 나타냅니다.
<code>_cap4WayKeyAS</code>	Flash Lite가 오른쪽, 왼쪽, 위쪽, 아래쪽 화살표 키와 관련된 키 이벤트 핸들러에 첨부되어 있는 ActionScript 표현식을 실행하는지 여부를 나타냅니다.
<code>\$version</code>	Flash Lite의 버전 번호를 포함합니다.
<code>fscommand()</code>	Launch 명령을 실행하기 위해 사용하는 함수입니다(다음 항목 참조).
<code>Launch</code>	(<code>fscommand()</code> 에 지원되는 유일한 명령) SWF 파일이 Flash Lite 또는 전화기나 장치의 운영 체제와 같은 호스트 환경과 통신할 수 있도록 합니다.
<code>fscommand2()</code>	이 표에 있는 명령을 실행하기 위해 사용하는 함수이며 <code>fscommand()</code> 는 제외됩니다.
<code>Escape</code>	임의의 문자열을 네트워크 전송에 안전한 형식으로 인코딩합니다.
<code>FullScreen</code>	렌더링에 사용할 디스플레이 영역 크기를 설정합니다.
<code>GetBatteryLevel</code>	현재 배터리 수준을 반환합니다.
<code>GetDateDay</code>	현재 날짜의 일을 숫자 값으로 반환합니다.
<code>GetDateMonth</code>	현재 날짜의 월을 숫자 값으로 반환합니다.

언어 요소	설명
<code>getDateWeekday</code>	현재 날짜의 요일을 숫자 값으로 반환합니다.
<code>getDateYear</code>	현재 날짜의 연도를 4자리 숫자 값으로 반환합니다.
<code>getDevice</code>	Flash Lite가 실행 중인 장치를 식별하는 매개 변수를 설정합니다.
<code>getDeviceID</code>	장치의 고유한 식별자(예: 일련 번호)를 나타내는 매개 변수를 설정합니다.
<code>getFreePlayerMemory</code>	현재 Flash Lite에서 사용 가능한 힙 메모리의 양(KB)을 반환합니다.
<code>getLanguage</code>	현재 장치가 사용하고 있는 언어를 식별하는 매개 변수를 설정합니다.
<code>getLocaleLongDate</code>	현재 정의된 지역에 따라 긴 형식으로 현재 날짜를 나타내는 문자열로 매개 변수를 설정합니다.
<code>getLocaleShortDate</code>	현재 정의된 지역에 따라 축약 형식으로 현재 날짜를 나타내는 문자열로 매개 변수를 설정합니다.
<code>getLocaleTime</code>	현재 정의된 지역에 따라 지정된 형식으로 현재 시간을 나타내는 문자열로 매개 변수를 설정합니다.
<code>getMaxBatteryLevel</code>	장치의 최대 배터리 수준을 반환합니다.
<code>getMaxSignalLevel</code>	최대 신호 강도 수준을 반환합니다.
<code>getMaxVolumeLevel</code>	장치의 최대 볼륨 수준을 숫자 값으로 반환합니다.
<code>getNetworkConnectStatus</code>	현재 네트워크 연결 상태를 나타내는 값을 반환합니다.
<code>getNetworkName</code>	매개 변수를 현재 네트워크의 이름으로 설정합니다.
<code>getNetworkRequestStatus</code>	가장 최근의 HTTP 요청 상태를 나타내는 값을 반환합니다.
<code>getNetworkStatus</code>	전화의 네트워크 상태, 즉 등록된 네트워크가 있는지 그리고 전화가 현재 로밍 중인지 여부를 나타내는 값을 반환합니다.
<code>getPlatform</code>	장치의 클래스를 광범위하게 기술하는, 현재 플랫폼을 식별하는 매개 변수를 설정합니다. 공개 운영 체제가 설치된 장치의 경우 이 식별자는 일반적으로 운영 체제의 이름과 버전입니다.
<code>getPowerSource</code>	배터리를 사용하는지 아니면 외부 전원을 사용하는지 여부를 나타내는 값을 반환합니다.
<code>getSignalLevel</code>	현재 신호 강도를 숫자 값으로 반환합니다.
<code>getTimeHours</code>	현재 시간의 시를 24시간 방식의 숫자 값으로 반환합니다.
<code>getTimeMinutes</code>	현재 시간의 분을 숫자 값으로 반환합니다.
<code>getTimeSeconds</code>	현재 시간의 초를 숫자 값으로 반환합니다.
<code>getTimeZoneOffset</code>	지역 시간대와 표준시(UTC) 사이의 분 값으로 매개 변수를 설정합니다.
<code>getTotalPlayerMemory</code>	Flash Lite에 할당된 총 힙 메모리 크기(KB)을 반환합니다.
<code>getVolumeLevel</code>	장치의 현재 볼륨 수준을 숫자 값으로 반환합니다.
<code>Quit</code>	Flash Lite 플레이어가 재생을 중지하고 종료하도록 합니다.
<code>ResetSoftKeys</code>	소프트 키를 원래 설정으로 복구합니다.
<code>SetInputTextType</code>	입력 텍스트 필드가 열리는 모드를 지정합니다.
<code>SetQuality</code>	애니메이션의 렌더링 품질을 설정합니다.

언어 요소	설명
SetSoftKeys	장치의 왼쪽 및 오른쪽 소프트 키가 액세스 및 재맵핑이 가능한 경우에 이것을 다시 맵핑합니다.
StartVibrate	전화의 진동 기능을 시작합니다.
StopVibrate	현재 실행 중인 진동을 중지합니다.
Unescape	네트워크 전송에 안전하도록 인코딩된 임의의 문자열을 인코딩되지 않은 일반적인 형태로 디코딩합니다.

기능

이 단원에서는 Macromedia Flash Lite 1.1 이 인식하는 플랫폼 기능 및 변수를 설명합니다. 항목은 처음에 오는 밑줄을 무시하고 알파벳 순서로 나열되어 있습니다.

_capCompoundSound

지원 장치

Flash Lite 1.1.

구문

`_capCompoundSound`

설명

Flash Lite가 복합 사운드 데이터를 처리할 수 있는지 여부를 나타내는 숫자 변수입니다. 데이터를 로드할 수 있으면 이 변수가 정의되어 값이 1로 설정되고, 그렇지 않으면 변수가 정의되지 않습니다.

예를 들어, Flash 파일 하나에 MIDI 및 MFi 형식의 동일한 사운드가 포함될 수 있습니다. 그러면 플레이어는 장치에서 지원하는 형식에 따라 적절한 형식의 데이터를 재생합니다. 이 변수는 Flash Lite 플레이어가 현재 핸드셋에서 이 기능을 지원하는지 여부를 정의합니다.

다음 예제에서 `useCompoundSound`는 Flash Lite 1.1의 경우 1로 설정되지만 Flash Lite 1.0의 경우에는 정의되지 않습니다.

```
useCompoundSound = _capCompoundSound;

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

_capEmail

지원 장치

Flash Lite 1.1.

구문

`_capEmail`

설명

Flash Lite 클라이언트가 `GetURL()` ActionScript 명령을 사용하여 전자 메일 메시지를 보낼 수 있는지 여부를 나타내는 숫자 변수입니다. 데이터를 로드할 수 있으면 이 변수가 정의되어 값이 1로 설정되고, 그렇지 않으면 변수가 정의되지 않습니다.

예제

호스트 응용 프로그램이 `GetURL()` ActionScript 명령을 사용하여 전자 메일 메시지를 보낼 수 있는 경우 다음 예제에서는 `canEmail`을 1로 설정합니다.

```
canEmail = _capEmail;

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

`_capLoadData`

지원 장치

Flash Lite 1.1.

구문

`_capLoadData`

설명

호스트 응용 프로그램에서 `loadMovie()`, `loadMovieNum()`, `loadVariables()` 및 `loadVariablesNum()` 함수 호출을 통해 추가 데이터를 동적으로 로드할 수 있는지 여부를 나타내는 숫자 변수입니다. 데이터를 로드할 수 있으면 이 변수가 정의되어 값이 1로 설정되고, 그렇지 않으면 변수가 정의되지 않습니다.

예제

호스트 응용 프로그램이 동영상 및 변수의 동적 로드를 수행할 수 있는 경우 다음 예제에서는 `iCanLoad`를 1로 설정합니다.

```
canLoad = _capLoadData;

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

`_capMFi`

지원 장치

Flash Lite 1.1.

구문

`_capMFi`

설명

장치에서 MFi(Melody Format for i-mode) 오디오 형식으로 사운드 데이터를 재생할 수 있는지 여부를 나타내는 숫자 변수입니다. 데이터를 로드할 수 있으면 이 변수가 정의되어 값이 1로 설정되고, 그렇지 않으면 변수가 정의되지 않습니다.

예제

장치가 MFi 사운드 데이터를 재생할 수 있는 경우 다음 예제에서는 canMfi를 1로 설정합니다.

```
canMFi = _capMFi;

if (canMFi == 1) {
    // send movieclip buttons to frame with buttons that trigger events sounds
    tellTarget("buttons") {
        gotoAndPlay(2);
    }
}
```

_capMIDI

지원 장치

Flash Lite 1.1.

구문

```
_capMIDI
```

설명

장치에서 MIDI(Musical Instrument Digital Interface) 오디오 형식으로 사운드 데이터를 재생할 수 있는지 여부를 나타내는 숫자 변수입니다. 데이터를 로드할 수 있으면 이 변수가 정의되어 값이 1로 설정되고, 그렇지 않으면 변수가 정의되지 않습니다.

예제

장치가 MIDI 사운드 데이터를 재생할 수 있는 경우 다음 예제에서는 canMidi를 1로 설정합니다.

```
canMIDI = _capMIDI;

if (canMIDI == 1) {
    // send movieclip buttons to frame with buttons that trigger events sounds
    tellTarget("buttons") {
        gotoAndPlay(2);
    }
}
```

_capMMS

지원 장치

Flash Lite 1.1.

구문

```
_capMMS
```

설명

Flash Lite가 GetURL() ActionScript 명령을 사용하여 MMS(Multimedia Messaging Service) 메시지를 보낼 수 있는지 나타내는 숫자 변수입니다. 데이터를 로드할 수 있으면 이 변수가 정의되어 값이 1로 설정되고, 그렇지 않으면 변수가 정의되지 않습니다.

예제

다음 예제에서는 Flash Lite 1.1의 경우 canMMS를 1로 설정하지만 Flash Lite 1.0의 경우에는 이를 정의되지 않은 상태로 둡니다. 그러나 모든 Flash Lite 1.1 전화가 MMS 메시지를 보낼 수 있는 것은 아니므로 이 코드는 전화에 따라 달라집니다.

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

_capMP3

지원 장치

Flash Lite 1.1.

구문

```
_capMP3
```

설명

장치에서 MPEGAudio Layer 3(MP3) 오디오 형식으로 사운드 데이터를 재생할 수 있는지 여부를 나타냅니다. 데이터를 로드할 수 있으면 이 변수가 정의되어 값이 1로 설정되고, 그렇지 않으면 변수가 정의되지 않습니다.

예제

장치가 MP3 사운드 데이터를 재생할 수 있는 경우 다음 예제에서는 canMP3를 1로 설정합니다.

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

_capSMAF

지원 장치

Flash Lite 1.1.

구문

```
_capSMAF
```

설명

장치에서 SMAF(Synthetic music Mobile Application Format)로 멀티미디어 파일을 재생할 수 있는지 여부를 나타내는 숫자 변수입니다. 데이터를 로드할 수 있으면 이 변수가 정의되어 값이 1로 설정되고, 그렇지 않으면 변수가 정의되지 않습니다.

예제

다음 예제에서는 Flash Lite 1.1의 경우 canSMAF를 1로 설정하지만 Flash Lite 1.0의 경우에는 이를 정의되지 않은 상태로 둡니다. 그러나 모든 Flash Lite 1.1 전화가 SMAF 메시지를 보낼 수 있는 것은 아니므로 이 코드는 전화에 따라 달라집니다.

```
canSMAF = _capSMAF;

if (canSMAF) {
    // send movieclip buttons to frame with buttons that trigger events sounds
    tellTarget("buttons") {
        gotoAndPlay(2);
    }
}
```

_capSMS

지원 장치

Flash Lite 1.1.

구문

```
_capSMS
```

설명

Flash Lite가 GetURL() ActionScript 명령을 사용하여 **SMS(Short Message Service)** 메시지를 보낼 수 있는지 여부를 나타내는 숫자 변수입니다. 데이터를 로드할 수 있으면 이 변수가 정의되어 값이 1로 설정되고, 그렇지 않으면 변수가 정의되지 않습니다.

예제

다음 예제에서는 Flash Lite 1.1의 경우 canSMS를 1로 설정하지만 Flash Lite 1.0의 경우에는 이를 정의되지 않은 상태로 둡니다. 그러나 모든 Flash Lite 1.1 전화가 SMS 메시지를 보낼 수 있는 것은 아니므로 이 코드는 전화에 따라 달라집니다.

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

_capStreamSound

지원 장치

Flash Lite 1.1.

구문

```
_capStreamSound
```

설명

장치에서 스트리밍(동기식) 사운드를 재생할 수 있는지 여부를 나타내는 숫자 변수입니다. 데이터를 로드할 수 있으면 이 변수가 정의되어 값이 1로 설정되고, 그렇지 않으면 변수가 정의되지 않습니다.

예제

다음 예제에서는 canStreamSound가 활성화된 경우 스트리밍 사운드를 재생합니다.

```
on(press) {
    canStreamSound = _capStreamSound;
    if (canStreamSound) {
        // play a streaming sound in a movieclip with this button
        tellTarget("music") {
            gotoAndPlay(2);
        }
    }
}
```

_cap4WayKeyAS

지원 장치

Flash Lite 1.1.

구문

```
_cap4WayKeyAS
```

설명

Flash Lite가 오른쪽, 왼쪽, 위쪽, 아래쪽 화살표 키와 관련된 키 이벤트 핸들러에 첨부되어 있는 ActionScript 표현식을 실행하는지 여부를 나타내는 숫자 변수입니다. 이 변수는 호스트 응용 프로그램에서 4방향 키 탐색 모드를 사용하여 Flash 컨트롤(단추 및 입력 텍스트 필드) 사이에서 이동하는 경우에만 정의되고 값이 1로 설정됩니다. 그렇지 않은 경우에는 이 변수가 정의되지 않습니다.

방향의 키 중 하나를 누르면 변수의 값이 1인 경우 Flash Lite는 먼저 해당 키에 대한 핸들러를 찾습니다. 핸들러를 찾지 못하면 Flash 컨트롤 탐색이 실행됩니다. 그러나 이벤트 핸들러를 찾으면 해당 키에 대한 탐색 액션이 발생하지 않습니다. 예를 들어, 아래쪽 화살표 키에 대한 키 누르기 핸들러를 찾은 경우에는 사용자가 탐색할 수 없습니다.

예제

다음 예제에서는 Flash Lite 1.1의 경우 canUse4Way를 1로 설정하지만 Flash Lite 1.0의 경우에는 이를 정의되지 않은 상태로 둡니다. 그러나 모든 Flash Lite 1.1 전화가 4방향 키를 지원하는 것은 아니므로 이 코드는 전화에 따라 달라집니다.

```
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

지원 장치

Flash Lite 1.1.

구문

```
$version
```

설명

Flash Lite의 버전 번호를 포함하는 문자열 변수입니다. 주 번호, 보조 번호, 빌드 번호 및 모든 출시된 버전에서 일반적으로 0인 내부 빌드 번호가 포함됩니다.

모든 Flash Lite 1.x 제품에 대해 보고되는 주 버전 번호는 5입니다. Flash Lite 1.0의 부 버전 번호는 1이고 Flash Lite 1.1의 부 버전 번호는 2입니다.

예제

Flash Lite 1.1 플레이어에서 다음 코드는 myVersion 값을 "5, 2, 12, 0"으로 설정합니다.

```
myVersion = $version;
```

fscommand()

지원 장치

Flash Lite 1.1.

구문

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

매개 변수

"Launch" 명령 구분자. Launch 명령은 fscommand() 함수를 사용해서 실행하는 유일한 명령입니다.

"application-path, arg1, arg2,..., argn" 시작되는 응용 프로그램의 이름 및 이것에 대한 매개 변수이며, 모두 쉼표로 분리됩니다.

설명

SWF 파일이 Flash Lite 또는 전화기나 장치의 운영 체제와 같은 호스트 환경과 통신할 수 있도록 하는 함수입니다.

참고 사항

[fscommand2\(\)](#)

Launch

지원 장치

Flash Lite 1.1.

구문

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

매개 변수

"Launch" 명령 구분자. Flash Lite에서, fscommand() 함수는 Launch 명령을 실행하기 위해서만 사용 가능합니다.

"application-path, arg1, arg2,..., argn" 시작되는 응용 프로그램의 이름 및 이것에 대한 매개 변수이며, 모두 쉼표로 분리됩니다.

설명

fscommand() 함수를 통해 명령이 실행되었습니다. 장치에서 다른 응용 프로그램을 시작합니다. 실행 중인 응용 프로그램의 이름과 이것에 대한 매개 변수가 단일 인수로 전달됩니다.

참고: 이 기능은 운영 체제에 의존적으로 지원됩니다.

이 명령은 Flash Lite 플레이어가 독립 실행형 모드에서 실행될 때만 지원됩니다. 브라우저의 플러그인처럼 플레이어가 다른 응용 프로그램과 관련되어 실행되는 경우에는 지원되지 않습니다.

예제

다음 예제는 Series 60 전화에서 지원되는 서비스/웹 브라우저에서 wap.yahoo.com 페이지를 엽니다.

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

참고 사항

[fscommand2\(\)](#)

fscommand2()

지원 장치

Flash Lite 1.1.

구문

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

매개 변수

명령을 사용하여 해당 구성 요소의 액세스 가능한 부분을 활성화할 수 있습니다 여러 용도로 호스트 응용 프로그램에 전달되는 문자열이거나 Flash Lite에 전달되는 명령입니다.

parameter1...parameterN 쉼표로 구분된 문자열 목록이 명령으로 구분된 명령에 대한 매개 변수로 전달됩니다.

설명

SWF 파일이 Flash Lite 또는 전화기나 장치의 운영 체제와 같은 호스트 환경과 통신할 수 있도록 하는 함수입니다. fscommand2()가 반환하는 값은 특정 명령에 따라 달라집니다.

fscommand2() 함수는 fscommand() 함수와 비슷하지만 다음과 같은 차이점이 있습니다.

- fscommand2() 함수에는 인수를 원하는 수만큼 사용할 수 있습니다.
- Flash Lite에서 fscommand2()는 즉시 실행되지만 fscommand()는 처리 중인 프레임의 끝에서 실행됩니다.
- fscommand2() 함수는 성공, 실패 또는 명령 결과를 보고하는 데 사용할 수 있는 값을 반환합니다.

함수에 명령 및 매개 변수로 보내는 문자열 및 표현식은 본 단원의 표에 설명되어 있습니다.

표에는 3개의 열이 있습니다.

- [명령] 열은 명령을 식별하는 문자열 리터럴 매개 변수를 보여줍니다.
- [매개 변수] 열은 추가 매개 변수로 전달할 값의 종류를 설명합니다(있는 경우에 한함).
- [반환 값] 컬럼은 예상되는 반환 값을 설명합니다.

예제

예제에는 fscommand2() 함수를 사용해서 실행하는 특정 명령이 제공되어 있는데, 이것은 본 단원의 후반부에 설명되어 있습니다.

참고 사항

[fscommand\(\)](#)

Escape

지원 장치

Flash Lite 1.1.

설명

임의의 문자열을 네트워크 전송에 안전한 형식으로 인코딩합니다. 알파벳이 아닌 문자를 16진수 이스케이프 시퀀스(%xx, 또는 복수 바이트 문자의 경우 %xx%xx)로 교체합니다.

명령	매개 변수	반환값
"Escape"	<p><i>original</i> URL에 안전하게 사용할 수 있는 형식으로 인코딩될 문자열입니다.</p> <p><i>encoded</i> 최종적으로 인코딩된 문자열입니다.</p> <p>이러한 매개 변수는 변수의 이름이거나 상수 문자열 값(예: "Encoded_String")입니다.</p>	<p>0: 실패.</p> <p>1: 성공.</p>

예제

다음 예제는 샘플 문자열을 인코딩된 형태로 변환하는 것입니다.

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

참고 사항

[Unescape](#)

FullScreen

지원 장치

Flash Lite 1.1.

설명

렌더링에 사용할 디스플레이 영역 크기를 설정합니다. 크기는 전체 화면 또는 전체 화면보다 작을 수 있습니다.

이 명령은 Flash Lite가 독립 실행형 모드에서 실행될 때만 지원됩니다. 브라우저의 플러그인처럼 플레이어가 다른 응용 프로그램과 관련되어 실행되는 경우에는 지원되지 않습니다.

명령	매개 변수	반환값
"FullScreen"	<i>size</i> true(전체 화면) 또는 false(전체 화면보다 작은 화면) 중 하나의 값을 가지는 정의된 변수 또는 상수 문자열 값이 크기가 될 수 있습니다. 그 밖의 모든 값은 false 값으로 처리됩니다.	-1: 지원되지 않음. 0: 지원됨.

예제

다음 예제는 디스플레이 영역을 전체 화면으로 설정하는 것입니다. 반환값이 0이 아닌 값인 경우, 재생 헤드를 `smallScreenMode`라는 이름의 프레임으로 보냅니다.

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

GetBatteryLevel

지원 장치

Flash Lite 1.1.

설명

현재 배터리 수준을 반환합니다. 이 값은 범위가 [0~GetMaxBatteryLevel 변수가 반환하는 최대값]에 해당하는 숫자 값입니다.

명령	매개 변수	반환값
"GetBatteryLevel"	없음	-1: 지원되지 않음. 기타 숫자 값: 현재 배터리 수준.

예제

다음 예제에서는 `battLevel` 변수를 현재 배터리 수준으로 설정합니다.

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

참고 사항

[GetMaxBatteryLevel](#)

GetDateDay

지원 장치

Flash Lite 1.1.

설명

현재 날짜의 일을 반환합니다. 숫자 값입니다(처음에 0이 없음). 유효한 일은 1~31입니다.

명령	매개 변수	반환값
"GetDateDay"	없음	-1: 지원되지 않음. 1~31: 매월의 날짜(일)입니다.

예제

다음 예제는 날짜 정보를 모아서 완전한 날짜 문자열을 만듭니다.

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

참고 사항

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

GetDateMonth

지원 장치

Flash Lite 1.1.

설명

현재 날짜의 달을 선행 0이 없는 숫자 값으로 반환합니다.

명령	매개 변수	반환값
"GetDateMonth"	없음	-1: 지원되지 않음. 1 - 12: 당월의 번호입니다.

예제

다음 예제는 날짜 정보를 모아서 완전한 날짜 문자열을 만듭니다.

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

참고 사항

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

GetDateWeekday

지원 장치

Flash Lite 1.1.

설명

숫자 값으로 나타나는 현재 날짜의 일 이름인 숫자 값을 반환합니다.

명령	매개 변수	반환값
"getDateWeekday"	없음	-1: 지원되지 않음. 0: 일요일. 1: 월요일. 2: 화요일. 3: 수요일. 4: 목요일. 5: 금요일. 6: 토요일.

예제

다음 예제는 날짜 정보를 모아서 완전한 날짜 문자열을 만듭니다.

```
today = fscommand2("getDateDay");
weekday = fscommand2("getDateWeekday");
thisMonth = fscommand2("getDateMonth");
thisYear = fscommand2("getDateYear");
when = weekday add ", " add thisMonth add " " add today add ", " add thisYear;
```

참고 사항

[getDateDay](#), [getDateMonth](#), [getDateYear](#)

getDateYear

지원 장치

Flash Lite 1.1.

설명

현재 날짜의 연도를 4자리 숫자 값으로 반환합니다.

명령	매개 변수	반환값
"getDateYear"	없음	-1: 지원되지 않음. 0 - 9999: 현재 년도입니다.

예제

다음 예제는 날짜 정보를 모아서 완전한 날짜 문자열을 만듭니다.

```
today = fscommand2("getDateDay");
weekday = fscommand2("getDateWeekday");
thisMonth = fscommand2("getDateMonth");
thisYear = fscommand2("getDateYear");
when = weekday add ", " add thisMonth add " " add today add ", " add thisYear;
```

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

GetDevice

지원 장치

Flash Lite 1.1.

설명

Flash Lite가 실행 중인 장치를 식별하는 매개 변수를 설정합니다. 일반적으로 이 식별자는 모델명입니다.

명령	매개 변수	반환값
"GetDevice"	<i>device</i> 장치의 식별자를 받는 문자열입니다. 이 값은 변수의 이름이거나 변수 이름을 포함하는 문자열 값이 될 수 있습니다.	-1: 지원되지 않음. 0: 지원됨.

예제다음은 장치 식별자를 `statusdevice` 변수로 지정한 다음 일반 장치 이름으로 텍스트 필드를 업데이트하는 코드 예제입니다.

다음은 일부 샘플 결과와 결과가 나타내는 장치들입니다.

D506i Mitsubishi 506i 전화.

DFOMA1 Mitsubishi FOMA1 전화.

F506i Fujitsu 506i 전화.

FFOMA1 Fujitsu FOMA1 전화.

N506i NEC 506i 전화.

NFOMA1 NEC FOMA1 전화.

Nokia3650 Nokia 3650 전화.

p506i Panasonic 506i 전화.

PFOMA1 Panasonic FOMA1 전화.

SH506i Sharp 506i 전화.

SHFOMA1 Sharp FOMA1 전화.

SO506i Sony 506i 전화.

```

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 "SO506i":
    /:myText += "device: Sony 506i" add newline;
    break;
}

```

GetDeviceID

지원 장치

Flash Lite 1.1.

설명

장치의 고유한 식별자(예: 일련 번호)를 나타내는 매개 변수를 설정합니다.

명령	매개 변수	반환값
"GetDeviceID"	<i>id</i> 장치의 고유한 식별자를 받는 문자열입니다. 이 값은 변수의 이름이거나 변수 이름을 포함하는 문자열 값이 될 수 있습니다.	-1: 지원되지 않음. 0: 지원됨.

예제

다음은 고유한 식별자를 `deviceID` 변수로 지정하는 예제입니다.

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

GetFreePlayerMemory

지원 장치
Flash Lite 1.1.

설명
현재 Flash Lite에서 사용 가능한 힙 메모리의 양(KB)을 반환합니다.

명령	매개 변수	반환값
"GetFreePlayerMemory"	없음	-1: 지원되지 않음. 0 또는 양수 값: 사용 가능한 힙 메모리입니다(KB).

예제
다음은 status가 빈 메모리 양과 같도록 설정하는 예제입니다.

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

참고 사항
[GetTotalPlayerMemory](#)

GetLanguage

지원 장치
Flash Lite 1.1.

설명
현재 장치가 사용하고 있는 언어를 식별하는 매개 변수를 설정합니다. 언어는 이름에 의해 전달되는 변수의 문자열로 반환됩니다.

명령	매개 변수	반환값
"GetLanguage"	<p><i>language</i> 언어 코드를 받는 문자열입니다. 이 값은 변수의 이름이거나 변수 이름을 포함하는 문자열 값이 될 수 있습니다. 반환되는 값은 다음 중 하나입니다.</p> <p>cs: 체코어</p> <p>da: 덴마크어</p> <p>de: 독일어</p> <p>en-UK: 영국 또는 국제 영어</p> <p>en-US: 미국 영어</p> <p>es: 스페인어</p> <p>fi: 핀란드어</p> <p>fr: 프랑스어</p> <p>hu: 헝가리아어</p> <p>it: 이탈리아어</p> <p>ja: 일본어</p> <p>ko: 한국어</p> <p>nl: 네덜란드어</p> <p>no: 노르웨이어</p> <p>pl: 폴란드어</p> <p>pt: 포르투갈어</p> <p>ru: 러시아어</p> <p>sv: 스웨덴어</p> <p>tr: 터키어</p> <p>xu: 확정되지 않은 언어</p> <p>zh-CN: 중국어 간체</p> <p>zh-TW: 중국어 번체</p>	<p>-1: 지원되지 않음.</p> <p>0: 지원됨.</p>

참고: 일본어 전화가 영어로 표시되도록 설정된 경우 en_US가 *language*에 대해 반환됩니다.

예제

다음은 언어 코드를 *language* 변수로 지정한 다음 Flash Lite 플레이어에서 인식하는 언어로 텍스트 필드를 업데이트하는 예제입니다.

```
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

지원 장치
Flash Lite 1.1.

설명

현재 정의된 지역에 따라 긴 형식으로 현재 날짜를 나타내는 문자열로 매개 변수를 설정합니다.

명령	매개 변수	반환값
"GetLocaleLongDate"	<p><i>longdate</i>"October 16, 2004" 또는 "16 October 2004"와 같이 현재 날짜 값의 긴 형식을 받는 문자열 변수입니다.</p> <p>이 값은 변수의 이름이거나 변수 이름을 포함하는 문자열 값이 될 수 있습니다.</p> <p><i>longdate</i>에 반환되는 값은 멀티문자로 가변 길이 문자열입니다. 실제 서식은 장치 및 지역에 따라 달라집니다.</p>	<p>-1: 지원되지 않음.</p> <p>0: 지원됨.</p>

예제

다음은 현재 날짜의 긴 형식을 *longDate* 변수에 반환하는 예제입니다. 또한 *status* 값을 설정하여 지정할 수 있었는지 보고합니다.

```

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

```

참고 사항

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

GetLocaleShortDate

지원 장치
Flash Lite 1.1.

설명

현재 정의된 지역에 따라 축약 형식으로 현재 날짜를 나타내는 문자열로 매개 변수를 설정합니다.

명령	매개 변수	반환값
"GetLocaleShortDate"	<p><i>shortdate</i> receive the long form of the value of the current date, such as "10/16/2004" 또는 "16-10-2004"와 같이 현재 날짜 값의 긴 형식을 받는 문자열 변수입니다.</p> <p>이 값은 변수의 이름이거나 변수 이름을 포함하는 문자열 값이 될 수 있습니다.</p> <p><i>shortdate</i>에 반환되는 값은 멀티문자로 가변 길이 문자열입니다. 실제 서식은 장치 및 지역에 따라 달라집니다.</p>	<p>-1: 지원되지 않음. 0: 지원됨.</p>

예제

다음 예제는 현재 날짜의 짧은 형식을 *shortDate* 변수로 지정합니다. 또한 *status* 값을 설정하여 지정할 수 있었는지 보고합니다.

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

참고 사항

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

GetLocaleTime

지원 장치

Flash Lite 1.1.

설명

현재 정의된 지역에 따라 현재 시간을 나타내는 문자열에 매개 변수를 설정합니다.

명령	매개 변수	반환값
"GetLocaleTime"	<p><i>time</i>"6:10:44 PM" 또는 "18:10:44"와 같은 현재 시간의 값을 받는 문자열 변수입니다.</p> <p>이 값은 변수의 이름이거나 변수 이름을 포함하는 문자열 값이 될 수 있습니다.</p> <p><i>time</i>에 반환되는 값은 멀티문자로 가변 길이 문자열입니다. 실제 서식은 장치 및 지역에 따라 달라집니다.</p>	<p>-1: 지원되지 않음. 0: 지원됨.</p>

예제

다음은 현재 지역 시간을 *time* 변수로 지정하는 예제입니다. 또한 *status* 값을 설정하여 지정할 수 있었는지 보고합니다.

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

참고 사항

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

GetMaxBatteryLevel

지원 장치
Flash Lite 1.1.

설명
장치의 최대 배터리 수준을 반환합니다. 이 값은 0보다 큰 숫자 값입니다.

명령	매개 변수	반환값
"GetMaxBatteryLevel"	없음	-1: 지원되지 않음. 기타 값: 최대 배터리 수준

예제
다음은 maxBatt 변수를 최대 배터리 수준으로 설정하는 예제입니다.

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

GetMaxSignalLevel

지원 장치
Flash Lite 1.1.

설명
최대 신호 강도 수준을 반환합니다. 이 값은 0보다 큰 숫자 값입니다.

명령	매개 변수	반환값
"GetMaxSignalLevel"	없음	-1: 지원되지 않음. 기타 숫자 값: 최대 신호 수준.

예제
다음 예제에서는 sigStrengthMax 변수에 최대 신호 강도를 할당합니다.

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

GetMaxVolumeLevel

지원 장치
Flash Lite 1.1.

설명
장치의 최대 볼륨 수준을 숫자 값으로 반환합니다.

명령	매개 변수	반환값
"GetMaxVolumeLevel"	없음	-1: 지원되지 않음. 기타 값: 최대 볼륨 수준.

예제

다음 예제에서는 maxvolume 변수를 장치의 최대 볼륨 수준으로 설정합니다.

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

참고 사항

[GetVolumeLevel](#)

GetNetworkConnectStatus

지원 장치

Flash Lite 1.1.

설명

현재 네트워크 연결 상태를 나타내는 값을 반환합니다.

명령	매개 변수	반환값
"GetNetworkConnectStatus"	없음	-1: 지원되지 않음. 0: 현재 활성 네트워크 연결이 있습니다. 1: 장치에서 네트워크 연결을 시도하고 있습니다. 2: 현재 활성 네트워크 연결이 없습니다. 3: 네트워크 연결이 보류되었습니다. 4: 네트워크 연결을 확인할 수 없습니다.

예제

다음 예제에서는 connectstatus 변수에 네트워크 연결 상태를 할당한 다음 switch 문을 사용하여 해당 연결 상태로 텍스트 필드를 업데이트합니다.

```
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

지원 장치
Flash Lite 1.1.

설명
매개 변수를 현재 네트워크의 이름으로 설정합니다.

명령	매개 변수	반환값
"GetNetworkName"	<p><i>networkName</i> 네트워크 이름을 나타내는 문자열입니다. 이 값은 변수의 이름이거나 변수 이름을 포함하는 문자열 값이 될 수 있습니다.</p> <p>네트워크가 등록되어 이름을 확인할 수 있으면 <i>networkname</i>이 네트워크 이름으로 설정되며 그렇지 않은 경우 빈 문자열로 설정됩니다.</p>	<p>-1: 지원되지 않음.</p> <p>0: 등록된 네트워크가 없습니다.</p> <p>1: 네트워크가 등록되었지만 네트워크 이름을 알 수 없습니다.</p> <p>2: 네트워크가 등록되었으며 네트워크 이름을 알 수 있습니다.</p>

예제
다음은 현재 네트워크의 이름을 myNetName 변수 및 netNameStatus 변수의 상태 값으로 지정하는 예제입니다.

```
netNameStatus = fscommand2("GetNetworkName", myNetName);
```

GetNetworkRequestStatus

지원 장치
Flash Lite 1.1.

설명

가장 최근의 HTTP 요청 상태를 나타내는 값을 반환합니다.

명령	매개 변수	반환값
"GetNetworkRequestStatus"	없음	-1: 지원되지 않는 명령입니다. 0: 대기 중인 요청이 있어 네트워크에 연결되었으며 서버의 호스트 이름이 확인되어 서버에 연결되었습니다. 1: 대기 중인 요청이 있고 네트워크에 연결하는 중입니다. 2: 대기 중인 요청이 있지만 네트워크에 아직 연결되지 않았습니다. 3: 대기 요청이 있어 네트워크에 연결되었으며 서버의 호스트 이름을 확인하는 중입니다. 4: 네트워크 오류로 요청이 실패했습니다. 5: 서버에 연결하지 못하여 요청이 실패했습니다. 6: 서버에서 HTTP 오류(예: 404)를 반환했습니다. 7: DNS 서버에 액세스할 수 없거나 서버 이름을 확인할 수 없어 요청이 실패했습니다. 8: 요청을 성공적으로 수행했습니다. 9: 시간 초과로 요청이 실패했습니다. 10: 아직 요청을 수행하지 않았습니다.

예제

다음 예제에서는 `requeststatus` 변수에 가장 최근의 HTTP 요청 상태를 할당한 다음 `switch` 문을 사용하여 해당 상태로 텍스트 필드를 업데이트합니다.

```
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

지원 장치

Flash Lite 1.1.

설명

전화의 네트워크 상태, 즉 등록된 네트워크가 있는지 그리고 전화가 현재 로밍 중인지 여부를 나타내는 값을 반환합니다.

명령	매개 변수	반환값
"GetNetworkStatus"	없음	-1: 지원되지 않는 명령입니다. 0: 등록된 네트워크가 없습니다. 1: 홈 네트워크에 연결되었습니다. 2: 확장된 홈 네트워크에 연결되었습니다. 3: 홈 네트워크 외부에서 로밍 중입니다.

예제

다음 예제에서는 `networkstatus` 변수에 네트워크 연결 상태를 할당한 다음 `switch` 문을 사용하여 해당 상태로 텍스트 필드를 업데이트합니다.

```
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

지원 장치

Flash Lite 1.1.

설명

장치의 클래스를 광범위하게 기술하는, 현재 플랫폼을 식별하는 매개 변수를 설정합니다. 공개 운영 체제가 설치된 장치의 경우 이 식별자는 일반적으로 운영 체제의 이름과 버전입니다.

명령	매개 변수	반환값
"GetPlatform"	<i>platform</i> 플랫폼의 식별자를 받는 문자열입니다. 이 값은 변수의 이름이거나 변수 이름을 포함하는 문자열 값이 될 수 있습니다.	-1: 지원되지 않음. 0: 지원됨.

예제

다음은 플랫폼 식별자를 `statusplatform` 변수로 지정한 다음 일반 장치 이름으로 텍스트 필드를 업데이트하는 예제입니다.

다음 예제는 `myPlatform`의 일부 샘플 결과 및 결과가 나타내는 장치의 클래스입니다.

506i 506i 전화.

FOMA1 A FOMA1 전화.

Symbian6.1_s60.1 Symbian 6.1, Series 60 버전 1 전화.

Symbian7.0 Symbian 7.0 전화

```
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

지원 장치

Flash Lite 1.1.

설명

배터리를 사용하는지 아니면 외부 전원을 사용하는지 여부를 나타내는 값을 반환합니다.

명령	매개 변수	반환값
"GetPowerSource"	없음	-1: 지원되지 않음. 0: 배터리 전원으로 장치가 작동 중입니다. 1: 외부 전원으로 장치가 작동 중입니다.

예제

다음은 myPower 변수를 설정하여 전원을 표시하거나 그렇지 않은 경우 -1로 설정하는 예제입니다.

```
myPower = fscommand2("GetPowerSource");
```

GetSignalLevel

지원 장치

Flash Lite 1.1.

설명

현재 신호 강도를 숫자 값으로 반환합니다.

명령	매개 변수	반환값
"GetSignalLevel"	없음	-1: 지원되지 않음. 기타 숫자 값: 범위가 0부터 GetMaxSignalLevel에서 반환된 최대 값까지인 현재 신호 수준입니다.

예제

다음 예제에서는 sigLevel 변수에 신호 수준 값을 할당합니다.

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

GetTimeHours

지원 장치

Flash Lite 1.1.

설명

24시간제를 기준으로, 현재 시간 중 시를 반환합니다. 숫자 값입니다(처음에 0이 없음).

명령	매개 변수	반환값
"GetTimeHours"	없음	-1: 지원되지 않음. 0 - 23: 현재 시간입니다.

예제

다음은 hour 변수를 현재 날짜의 시 부분 또는 -1로 설정하는 예제입니다.

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

참고 사항

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

GetTimeMinutes

지원 장치

Flash Lite 1.1.

설명

현재 날짜 중 분을 반환합니다. 숫자 값입니다(처음에 0이 없음).

명령	매개 변수	반환값
"GetTimeMinutes"	없음	-1: 지원되지 않음. 0 - 59: 현재 시간 중 분입니다.

예제

다음은 minutes 변수를 현재 시간의 분 부분 또는 -1로 설정하는 예제입니다.

```
minutes = fscCommand2("GetTimeMinutes");
trace (minutes); // output: 38
```

참고 사항

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

GetTimeSeconds

지원 장치

Flash Lite 1.1.

설명

일의 현재 시간 중 초를 반환합니다. 숫자 값입니다(처음에 0이 없음).

명령	매개 변수	반환값
"GetTimeSeconds"	없음	-1: 지원되지 않음. 0 - 59: 현재 시간 중 초입니다.

예제

다음은 seconds 변수를 현재 시간의 초 부분 또는 -1로 설정하는 예제입니다.

```
seconds = fscCommand2("GetTimeSeconds");
trace (seconds); // output: 41
```

참고 사항

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

GetTimeZoneOffset

지원 장치

Flash Lite 1.1.

설명

지역 시간대와 표준시(UTC) 사이의 분 값으로 매개 변수를 설정합니다.

명령	매개 변수	반환값
"GetTimeZoneOffset"	<i>timezoneOffset</i> 로컬 시간대와 UTC의 분 단위 차이를 나타내는 숫자입니다. 이 값은 변수의 이름이거나 변수 이름을 포함하는 문자열 값이 될 수 있습니다. 다음과 같은 음수 또는 양수 숫자 값이 반환됩니다. 540: 일본 표준시 -420: 태평양 일광 절약 시간제	-1: 지원되지 않음. 0: 지원됨.

예제

다음은 UTC의 오프셋 분을 `timezoneoffset` 변수에 설정하고 `status`를 0으로 설정하거나 `status`를 -1로 설정하는 예제입니다.

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

참고 사항

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

GetTotalPlayerMemory

지원 장치

Flash Lite 1.1.

설명

Flash Lite에 할당된 총 힙 메모리 크기(KB)를 반환합니다.

명령	매개 변수	반환값
"GetTotalPlayerMemory"	없음	-1: 지원되지 않음. 0 또는 양수 값: 힙 메모리의 전체 양(KB 단위)입니다.

예제

다음 예제에서는 `status` 변수를 총 누적 메모리 크기로 설정합니다.

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

참고 사항

[GetFreePlayerMemory](#)

GetVolumeLevel

지원 장치

Flash Lite 1.1.

설명

장치의 현재 볼륨 수준을 숫자 값으로 반환합니다.

명령	매개 변수	반환값
"GetVolumeLevel"	없음	-1: 지원되지 않음. 기타 숫자 값: 범위가 0부터 <code>fscommand2("GetMaxVolumeLevel")</code> 에서 반환한 값까지인 현재 볼륨 수준입니다.

예제

다음은 현재 볼륨 수준을 `volume` 변수로 지정하는 예제입니다.

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

Quit

지원 장치

Flash Lite 1.1.

설명

Flash Lite 플레이어가 재생을 중지하고 종료하도록 합니다.

이 명령은 Flash Lite가 독립 실행형 모드에서 실행될 때만 지원됩니다. 브라우저의 플러그인처럼 플레이어가 다른 응용 프로그램과 관련되어 실행되는 경우에는 지원되지 않습니다.

명령	매개 변수	반환값
"Quit"	없음	-1: 지원되지 않음.

예제

다음 예제에서는 Flash Lite가 독립 실행형 모드에서 실행될 때 재생을 중지하고 종료하도록 합니다.

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

ResetSoftKeys

지원 장치

Flash Lite 1.1.

설명

소프트 키를 원래 설정으로 복구합니다.

이 명령은 Flash Lite가 독립 실행형 모드에서 실행될 때만 지원됩니다. 브라우저의 플러그인처럼 플레이어가 다른 응용 프로그램과 관련되어 실행되는 경우에는 지원되지 않습니다.

명령	매개 변수	반환값
"ResetSoftKeys"	없음	-1: 지원되지 않음. 0: 지원됨.

예제

다음 명령문은 소프트 키를 원래 설정으로 복구합니다.

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

참고 사항

[SetSoftKeys](#)

SetInputTextType

지원 장치
Flash Lite 1.1.

설명
입력 텍스트 필드가 열리는 모드를 지정합니다:

명령	매개 변수	반환값
"SetInputTextType"	<i>variableName</i> 입력 텍스트 필드 이름입니다. 이 값은 변수의 이름이거나 변수 이름을 포함하는 문자열 값이 될 수 있습니다. <i>typeNumeric</i> , Alpha, Alphanumeric, Latin, NonLatin 또는 NoRestriction 값 중 하나입니다.	0: 실패. 1: 성공.

Flash Lite는 호스트 응용 프로그램이 FEP(Front-End Processor)라고도 하는 일반 장치 전용 텍스트 입력 인터페이스를 시작하도록 요청하여 입력 텍스트 기능을 지원합니다. SetInputTextType 명령이 사용되지 않을 때 FEP는 기본 모드에서 열립니다. 다음 표에서는 각 모드의 효과 및 대체 모드에 대해 설명합니다.

지정된 모드	FEP에 설정되는 모드(다른 모드와 함께 사용할 수 없음)	현재 장치에서 지원되지 않는 경우 FEP를 여는 대체 모드
Numeric	숫자 전용(0~9)	Alphanumeric
Alpha	영문자 전용(A~Z, a~z)	Alphanumeric
Alphanumeric	영숫자 전용(0~9, A~Z, a~z)	Latin
Latin	라틴 문자 전용(영숫자 및 문장 부호)	NoRestriction
NonLatin	라틴어 이외의 문자 전용(예: 일본어 간지와 가나)	NoRestriction
NoRestriction	기본 모드(FEP에 제한 없음)	

참고: 이러한 입력 텍스트 필드 유형을 지원하지 않는 휴대 전화도 있습니다. 따라서 입력 텍스트 데이터의 유효성을 검사해야 합니다.

예제
다음 코드 행은 input1 변수와 관련된 필드에 숫자 데이터를 입력할 수 있도록 입력 텍스트 유형을 설정합니다.

```
status = fscommand2("SetInputTextType", "input1", "Numeric");
```

SetQuality

지원 장치
Flash Lite 1.1.

설명
애니메이션 렌더링의 품질을 설정합니다.

명령	매개 변수	반환값
"SetQuality"	<i>quality</i> 렌더링 품질은 "high", "medium" 또는 "low" 중 하나로 설정해야 합니다.	-1: 지원되지 않음. 0: 지원됨.

예제

다음은 렌더링 품질을 LOW로 설정하는 예제입니다.

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

SetSoftKeys

지원 장치

Flash Lite 1.1.

설명

장치의 왼쪽 및 오른쪽 소프트 키가 액세스 및 재매핑이 가능한 경우에 이것을 다시 매핑합니다.

이 명령이 실행된 후 왼쪽 키를 누르면 PageUp 키 누르기 이벤트가 생성되며 오른쪽 키를 누르면 PageDown 키 누르기 이벤트가 생성됩니다. PageUp 및 PageDown 키 누르기 이벤트와 관련된 ActionScript는 해당 키를 누르면 실행됩니다.

이 명령은 Flash Lite가 독립 실행형 모드에서 실행될 때만 지원됩니다. 브라우저의 플러그인처럼 플레이어가 다른 응용 프로그램 램과 관련되어 실행되는 경우에는 지원되지 않습니다.

명령	매개 변수	반환값
"SetSoftKeys"	<i>left</i> 왼쪽 소프트 키에 표시되는 텍스트입니다. <i>right</i> 오른쪽 소프트 키에 표시되는 텍스트입니다. 이러한 매개 변수는 변수의 이름 또는 상수 문자열 값(예: "Previous")입니다.	-1: 지원되지 않음. 0: 지원됨.

예제

다음은 왼쪽 소프트 키 레이블로 Previous를 지정하고 오른쪽 소프트 키는 Next로 지정하는 예제입니다.

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

참고 사항

[ResetSoftKeys](#)

StartVibrate

지원 장치

Flash Lite 1.1.

설명

전화의 진동 기능을 시작합니다. 진동 기능이 이미 실행 중인 경우에는 새로 시작하기 전에 기존의 진동 기능을 중지합니다. 또한 Flash 응용 프로그램 재생이 중단 또는 일시 정지된 경우나 Flash Lite 플레이어가 종료되는 경우에도 진동이 중단됩니다.

명령	매개 변수	반환값
"StartVibrate"	<p><i>time_on</i> 진동이 실행되는 시간(밀리초)을 나타내며 최대값은 5초입니다.</p> <p><i>time_off</i> 진동 해제 시간(밀리초)이며 최대값은 5초입니다.</p> <p><i>repeat</i> 이 진동을 반복하는 횟수(최대 3회)입니다.</p>	<p>-1: 지원되지 않음.</p> <p>0: 진동이 시작되었습니다.</p> <p>1: 오류가 발생하여 진동을 시작하지 못했습니다.</p>

예제

다음 예제에서는 2.5초 동안 실행한 후 1초 동안 끄는 과정을 두 번 반복하는 진동 시퀀스를 시작합니다. 이때 성공 또는 실패를 나타내는 값을 `status` 변수에 할당합니다.

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

참고 사항

[StopVibrate](#)

StopVibrate

지원 장치

Flash Lite 1.1.

설명

현재 실행 중인 진동을 중지합니다.

명령	매개 변수	반환값
"StopVibrate"	없음	<p>-1: 지원되지 않음.</p> <p>0: 진동이 중지됨.</p>

예제

다음 예제에서는 `StopVibrate` 를 호출하고 결과(지원되지 않음 또는 진동이 중지됨)를 `status` 변수에 저장합니다.

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

참고 사항

[StartVibrate](#)

Unescape

지원 장치

Flash Lite 1.1.

설명

네트워크 전송에 안전하도록 인코딩된 임의의 문자열을 인코딩되지 않은 일반적인 형태로 디코딩합니다. 두 개의 16진수 숫자 뒤에 % 문자가 오는 16진수 포맷의 모든 문자는 디코딩된 포맷으로 변환됩니다.

명령	매개 변수	반환값
"Unescape"	<p><i>original</i> URL에 안전한 형식에서 일반적인 포맷으로 디코딩할 문자열입니다.</p> <p><i>decoded</i> 디코딩된 문자열의 결과입니다.</p> <p>이 매개 변수는 변수 이름 또는 변수 이름을 포함하는 문자열 값이 될 수 있습니다.</p>	<p>0: 실패.</p> <p>1: 성공.</p>

예제

다음 예제는 인코딩된 문자열의 디코딩 방법을 보여 줍니다.

```

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

```

참고 사항

[Escape](#)

색인

기호

_alpha 변수 31
 _cap4WayKeyAS 변수 82
 _capCompoundSound variable 77
 _capEmail 변수 77
 _capLoadData 변수 78
 _capMFi 변수 78
 _capMIDI 변수 79
 _capMMS 변수 79
 _capSMAF 변수 80
 _capSMSx0d x0d 변수 81
 _capStreamSoundx0d x0d 변수 81
 _currentframe 속성 32
 _focusrect 속성 32
 _framesloaded 속성 33
 _height 속성 33
 _highquality 속성 34
 _level 속성 34
 _name 속성 35
 _rotation 속성 36
 _scroll 속성 36
 _target 속성 37
 _visible 속성 37
 _width 속성 38
 _x 속성 38
 _xscale 속성 39
 _y 속성 39
 _yscale 속성 40
 ,(첨표) 연산자 54
 !(논리 NOT) 연산자를 사용합니다 60
 ? (조건부) 연산자 56
 . (도트) 연산자 58
 " " (문자열 구분 기호) 연산자 68
 * (곱하기) 연산자 63
 *=(선행 곱하기 대입) 연산자 62
 / (슬래시 - 루트 타임라인) 속성 31
 /(나누기) 연산자 57
 /* (블록 주석) 연산자 54
 //(주석) 연산자 55

/=(나누기) 연산자 57
 \ 66, 67
 \ (숫자 비항등) 연산자 66
 || (논리 OR) 연산자 61
 &&(논리 AND) 연산자 59
 % (모듈러스) 연산자 61
 %=(선행 모듈러스 대입) 연산자 62
 +(숫자 더하기) 연산자 64
 ++ (증가) 연산자 59
 += (선행 증가 대입) 연산자 52
 =(대입) 연산자 53
 -= (선행 감소 대입) 연산자 74
 == (숫자 항등) 연산자 64
 > (보다 크거나 같음) 연산자 65
 > (보다 큼) 연산자 65
 \$version 변수 83

A

add(문자열 결합) 연산자 52
 _alpha variable 31
 AND 연산자 59
 and 연산자 53

B

break 문 41

C

_cap4WayKeyAS variable 82
 _capCompoundSound variable 77
 _capEmail variable 77
 _capLoadData variable 78
 _capMFi variable 79
 _capMMS variable 79
 _capSMAF variable 80
 _capSMS variable 81
 _capStreamSoundx0d x0d variable 81
 case 문 42
 chr() 함수 4

continue 문 43
 _currentframe property 32

D

do..while 문 44
 duplicateMovieClip() 함수 4

E

else if 명령문 45
 else 문 45
 eq(문자열 항등) 연산자 69
 eval() 함수 5

F

_focusrect property 32
 for 루프 46
 for 문 46
 _framesloaded property 33
 fscommand() 명령 83
 functions
 chr() 4
 duplicateMovieClip() 4
 eval() 5
 getProperty() 6
 getTimer() 6
 getURL() 7
 gotoAndPlay() 9
 gotoAndStop() 9
 ifFrameLoaded() 10
 int() 11
 length() 11
 loadMovie() 12
 loadMovieNum() 13
 loadVariables() 13
 loadVariablesNum() 14
 mbchr() 15
 mbsubstring() 17
 nextFrame() 17
 nextScene() 18

- Number() 18
 - on() 19
 - ord() 20
 - play() 20
 - prevFrame() 21
 - prevScene() 21
 - random() 22
 - removeMovieClip() 22
 - set() 23
 - setProperty() 23
 - stop() 24
 - stopAllSounds() 24
 - String() 25
 - substring() 26
 - tellTarget() 26
 - toggleHighQuality() 27
 - trace() 27
 - unloadMovie() 28
 - unloadMovieNum() 28
- G**
- ge(보다 크거나 같음 문자열) 연산자 70
 - getProperty() 함수 6
 - getTimer() 함수 6
 - getURL() 함수 7
 - gotoAndPlay() 함수 9
 - gotoAndStop() 함수 9
 - gt(보다 큼 문자열) 연산자 69
- H**
- _height property 33
 - _highquality property 34
- I**
- if 문 47
 - ifFrameLoaded() 함수 10
 - int() 함수 11
- L**
- le(보다 작거나 같음 문자열) 연산자 72
 - length() 함수 11
 - _level property 34
 - loadMovie() 함수 12
 - loadMovieNum() 함수 13
 - loadVariables() 함수 13
 - loadVariablesNum() 함수 14
 - lt(보다 작음 문자열) 연산자 71
- M**
- maxscroll 속성 35
 - mbchr() 함수 15
 - mbsubstring() 함수 17
 - MFI 사운드 78
 - MIDI 사운드 79
 - MMS 메시징 79
- N**
- _name property 35
 - ne(문자열 비항등) 연산자 71
 - nextFrame() 함수 17
 - nextScene() 함수 18
 - NOT 연산자 60
 - Number() 함수 18
- O**
- on() 함수 19
 - OR 연산자 61
 - ord() 함수 20
- P**
- play() 함수 20
 - prevFrame() 함수 21
 - prevScene() 함수 21
- R**
- random() 함수 22
 - removeMovieClip() 함수 22
 - _rotation property 36
- S**
- scroll 속성 36
 - set() 함수 23
 - setProperty() 함수 23
 - stop() 함수 24
 - stopAllSounds() 함수 24
 - String() 함수 25
 - substring() 함수 26
 - switch 문 47
- T**
- _target property 37
 - tellTarget() 함수 26
 - toggleHighQuality() 함수 27
 - _totalframes property 37
 - trace() 함수 27
- U**
- unloadMovie() 함수 28
 - unloadMovieNum() 함수 28
- V**
- variables
 - _capCompoundSound 77
 - variables, sound
 - _capCompoundSound 77
 - _visible property 37
- W**
- while 루프 44
 - while 문 48
 - _widthx11 property 38
- X**
- _x property 38
 - xd0 (빼기) 연산자 73
 - xd0 xd0 (감소) 연산자 56
 - _xscale property 39
- Y**
- _y property 39
 - _yscale property 40
- ㄱ**
- 결합 52
 - 곱하기 63

L

나누기 57
나누기 대입 연산자 57
논리 AND 연산자 59
논리 NOT 연산자 60
논리 OR 연산자 61

ㄷ

대입 연산자 53
더하기 대입 연산자 52
도트 연산자 58

ㄹ

루트 타임라인 식별자 31

ㅁ

메시징 변수 79, 81

명령문

break 41
case 42
continue 43
do..while 44
else 45
else if 45
for 46
if 47
switch 47
while 48
논리 NOT 60

명령어

블록 54
모듈러스 연산자 61
문자열 결합 52
문자열 구분 기호 연산자 68
문자열 향등 연산자 69

ㅂ

변수

_alpha 31
_cap4WayKeyAS 82
_capEmail 77
_capLoadData 78
_capMFi 78
_capMIDI 79

_capMMS 79
_capSMAF 80
_capSMS 81
_capStreamSound 81
\$version 83
Flash Lite의 버전 번호 83
데이터를 로드하는 기능 78

전자 메일 기능 77
화살표 키 탐색 82

변수, 메시징

_capMMS 79
_capSMS 81
변수, 사운드
_capMFi 78
_capMIDI 79
_capSMAF 80
_capStreamSound 81

보다 작거나 같음 문자열 72
보다 작거나 같음 연산자 67
보다 작음 연산자 66
보다 크거나 같음 문자열 70
보다 크거나 같음 연산자 65
보다 큼 문자열 연산자 69
보다 큼 연산자 65
블록 주석 연산자 54
비향등 연산자 66
빼기 대입 연산자 74

ㅅ

사운드 변수 77, 78, 79, 80, 81

선행 모듈러스 대입 62

속성 31

_alpha 31
_currentframe 32
_focusrect 32
_framesloaded 33
_height 33
_highquality 34
_level 34
_name 35
_rotation 36
_scroll 36
_target 37
_visible 37

_width 38
_x 38
_xscale 39
_y 39
_yscale 40
maxscroll 35
scroll 36

숫자 더하기 64

점표 연산자 54

슬래시 31

ㅇ

연산자 52

곱하기 63

나누기 57

나누기 대입 57

논리 AND 59

논리 NOT 60

논리 OR 61

대입 53

더하기 대입 52

도트 58

모듈러스 61

모듈러스 대입 62

문자열 기호 68

문자열 비향등 71

문자열 향등 69

및 53

보다 작거나 같음 문자열 72

보다 작거나 같음 숫자 67

보다 작음 문자열 71

보다 작음 숫자 66

보다 크거나 같음 65

보다 크거나 같음 문자열 70

보다 큼 65

보다 큼 문자열 69

블록 연산자 54

빼기 대입 74

숫자 더하기 64

숫자 비향등 66

숫자 향등 64

점표 54

조건부 56

주석 55
증가 연산자 59

ㅈ

전자 메일 기능 변수 77
조건부 연산자 56
조건에 사용할 수 있는 두 가지 내장 변수를 포
함합니다 47
주석
1행 55
증가 연산자 59

ㅎ

함수
fscommand() 83
호출입니다 3