

# FLASH<sup>®</sup> LITE<sup>™</sup> 1.x ACTIONSCRIPT<sup>™</sup> 학습

© 2007 Adobe Systems Incorporated. All rights reserved.

## Flash® Lite™ 1.x ActionScript™ 학습

본 안내서가 최종 사용자 사용권 계약서가 포함된 소프트웨어와 함께 배포되는 경우, 본 안내서 및 설명된 소프트웨어는 사용권 하에서 제공되며 해당 사용권 약관에 따라서만 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 안됩니다. Adobe Systems Incorporated는 이 안내서에 있을 수 있는 정보의 오류나 부정확성에 대해 어떠한 책임이나 의무도 없습니다.

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 조직을 의미하지는 않습니다.

Adobe, Adobe 로고, Flash Lite 및 Flash는 미국 및/또는 기타 국가에서 Adobe Systems Incorporated의 등록 상표 또는 상표입니다.

## 타사 정보

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 액세스할 때 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.



Sorenson™ Spark™ 비디오 압축 및 압축 해제 기술은 Sorenson Media, Inc.에서 사용권을 허가 받은 기술입니다.

Fraunhofer-IIS/Thomson Multimedia: MPEG Layer-3 오디오 압축 기술은 Fraunhofer IIS 및 Thomson Multimedia(<http://www.iis.fhg.de/amm/>)에서 사용권을 부여 받은 기술입니다.

Independent JPEG Group: 이 소프트웨어는 Independent JPEG Group 작업의 일부에 기반하고 있습니다.

Nellymoser, Inc.: 음성 압축 및 압축 해제 기술은 Nellymoser, Inc. (<http://www.nellymoser.com>)에서 사용권을 부여 받은 기술입니다.

Opera® 브라우저 Copyright © 1995-2002 Opera Software ASA 및 공급자. All rights reserved.

Macromedia Flash 8 비디오는 On2 TrueMotion에서 제공하는 비디오 기술을 사용합니다. © 1992-2005 On2 Technologies, Inc. All Rights Reserved. <http://www.on2.com>.

Visual SourceSafe는 미국 및/또는 기타 국가에서 Microsoft Corporation의 등록 상표 또는 상표입니다.

업데이트 정보/추가된 타사 코드 정보는 [http://www.adobe.com/go/thirdparty\\_kr/](http://www.adobe.com/go/thirdparty_kr/)를 참조하십시오.

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

미국 정부 최종 사용자 고지 사항. 이 소프트웨어와 설명서는 “상용 컴퓨터 소프트웨어” 및 “상용 컴퓨터 소프트웨어 설명서”(미국 연방 규정 48편 12장 212절(48 C.F.R. §2.212) 또는 227장 7202절(48 C.F.R. §27.7202)에 사용된 용어)로 이루어진 “상용 품목”(미국 연방 규정 48편 2장 101절(48 C.F.R. §.101)에 정의된 용어)입니다. 48 C.F.R. §12.212 또는 48 C.F.R. §§227.7202-1에서 227.7202-4까지의 해당 조항에 따라 상용 컴퓨터 소프트웨어 및 상용 컴퓨터 소프트웨어 설명서는 미국 정부 최종 사용자에게 (a) 상용 품목으로서만 사용권이 허가되고 (b) 관련 조건 및 조항에 따라 다른 모든 최종 사용자에게 허용되는 정도로만 사용권이 부여됩니다. 기타 공개되지 않은 권리도 미국 저작권법에 의해 보호됩니다. Adobe Systems Incorporated, 345 Park Avenue, San Jose, CA 95110-2704, USA. 미국 정부 최종 사용자에게 Adobe는 해당하는 경우 미 대통령 시행령 11246의 수정안, Vietnam Era Veterans Readjustment Assistance Act(1974)의 402호 (38 USC 4212), 장애인보호법(1973) 503호 수정안, 41 CFR 60-1부에서 60-60, 60-250부 및 60-741부의 규정 등을 준수하는 데 동의합니다. 상기 조항에 포함된 차별 철폐 조치 조항 및 규정이 관련 조항으로 포함됩니다.



# 목 차

<b>제1장: Flash Lite 1.x ActionScript</b> .....	<b>7</b>
Flash Lite 1.x ActionScript 개요 .....	7
Flash Lite 1.0과 Flash Lite 1.1 ActionScript 차이점 .....	8
Flash Lite 1.x ActionScript에서 지원되지 않는 Flash 4 ActionScript .....	8
Flash Lite 1.x ActionScript에서 사용할 수 없는 기능 .....	9
<b>제2장: Flash 4 ActionScript 입문서</b> .....	<b>11</b>
무비 클립 속성 가져오기 및 설정 .....	11
다른 타임라인 제어 .....	12
변수 사용 .....	13
배열 애니메이션 .....	13
텍스트 및 문자열 작업 .....	14
call() 함수를 사용하여 함수 만들기 .....	15
eval() 함수 사용 .....	19
<b>제3장: 일반적인 스크립팅 작업</b> .....	<b>21</b>
장치 및 플랫폼 기능 확인 .....	21
웹 페이지 열기 .....	22
전화 걸기 .....	22
텍스트 또는 멀티미디어 메시지 전송 .....	23
전자 메일 메시지 전송 .....	24
외부 SWF 파일 로드 .....	24
외부 데이터 로드 .....	24
<b>색 인</b> .....	<b>27</b>



ActionScript를 사용하여 Adobe® Flash® Lite™ 응용 프로그램에 프로그래밍 논리 및 상호 작용 기능을 추가할 수 있습니다. Flash Lite 1.x ActionScript(Adobe의 Macromedia® Flash® Lite™ 1.0 및 Macromedia® Flash® Lite™ 1.1 소프트웨어 ActionScript 버전)에는 다양한 Adobe의 Macromedia® Flash® 4 ActionScript, 추가 명령 및 Flash Lite 플레이어의 특정 속성(예: 전화 걸기, 문자 메시지 기능 또는 시간 및 날짜 알림 기능)이 포함되어 있습니다.

이 장에서 설명하는 항목은 다음과 같습니다.

Flash Lite 1.x ActionScript 개요 .....	7
Flash Lite 1.0 과 Flash Lite 1.1 ActionScript 차이점 .....	8
Flash Lite 1.x ActionScript 에서 지원되지 않는 Flash 4 ActionScript .....	8
Flash Lite 1.x ActionScript 에서 사용할 수 없는 기능 .....	9

## Flash Lite 1.x ActionScript 개요

Flash Lite 1.x ActionScript는 다음과 같은 내용으로 구성되어 있습니다.

**Flash Player 4 ActionScript 기능** 연산자(예: 비교 및 대입 연산자), 무비 클립 속성(예: `_height`, `_x` 및 `_y`), 타임라인 컨트롤 함수(예: `gotoAndPlay()` 또는 `stop()`)뿐만 아니라 `loadVariables()` 및 `loadMovie()` 함수(Flash Lite 1.1에만 해당)와 같은 네트워크 함수가 포함되어 있습니다. 지원되지 않는 Flash 4 ActionScript 기능 목록은 8페이지의 “Flash Lite 1.x ActionScript에서 지원되지 않는 Flash 4 ActionScript”를 참조하십시오.

**전화 통합 명령 및 속성** 예를 들어, Flash Lite에서는 장치에서 날짜 및 시간 정보 쿼리, 전화 걸기, SMS(Short Message Service) 텍스트 메시지 보내기, 장치에 설치된 외부 응용 프로그램 시작 등을 수행할 수 있는 다양한 명령을 제공합니다.

**플랫폼 기능 변수(Flash Lite 1.1에만 해당)** 여기서 제공되는 속성들은 장치 또는 Flash Lite 런타임 환경의 기능에 대한 정보를 제공합니다. 예를 들어, `_capLoadData` 변수는 사용자의 응용 프로그램에서 네트워크를 통해 데이터를 로드할 수 있는지 여부를 나타냅니다.

**fscommand2()** 함수 fscommand() 함수처럼 fscommand2()를 사용하여 호스트 환경 또는 시스템(여기서는 휴대폰 또는 장치)과 통신할 수 있습니다. fscommand2() 함수는 fscommand() 함수에 비해 향상된 기능을 갖고 있으며, 임의의 개수의 인수를 전달하고 즉시 반환값을 가져오는(fscommand()에서는 다음 프레임까지 대기해야 함) 기능 등 새로운 기능을 갖고 있습니다.

## Flash Lite 1.0과 Flash Lite 1.1 ActionScript 차이점

아래에 나열된 Flash Lite 1.1 ActionScript 기능은 Flash Lite 1.0에서 사용할 수 없습니다.

- 네트워크 액세스 기능 또는 네트워크 상태 정보를 확인하는 기능. 예를 들어, Flash Lite 1.0에서는 loadVariables() 또는 loadMovie() 함수를 사용하여 외부 데이터 또는 SWF 파일을 로드할 수 없으며, 장치의 연결 신호 세기 또는 네트워크 요청 상태를 확인하는 다양한 fscommand2() 명령을 사용할 수 없습니다.
- 장치에서 시간과 날짜 정보 가져오는 기능.
- Flash Lite 플랫폼 및 장치의 기능에 대한 정보를 제공하는 플랫폼 기능 변수.
- fscommand2() 함수 및 SetSoftKeys와 FullScreen과 같은 관련 명령.
- scroll 및 maxscroll 텍스트 필드 속성.

## Flash Lite 1.x ActionScript에서 지원되지 않는 Flash 4 ActionScript

아래에 나열된 Flash 4 ActionScript 기능은 Flash Lite 1.x ActionScript에서 지원되지 않거나 일부만 지원되거나 기능입니다.

- startDrag() 및 stopDrag() 함수.
- Flash Lite 1.x ActionScript는 Flash Player 4에서 지원되는 버튼 이벤트 중 일부만 지원합니다. 버튼 이벤트 처리에 대한 자세한 내용은 Flash Lite 응용 프로그램 개발의 제1장, “상호 작용 및 탐색 기능 만들기”를 참조하십시오.
- Flash Lite 1.x ActionScript는 Flash Player 4에서 지원되는 키 이벤트 중 일부만 지원합니다. Flash Lite에서 지원되는 키 이벤트에 대한 자세한 내용은 Flash Lite 응용 프로그램 개발의 제1장, “상호 작용 및 탐색 기능 만들기”를 참조하십시오.
- \_dropTarget 속성.
- \_soundBufTime 속성.
- \_url 속성.
- String() 변환 함수.

# Flash Lite 1.x ActionScript에서 사용할 수 없는 기능

Flash Lite 플레이어는 이전 버전의 Flash Player를 기반으로 제작된 것이기 때문에 최신 버전의 Flash Player 또는 사용자에게 친숙한 다른 프로그래밍 언어에서 사용할 수 있는 프로그래밍 기능을 모두 지원하지는 않습니다. 이 단원에서는 Flash Lite 1.x ActionScript에서 사용할 수 없는 프로그래밍 기능에 대해 살펴보고 이러한 기능을 대체할 수단이나 차선택이 있는지에 대해 설명합니다.

**사용자 정의 함수** Flash Lite 1.x에서는 사용자 정의 함수를 정의하고 호출하는 기능을 지원하지 않지만 `call()` 함수를 사용하여 타임라인에서 임의의 프레임에 있는 코드를 실행할 수 있습니다. 자세한 내용은 [15페이지의 “call\(\) 함수를 사용하여 함수 만들기”](#)를 참조하십시오.

**기본 배열, 객체 또는 기타 복합 데이터 유형** Flash Lite 1.x에서는 기본 배열 데이터 구조나 기타 복합 데이터 유형을 지원하지 않습니다. 그러나 의사 배열(`eval()` 함수를 사용하여 연결된 문자열을 동적으로 평가하는 방법)을 사용하여 배열을 에뮬레이션할 수 있습니다. 자세한 내용은 [13페이지의 “배열 에뮬레이션”](#)을 참조하십시오.

**외부 이미지 또는 사운드 파일의 런타임 로드** Flash Player의 데스크톱 버전과 달리 Flash Lite 1.x ActionScript에서는 외부 JPEG 파일 또는 MP3 파일을 로드할 수 없습니다. Flash Lite 1.1에서는 `loadMovie()` 함수를 사용하여 외부 SWF 파일을 로드할 수 있습니다. 자세한 내용은 [24페이지의 “외부 SWF 파일 로드”](#)를 참조하십시오.



Adobe의 Macromedia Flash Lite 1.x ActionScript는 Adobe의 Macromedia Flash Player 4에서 처음 도입된 ActionScript 버전을 기반으로 제작되었습니다. 따라서 이후 Flash Player 버전(데스크톱 시스템용)에 추가된 프로그래밍 기능 중 일부 기능은 Flash Lite 1.x 응용 프로그램에서 사용할 수 없습니다.

Flash 4 ActionScript 구문 및 기능에 익숙하지 않거나 Flash 개발 작업을 한 지 오래되어 있어 버린 내용이 많다면 이 장에서 Flash Lite 응용 프로그램 개발에 필요한 Flash 4 ActionScript 사용 방법에 대한 기초적인 내용을 살펴보십시오.

이 장에서 설명하는 항목은 다음과 같습니다.

무비 클립 속성 가져오기 및 설정 .....	11
다른 타임라인 제어 .....	12
변수 사용 .....	13
배열 에뮬레이션 .....	13
텍스트 및 문자열 작업 .....	14
call() 함수를 사용하여 함수 만들기 .....	15
eval() 함수 사용 .....	19

## 무비 클립 속성 가져오기 및 설정

무비 클립 속성을 가져오거나 설정하려면(설정 가능한 경우) 도트 구문 또는 `setProperty()`나 `getProperty()` 함수를 사용합니다. `tellTarget()` 함수를 사용할 수도 있습니다.

도트 구문을 사용하려면 무비 클립 인스턴스 이름을 지정하고, 이어서 도트(.)를 입력한 다음 해당 속성 이름을 지정합니다. 예를 들어, 다음 코드는 `cartoonArea`라는 무비 클립의 `x` 스크린 좌표(`_x` 무비 클립 속성으로 표시)를 가져와서 그 결과를 `x_pos`라는 변수에 대입합니다.

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

다음 예제는 앞의 예제와 동일한 내용이지만 `getProperty()` 함수를 사용하여 무비 클립의 `x` 좌표를 가져옵니다.

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

`setProperty()` 함수는 다음 예제와 같이 무비 클립 인스턴스의 속성을 설정합니다.

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

다음 예제는 앞의 예제와 동일한 내용이지만 도트 구문을 사용합니다.

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

`tellTarget()` 문을 통해 무비 클립 속성을 가져오거나 설정할 수도 있습니다. 다음 코드는 앞에서 살펴본 `setProperty()` 예제와 동일한 내용입니다.

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

`tellTarget()` 함수에 대한 자세한 내용은 [12페이지의 “다른 타임라인 제어”](#)를 참조하십시오.

## 다른 타임라인 제어

타임라인에 대한 경로를 지정하려면 슬래시(/) 구문과 도트(.)를 사용하여 경로 참조를 만듭니다. 또한, `_levelN`, `_root` 또는 `_parent`와 같은 Flash 5 표기법을 사용하여 특정 무비 레벨, 응용 프로그램의 루트 타임라인 또는 부모 타임라인을 각각 참조할 수 있습니다.

예를 들어, 사용자가 만든 SWF 파일의 기본 타임라인에 `box`라는 이름의 무비 클립 인스턴스가 있다고 가정해 봅시다. `box` 인스턴스에는 `cards`라는 또 다른 무비 클립 인스턴스가 포함되어 있습니다. 다음 예제는 기본 타임라인의 무비 클립 `cards`를 대상으로 합니다.

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

다음 예제는 무비 클립 `cards`의 기본 타임라인을 대상으로 합니다.

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

다음 예제는 부모 무비 클립 `cards`를 대상으로 합니다.

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

# 변수 사용

타임라인에서 변수를 지정하려면 슬래시(/)과 함께 도트(.)와 콜론(:)을 사용합니다. 도트 표기법을 사용할 수도 있습니다.

다음 코드는 기본 타임라인의 car 변수를 참조합니다.

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

다음 예제는 기본 타임라인에 있는 무비 클립 인스턴스에서 car 변수를 표시합니다.

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

다음 예제는 현재 타임라인에 있는 무비 클립 인스턴스에서 car 변수를 표시합니다.

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

# 배열 에뮬레이션

배열은 변수 및 값과 같은 정보를 정렬된 목록으로 만들고 조작하는 데 유용합니다. 그러나 Flash Lite 1.1에서는 기본 배열 데이터 구조가 지원되지 않습니다. Flash Lite(및 Flash 4) 프로 그래밍에서는 기본 배열 기능을 대신하여 문자열 처리를 통한 배열 에뮬레이션 기법을 일반적으로 사용하고 있습니다. 에뮬레이션된 배열을 의사 배열이라고도 합니다. 의사 배열 처리의 핵심인 eval() ActionScript 함수를 사용하면 해당 이름으로 변수, 속성 또는 무비 클립에 액세스할 수 있습니다. 자세한 내용은 19페이지의 “eval() 함수 사용”을 참조하십시오.

의사 배열은 일반적으로 두 개 이상의 변수로 구성되며, 동일한 기본 이름에 숫자 접미어가 붙은 형태입니다. 접미어는 각 배열 요소의 인덱스 역할을 합니다.

예를 들어, 다음과 같은 ActionScript 변수를 만든다고 가정해 봅시다.

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

다음 코드를 사용하여 의사 배열의 요소에 대해 반복 작업을 수행할 수 있습니다.

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

기존 변수를 참조하는 것은 물론이고 변수 대입 시 왼쪽에 eval() 함수를 사용하여 런타임에서 변수를 만들 수도 있습니다. 예를 들어, 게임 점수 순위 목록을 관리하는 코드를 작성해야 할 경우 게임을 마칠 때마다 게임 점수가 목록에 추가되도록 해야 합니다.

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

이 코드는 실행될 때마다 순위 목록에 새 항목을 추가하고 `scoreIndex` 변수를 증가시켜 목록에서 각 항목의 인덱스를 결정합니다. 예를 들어, 다음과 같은 변수로 끝낼 수 있습니다.

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

## 텍스트 및 문자열 작업

Flash Lite에서는 텍스트 작업을 위해 몇 가지 기본적인 `ActionScript` 명령과 속성을 제공합니다. 텍스트 필드의 값을 가져오고 설정할 수 있으며 문자열을 결합할 수 있고 텍스트 문자열을 URL 인코딩 또는 URL 디코딩할 수 있으며 스크롤 텍스트 필드를 만들 수 있습니다.

이 단원에서 설명하는 항목은 다음과 같습니다.

- [14페이지의 “문자열 결합”](#)
- [14페이지의 “스크롤 텍스트”](#)

## 문자열 결합

Flash Lite에서 문자열을 결합하려면 다음 예제와 같이 `add` 연산자를 사용합니다.

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

## 스크롤 텍스트

동적 텍스트 필드 및 입력 텍스트 필드의 `scroll` 속성을 사용하여 필드의 현재 스크롤 위치를 가져오거나 설정할 수 있습니다. 또한 `maxscroll` 위치를 사용하여 텍스트 필드의 현재 스크롤 위치를 최대 스크롤 위치에 대해 상대적인 값으로 지정할 수 있습니다. 스크롤 텍스트 필드를 만드는 방법에 대한 예제는 Flash Lite 응용 프로그램 개발의 “스크롤 텍스트 만들기”를 참조하십시오.

# call() 함수를 사용하여 함수 만들기

Flash Lite에서는 사용자 정의 함수를 정의하거나 호출할 수는 없지만(Flash Player 5 이상 버전에서는 가능) call() ActionScript 함수를 사용하여 타임라인에서 임의의 프레임에 있는 코드를 실행할 수 있습니다. 이 방법을 사용하면 자주 사용하는 코드를 한 곳에 캡슐화하여 쉽게 관리할 수 있습니다.

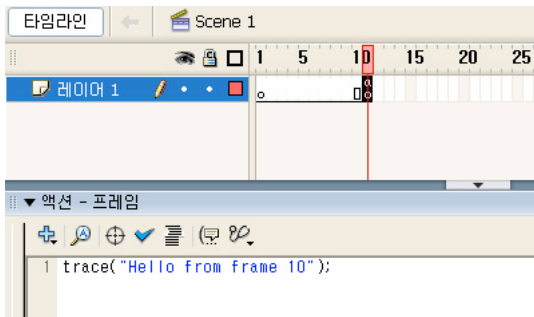
call() 함수는 프레임 번호 또는 프레임 레이블을 매개 변수로 사용합니다. 예를 들어, 다음 ActionScript는 프레임 레이블이 moveUp인 프레임에 있는 코드를 호출합니다.

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

call() 함수는 동기적으로 작동합니다. 즉, call() 함수 호출 뒤에 오는 모든 ActionScript는 지정된 프레임에서 모든 ActionScript 실행이 완료될 때까지 실행되지 않습니다.

## 다른 프레임에 있는 ActionScript를 호출하려면

1. 새 Flash 문서에서 프레임 10에 키프레임을 삽입합니다.



2. 삽입한 키프레임을 선택하고 **액션** 패널을 열어(윈도우 > 액션) 다음 코드를 입력합니다.  
`trace("Hello from frame 10");`

3. 프레임 1에서 키프레임을 선택하고 **액션** 패널에 다음 코드를 입력합니다.

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

이 코드는 프레임 1의 재생 헤드를 중단하고 프레임 10에 있는 코드를 호출합니다.

#### 4. Adobe® Device Central™에서 응용 프로그램을 테스트합니다.

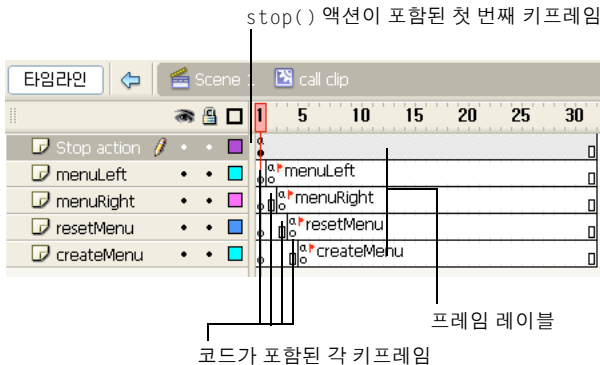
에뮬레이터에 “Hello from frame 10” 메시지가 표시됩니다.

또한 다른 타임라인(예: 무비 클립 타임라인)에 있는 코드를 호출할 수도 있습니다. 코드를 실행하려면 무비 클립 인스턴스 이름 뒤에 콜론을 입력한 다음 프레임 번호 또는 레이블을 지정합니다. 예를 들어, 다음 ActionScript는 callClip이라는 무비 클립 인스턴스에서 프레임 레이블이 moveUp인 프레임에 있는 코드를 호출합니다.

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

이 기법은 호출 클립 또는 함수 클립(자주 사용하는 코드를 캡슐화하기 위한 목적의 무비 클립)을 만들 때 종종 사용됩니다. 호출 클립에는 만들려는 각 함수에 대한 키프레임이 포함됩니다. 각 키프레임의 레이블은 사용 목적을 의미하는 이름으로 지정하는 것이 일반적입니다. 새로운 키프레임마다 새 레이어를 만드는 것이 좋으며, 각 레이어의 이름은 키프레임에 지정된 프레임 레이블과 같은 이름으로 지정하는 것이 좋습니다.

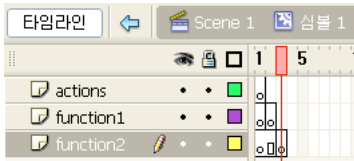
다음 그림은 예제로 만든 호출 클립의 타임라인을 보여줍니다. 호출 클립의 첫 번째 키프레임은 항상 stop() 액션을 포함하고 있어 재생 헤드가 타임라인의 여러 프레임에 걸쳐 계속 해서 반복되지 않도록 합니다. 이후 키프레임에는 각 “함수”에 대한 코드를 포함하고 있습니다. 각 함수 키프레임은 함수의 기능을 식별할 수 있는 레이블이 지정되어 있습니다. 각 함수 키프레임을 각각 별도의 레이어에 삽입하면 호출 클립을 쉽게 편집하고 표시할 수 있습니다.



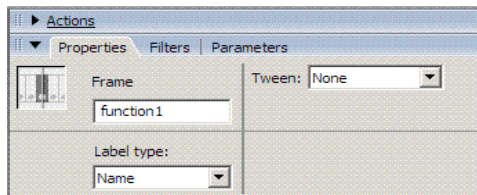
다음 절차에서는 호출 클립을 만들고 사용하는 방법을 설명합니다.

## 호출 클립을 만들고 사용하려면

1. Adobe® Flash® CS3에서 Flash Lite 1.1 Symbian Series 60 문서 템플릿을 사용하여 새 문서를 만듭니다.
2. **삽입 > 새 심볼**을 선택합니다.
3. **새 심볼 생성** 대화 상자의 **이름** 텍스트 상자에 **호출 클립**을 입력하고 **확인**을 클릭합니다. 그러면, 무비 클립이 편집 모드로 열립니다.
4. 타임라인 윈도우에서 레이어 추가 버튼을 두 번 클릭하여 두 개의 새로운 레이어를 삽입합니다.  
맨 위에 있는 레이어 이름을 **Actions**라고 지정하고 두 번째 레이어는 **function1** 그리고 세 번째 레이어는 **function2**라고 지정합니다.
5. 아래 그림과 같이 **function1** 레이어의 프레임 2에 키프레임을 삽입하고 **function2** 레이어의 프레임 3에 또 다른 키프레임을 삽입합니다.



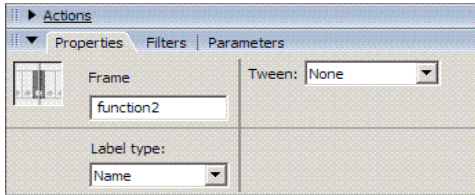
6. Actions 레이어에서 키프레임을 선택하고 **액션** 패널을 엽니다.
7. **액션** 패널에 `stop()` 액션을 추가합니다.
8. **function1** 레이어의 프레임 2에 있는 키프레임을 선택하고 다음 작업을 수행합니다.
  - a. 속성 관리자의 **프레임 레이블** 텍스트 상자에 **function1**을 입력합니다.



- b. **액션** 패널(윈도우 > 액션)에 다음 코드를 입력합니다.

```
trace("function1 was called.");
```

9. function2 레이어의 프레임 3에 있는 키프레임을 선택하고 다음 작업을 수행합니다.
  - a. 속성 관리자의 **프레임 레이블** 텍스트 상자에 **function2**를 입력합니다.



- b. **액션** 패널(윈도우 > 액션)에 다음 코드를 입력합니다.

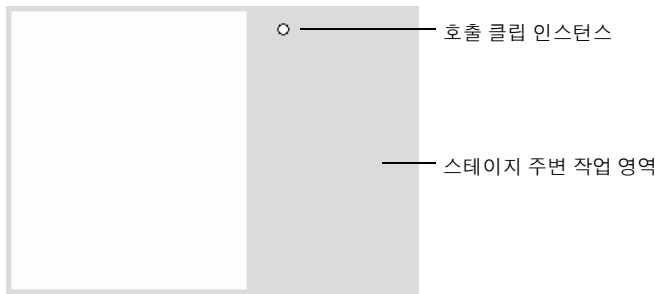
```
trace("function2 was called.");
```

10. Ctrl+E(Windows) 또는 Command+E(Macintosh)를 눌러 기본 타임라인으로 이동합니다.
11. **보기 > 작업 영역**을 선택하여 스테이지 주변에 작업 영역이 나타나도록 문서의 보기를 설정합니다.

호출 클립은 사용자에게 보여 질 필요가 없기 때문에 작업 영역에 배치할 수 있습니다.

12. **라이브러리** 패널(윈도우 > 라이브러리)을 열고 **호출 클립** 심볼을 스테이지 주변에 있는 작업 영역으로 드래그합니다.

호출 클립에는 시각적인 요소가 없기 때문에 스테이지로 드래그하면 무비 클립의 등록 포인트를 나타내는 작은 원 모양으로 나타납니다.



**참고**

호출 클립의 타임라인에서 첫 번째 키프레임에 간단한 텍스트나 시각적 요소를 추가하면 스테이지에서 호출 클립을 쉽게 식별할 수 있습니다.

13. 속성 관리자에서 **인스턴스 이름** 텍스트 상자에 **callClip**을 입력합니다.

14. 타임라인에서 **ActionScript**라는 레이어에서 프레임 1을 선택합니다.

15. 액션 패널에 다음 코드를 입력합니다.

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

16. 에플레이터에서 응용 프로그램을 테스트합니다(**컨트롤 > 무비 테스트**).

다음 텍스트가 표시됩니다.

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

## eval() 함수 사용

eval() 함수를 사용하면 런타임에서 변수 및 무비 클립 인스턴스를 동적으로 참조할 수 있습니다. eval() 함수는 매개 변수로 문자열 표현식을 사용하며 해당 표현식에서 나타내는 변수의 값 또는 무비 클립에 대한 참조를 반환합니다.

예를 들어, 다음 코드는 name ActionScript 변수의 값을 평가하고 그 결과 값을 nameValue에 지정합니다.

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

eval() 함수는 종종 for() 루프 및 add(문자열 결합) 연산자와 함께 사용되어 문자열 기반의 배열을 만듭니다. Flash Lite에서는 기본 배열 데이터 구조를 지원하지 않기 때문에 이와 같은 방식으로 배열 기능을 대체합니다. 자세한 내용은 13페이지의 “배열 애니메이션”을 참조하십시오.

eval() 함수를 사용하여 무비 클립 인스턴스를 이름으로 참조할 수도 있습니다. 예를 들어, clip1, clip2 및 clip3이라는 이름으로 세 개의 무비 클립을 갖고 있다고 가정해 봅시다. 다음 for() 루프는 각 클립의 x 위치를 10픽셀씩 증가시킵니다.

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



## 일반적인 스크립팅 작업

이 장에서는 사용자의 장치를 사용하여 작업하는 데 필요한 일반적인 Flash Lite 스크립팅 작업에 대해 설명합니다. 예를 들어, 장치 기능 정보 가져오기, 전화 걸기와 텍스트 메시지 전송 및 네트워크 상태 확인과 같은 작업에 대해 설명합니다.

이 장에서 설명하는 항목은 다음과 같습니다.

장치 및 플랫폼 기능 확인.....	21
웹 페이지 열기.....	22
전화 걸기.....	22
텍스트 또는 멀티미디어 메시지 전송.....	23
전자 메일 메시지 전송.....	24
외부 SWF 파일 로드.....	24
외부 데이터 로드.....	24

### 장치 및 플랫폼 기능 확인

Adobe의 Macromedia Flash Lite 1.1에는 특정 장치에서 실행되는 Flash Lite 응용 프로그램에 사용할 수 있는 기능에 대한 정보를 제공하는 다양한 ActionScript 변수가 포함되어 있습니다. 예를 들어, `_capLoadData` 변수는 해당 장치가 외부 데이터 로드를 지원하는지 여부를 나타내며, `_capSMS` 변수는 해당 장치가 SMS(Short Message Service) 메시지 전송을 지원하는지 여부를 나타냅니다. 기능 변수의 전체 목록은 Flash Lite 1.x ActionScript 언어 참조 설명서의 “기능”을 참조하십시오.

일반적으로 특정 기능을 실제로 사용하기에 앞서 장치에서 해당 기능을 지원하는지 확인하기 위한 목적으로 기능 변수를 사용합니다. 예를 들어, `loadVariables()` 함수를 사용하여 웹 서버에서 데이터를 다운로드하는 응용 프로그램을 개발하려 한다고 가정해 봅시다. 데이터 로드를 시도하기에 앞서 다음과 같이 `_capLoadData` 변수를 확인하여 해당 장치에서 이 기능을 지원하는지 확인합니다.

```

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

```

Flash Lite에서는 기본 SWF 파일의 루트 타임라인에 기능 변수를 정의합니다. 그러므로 다른 타임라인(예: 무비 클립 타임라인)에서 이러한 변수에 액세스하려면 해당 변수에 대한 경로를 정해 주어야 합니다. 예를 들어, 다음 예제는 슬래시(/)를 사용하여 `_capSMS` 변수에 대한 정규화된 경로를 제공합니다.

```
canSendSMS = /:_capSMS
```

## 웹 페이지 열기

`getURL()` 명령을 사용하여 장치의 웹 브라우저에서 웹 페이지를 열 수 있습니다. 이 방법은 데스크톱 Flash 응용 프로그램에서 웹 페이지를 여는 것과 동일합니다. 예를 들어, 다음 명령을 사용하면 Adobe 웹 페이지가 열립니다.

```
getURL("http://www.adobe.com");
```

Flash Lite에서는 프레임 또는 이벤트 핸들러마다 단 하나의 `getURL()` 액션만 처리합니다. 일부 통화 장치에서는 `getURL()` 액션을 키 누르기 이벤트로만 제한하여 `getURL()` 호출이 키 눌림 이벤트 핸들러 내에서 트리거되었을 경우에만 처리됩니다. 이러한 상황에서도 키 누르기 이벤트 핸들러마다 단 하나의 `getURL()` 액션만 처리됩니다. 스테이지에 있는 버튼 인스턴스에 첨부된 다음 코드는 사용자가 해당 장치에서 **Select** 버튼을 누를 때 웹 페이지를 엽니다.

```

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

```

## 전화 걸기

Flash Lite 응용 프로그램에서 전화 걸기 작업을 하려면 `getURL()` 함수를 사용합니다. 일반적으로는 웹 페이지를 여는 데 이 함수를 사용합니다. 그러나 이 경우에는 프로토콜로 `tel:`을 http 대신 지정한 다음 전화를 걸 전화 번호를 지정합니다. 이 함수를 호출할 때 Flash Lite는 지정한 번호로 전화를 걸 것인지 사용자의 승인을 요청하는 확인 대화 상자를 표시합니다.

다음 코드는 555-1212번으로 전화 걸기를 시도합니다.

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

Flash Lite에서는 프레임 또는 이벤트 핸들러마다 단 하나의 `getURL()` 액션만 처리합니다. 일부 통화 장치에서는 `getURL()` 액션을 키 누르기 이벤트로만 제한하여 `getURL()` 호출이 키 눌림 이벤트 핸들러 내에서 트리거되었을 경우에만 처리됩니다. 이러한 상황에서도 키 누르기 이벤트 핸들러마다 단 하나의 `getURL()` 액션만 처리됩니다. 다음 예제는 사용자가 해당 장치에서 **Select** 버튼을 누를 때 전화 걸기를 시작합니다.

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

## 텍스트 또는 멀티미디어 메시지 전송

Flash Lite를 사용하여 SMS(Short Message Service) 또는 MMS(Multimedia Message Service) 메시지를 전송할 수 있습니다. Flash Lite 응용 프로그램에서 SMS 또는 MMS 메시지를 전송하려면 `getURL()` 명령을 사용하여 일반적인 `http` 프로토콜 대신 `sms:` 또는 `mms:` 프로토콜로 전달한 다음 메시지를 보낼 전화 번호를 지정합니다.

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

다음 코드와 같이 선택적으로 URL 쿼리 문자열에 메시지 본문을 지정할 수도 있습니다.

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

MMS 메시지를 전송하려면 다음과 같이 `sms:` 대신 `mms:` 프로토콜을 사용합니다.

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



Flash Lite에서 MMS 메시지에 첨부 파일은 지정할 수 없습니다.

Flash Lite에서는 프레임 또는 이벤트 핸들러마다 단 하나의 `getURL()` 액션만 처리합니다. 일부 통화 장치에서는 `getURL()` 액션을 키 누르기 이벤트로만 제한하여 `getURL()` 호출이 키 눌림 이벤트 핸들러 내에서 트리거되었을 경우에만 처리됩니다. 이러한 상황에서도 키 누르기 이벤트 핸들러마다 단 하나의 `getURL()` 액션만 처리됩니다. 다음은 사용자가 해당 장치에서 **Select** 버튼을 누를 때 SMS 메시지를 전송하는 예제입니다.

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

## 전자 메일 메시지 전송

Flash Lite를 사용하여 전자 메일 메시지를 전송할 수 있습니다. 전자 메일 메시지를 전송하려면 `getURL()` 명령을 `mailto:` 프로토콜로 전달한 다음 수신자의 전자 우편 주소를 덧붙입니다. 선택적으로 다음과 같이 URL의 쿼리 문자열에 메시지 제목과 본문을 지정할 수도 있습니다.

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

쿼리 문자열에 메시지 본문만 지정하려면 다음 코드를 사용합니다.

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

## 외부 SWF 파일 로드

`loadMovie()` 함수를 사용하여 네트워크 또는 로컬 파일에서 SWF 파일을 로드할 수 있습니다. 이 기능은 Flash Lite 1.1 이상에서만 사용할 수 있습니다. 외부 SWF 파일을 로딩할 때 아래와 같은 제한 사항이 적용됩니다.

- Flash Lite에서는 다른 Flash Lite 1.0 또는 Flash Lite 1.1 SWF 파일이나 Flash 4 이하 버전으로 포맷된 SWF 파일을 로드할 수 있습니다. 그외 다른 포맷의 SWF 파일(예: Flash Player 6 SWF 파일)을 로드하려고 시도하면 Flash Lite에서 런타임 오류가 발생합니다.
- Flash Lite에서 JPEG 또는 GIF 이미지와 같은 외부 이미지 파일을 직접 로드할 수 없습니다. 이러한 미디어 유형을 로드하려면 이미지 데이터를 SWF 파일 포맷으로 변환해야 합니다. 이미지 파일을 새 문서로 가져온 다음 Flash Lite 또는 Flash 4 SWF 파일 포맷으로 내보내는 방법으로 Flash 제작 도구를 사용하여 “사용자가 직접” 변환하거나, 타사 유틸리티를 사용하여 변환 작업을 자동으로 처리할 수도 있습니다.

SWF 파일 로드에 대한 더 자세한 정보는 **Flash Lite 1.x ActionScript 언어 참조 설명서**의 `loadMovie()`를 참조하십시오.

## 외부 데이터 로드

외부 데이터를 Flash Lite 응용 프로그램에 로드하려면 `loadVariables()` 함수를 사용합니다. 네트워크를 통해(HTTP 주소 사용) 데이터를 로드하거나 로컬 파일 시스템에서 데이터를 로드할 수 있습니다. 이 기능은 Flash Lite 1.1 이상에서만 사용할 수 있습니다.

이 단원에서는 `loadVariables()` 함수를 사용하여 외부 파일에서 데이터를 로드하고, 로드한 데이터를 동적 텍스트 필드에 표시하는 방법을 보여줍니다. 우선 앰퍼샌드(&) 기호로 구분된 5개의 이름 값 쌍을 포함하는 텍스트 파일 포맷의 데이터 파일을 만듭니다. 그런 다음 이 텍스트 파일에 포함된 데이터를 로드하고 표시하는 Flash Lite 응용 프로그램을 만듭니다.

이 예제에서는 데이터 파일과 SWF 파일이 모두 동일한 폴더(에뮬레이터에서 테스트하는 경우에는 컴퓨터, 실제 장치에서 테스트 하는 경우에는 장치의 메모리 카드)에 있는 것으로 가정합니다. 장치에서 응용 프로그램을 테스트하려면 다음 중 하나를 수행해야 합니다.

- 텍스트 파일을 장치로 전송하고 SWF 파일과 동일한 폴더에 배치합니다.
- 텍스트 파일을 웹 서버의 URL에 게시(예: `www.your-server.com/data.txt`)합니다. 그런 다음, 다음과 같이 샘플 응용 프로그램에서 이 URL을 가리키도록 `loadVariables()` 호출을 수정합니다.

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

네트워크를 통해 데이터를 로드하는 응용 프로그램 예제는 Flash 샘플의 “Flash Lite 뉴스 읽기 프로그램”을 참조하십시오.

### 데이터 파일을 만들려면

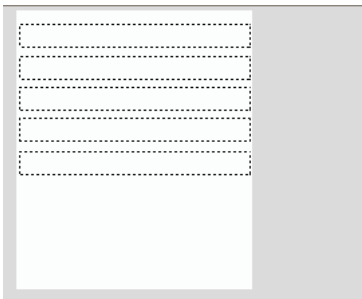
1. 텍스트 편집기(예: Notepad 또는 SimpleText)를 사용하여 다음 텍스트를 포함하는 파일을 만듭니다.

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

2. 파일을 `data.txt`라는 이름으로 저장합니다.

### 데이터를 로드하는 Flash Lite 응용 프로그램을 만들려면

1. Flash Lite 1.1 Symbian Series 60 문서 템플릿으로 새 문서를 만듭니다.  
Flash Lite 문서 템플릿 사용에 대한 자세한 내용은 Flash Lite 1.x 시작의 “Flash Lite 문서 템플릿 사용”을 참조하십시오.
2. 파일을 앞에서 만든 텍스트 파일(`data.txt`)이 포함된 폴더에 `dataloading fla`라는 이름으로 저장합니다.
3. 타임라인에서 Content 레이어의 프레임 1을 선택합니다.
4. 텍스트 도구를 사용하여 스테이지에 다음 그림과 같이 5개의 동적 텍스트 필드를 만듭니다.



5. 첫 번째(맨 위에 있는) 텍스트 필드를 선택하고 속성 관리자의 **변수** 텍스트 상자에 item\_1을 입력합니다.  
이 변수 이름은 앞에서 만든 data.txt 파일에 첫 번째로 정의된 변수(item\_1=Hello) 이름과 대응됩니다.
6. 앞의 두 단계에서 설명한 것과 동일한 방법으로 나머지 4개의 텍스트 필드에 변수 이름을 item\_2, item\_3, item\_4 및 item\_5와 같이 지정합니다.
7. Shift 키를 누른 상태에서 각 텍스트 필드를 클릭하여 모든 텍스트 필드를 선택하고 **수정 > 심볼로 변환**을 선택합니다.
8. **심볼로 변환** 대화 상자에서 심볼 유형으로 **무비 클립**을 선택하고 **확인**을 클릭합니다.
9. 방금 만든 무비 클립을 선택하고 속성 관리자에서 **인스턴스 이름** 텍스트 상자에 **data\_clip**을 입력합니다.
10. 타임라인에서 Actions 레이어의 프레임 1을 선택하고 **액션 패널(윈도우 > 액션)**을 엽니다.
11. 다음 코드를 **액션** 패널에 입력합니다.  

```
loadVariables("data.txt", "data_clip");
```
12. 변경 사항을 저장(**파일 > 저장**)하고 응용 프로그램을 에뮬레이터에서 테스트(**컨트롤 > 무비 테스트**)합니다.

다음 그림과 같이 각 텍스트 필드가 텍스트 파일의 데이터로 채워지는 것을 볼 수 있습니다.



# 색 인

## 마

멀티미디어 메시지, 시작 23

메시지 전송 23

무비 클립

동적으로 참조 19

속성 가져오기 및 설정 11

문자열 결합 14

문자열, 결합 14

## 바

배열, 문자열을 사용하여 애플리케이션 13

변수

도트 구문 및 슬래시 구문 13

동적으로 참조 19

참조 13

## 사

스크롤 텍스트, 만들기 14

## 아

외부 데이터 로드 24

외부 SWF 파일 로드 24

웹 페이지 열기 22

웹 페이지, 열기 22

## 자

전자 메일 메시지, 시작 24

전화 걸기 22

전화 통화, 시작 22

## 타

타임라인, ActionScript를 사용하여 제어 12

텍스트 메시지, 시작 23

## 파

플랫폼 기능 변수, 정보 21

## 하

함수 클립, 만들기 15

함수, call()을 사용하여 애플리케이션 15

## C

call() 함수, 사용 15

## E

eval() 함수, 사용 19

## F

Flash Lite 1.x ActionScript

개요 7

사용할 수 없는 기능 9

지원되지 않는 Flash 4 ActionScript 8

1.0과 1.1의 차이점 8

## G

getURL() 함수

멀티미디어 메시지 시작 23

웹 페이지 열기 22

전자 메일 메시지 시작 24

전화 통화 시작 22

텍스트 메시지 시작 23

## L

loadMovie() 함수, 사용 24

loadVariables() 함수, 사용 24

