

# ACTIONSCRIPT™ 2.0

## 구성 요소 사용 설명서

© 2007 Adobe Systems Incorporated. All rights reserved.

## ActionScript 2.0 구성 요소 사용 설명서

이 안내서와 함께 배포되는 소프트웨어에 최종 사용자 계약서가 포함되는 경우, 안내서에 기술된 소프트웨어 및 이 안내서는 해당 사용권에 따라 제공되며 이러한 사용권 계약 조건에 따라서만 사용하거나 복사할 수 있습니다. 사용권에서 허용된 경우를 제외하고는 Adobe Systems Incorporated의 사전 서면 동의 없이 이 안내서의 어떠한 부분도 재생하거나 검색 시스템에 저장할 수 없으며 전자적, 기계적, 녹화 또는 그 외 어떠한 방법이나 형태로든 전송할 수 없습니다. 이 안내서의 내용은 함께 배포된 소프트웨어에 최종 사용자 사용권 계약서가 포함되지 않는 경우에도 저작권법의 보호를 받습니다.

이 안내서의 내용은 통지 없이 변경될 수 있고, 오직 정보 제공의 목적으로만 제공되며, 이 내용을 Adobe Systems Incorporated측의 계약으로 해석해서는 안 됩니다. Adobe Systems Incorporated는 이 안내서에 포함된 내용에서 발생할 수 있는 오류 또는 부정확성에 대한 어떠한 책임도 지지 않습니다.

사용자의 작업 프로젝트에 포함되는 기존 아트웍 또는 이미지는 저작권법의 보호를 받을 수 있습니다. 이러한 자료를 사용자의 작업에 무단으로 사용하면 해당 저작권 소유자의 권리를 침해할 수 있습니다. 저작권 소유자로부터 필요할 동의를 구하십시오.

샘플 템플릿에 사용된 회사 이름은 설명용으로만 표시되었으며, 실제 회사를 나타낼 의도가 없습니다.

Adobe®, Flash®, Flash® Player, Flash® Video 및 Macromedia®는 미국 및 기타 국가에서 Adobe Systems Incorporated의 등록 상표 또는 상표입니다.

Macintosh®는 미국 및 기타 국가에서 등록된 Apple Computer, Inc.의 상표입니다. Windows®는 미국 및 기타 국가에서 Microsoft Corporation의 상표 또는 등록 상표입니다. 그외 모든 상표는 해당 소유자의 자산입니다.

이 제품의 일부에는 Nellymoser에서 허가 받은 코드가 사용됩니다([www.nellymoser.com](http://www.nellymoser.com)).

**Sorenson  
Spark**

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

Flash CS3 비디오는 On2 TrueMotion 비디오 기술로 제공됩니다.

© 1992-2005 On2 Technologies, Inc. All Rights Reserved. <http://www.on2.com>.

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA. 미국 정부 최종 사용자에 대한 알림 내용. 소프트웨어 및 설명서는 48 C.F.R. §12.212 또는 48 C.F.R. §227.7202에 각각 표현된 “Commercial Computer Software (상업용 컴퓨터 소프트웨어)” 및 “Commercial Computer Software Documentation(상업용 컴퓨터 소프트웨어 설명서)” 및 구성되는 “Commercial Items(상업용 제품)”(48 C.F.R. §2.101 참조)입니다. 227.7202-4를 통해 48 C.F.R. §12.212 또는 48 C.F.R. §§227.7202-1을 따르는 Commercial Computer Software(상업용 컴퓨터 소프트웨어) 및 Commercial Computer Software Documentation(상업용 컴퓨터 소프트웨어 설명서)는 (a) Commercial Items(상업용 제품)로서만, (b) 본 규정 및 조건을 따르는 모든 최종 사용자에게 권한을 부여하는 조건으로 미국 정부 최종 사용자에게 사용권이 제공됩니다. 공개되지 않은 권리도 미국 저작권법의 보호를 받습니다. Adobe Systems Incorporated, 345 Park Avenue, San Jose, CA 95110-2704, USA. 미국 정부 최종 사용자를 대상으로 Adobe는 대통령령 11246 수정 조항, 1974년 베트남전 참전용사 복귀 원조법(38 USC 4212) 402조, 1973년 재활법 수정 조항 503조 및 41 CFR 부분 60-1에서 60-60, 60-250 및 60-741의 조례를 포함하여 적용 가능한 모든 고용 기회 평등법을 준수할 것을 약속합니다. 이러한 법규에 포함된 차별 시정 조항 및 조례는 참조를 통해 포함될 것입니다.

# 목 차

<b>소개</b> .....	<b>7</b>
이 설명서의 대상 .....	8
시스템 요구 사항 .....	8
설명서 .....	8
인쇄 규칙 .....	9
설명서에 사용된 용어 .....	9
추가 리소스 .....	9
<b>제1장: 구성 요소</b> .....	<b>11</b>
구성 요소 설치 .....	12
구성 요소 파일이 저장되는 위치 .....	14
구성 요소 파일 수정 .....	15
구성 요소 사용의 장점 .....	16
구성 요소 범주 .....	16
버전 2 구성 요소 아키텍처 .....	17
버전 2 구성 요소 기능 .....	18
컴파일된 클립 및 SWC 파일 .....	19
액세스 가능성 및 구성 요소 .....	20
<b>제2장: 구성 요소를 사용하여 응용 프로그램 만들기</b> .....	<b>21</b>
실수 만회 지습 과정에 대한 설명 .....	21
기본 페이지 작성 .....	23
Gift Ideas를 표시하도록 데이터 구성 요소 바인딩 .....	28
선물 세부 정보 표시 .....	32
체크 아웃 화면 만들기 .....	37
응용 프로그램 테스트 .....	45
완성된 응용 프로그램 보기 .....	45

<b>제3장: 구성 요소를 사용한 작업</b>	<b>47</b>
구성 요소 패널	48
Flash 문서에 구성 요소 추가	48
라이브러리 패널의 구성 요소	52
구성 요소 매개 변수 설정	52
구성 요소 크기 조절	54
Flash 문서에서 구성 요소 삭제	55
코드 힌트 사용	55
스크린에서 ActionScript 사용	56
스크린과 ActionScript의 상호 작용 방식	57
사용자 정의 포커스 탐색 기능 만들기	58
문서 내에서 구성 요소 심도 관리	59
실시간 미리 보기의 구성 요소	60
구성 요소에 프리로더 사용	61
구성 요소 로드	62
버전 1 구성 요소를 버전 2 아키텍처로 업그레이드	63
<b>제4장: 구성 요소 이벤트 처리</b>	<b>65</b>
리스너를 사용하여 이벤트 처리	66
이벤트 위임	74
이벤트 객체	77
on() 이벤트 핸들러 사용	78
<b>제5장: 구성 요소 사용자 정의</b>	<b>79</b>
스타일을 사용하여 구성 요소 색상 및 텍스트 사용자 정의	80
구성 요소 스키닝	93
테마	104
스키닝 및 스타일을 통합하여 구성 요소 사용자 정의	115
<b>제6장: 구성 요소 만들기</b>	<b>121</b>
구성 요소 소스 파일	121
구성 요소 구조 개요	122
첫 번째 구성 요소 만들기	123
부모 클래스 선택	131
구성 요소 무비 클립 만들기	134
ActionScript 클래스 파일 만들기	138
기존 구성 요소를 자신의 구성 요소에 통합	166
구성 요소 내보내기 및 배포	175
구성 요소 개발의 마지막 단계	178

<b>제7장: 컬렉션 속성</b> .....	<b>181</b>
컬렉션 속성 정의 .....	182
간단한 컬렉션 예제 .....	183
컬렉션 항목의 클래스 정의 .....	185
프로그래밍 방식으로 컬렉션 정보 액세스 .....	185
컬렉션이 있는 구성 요소를 SWC 파일로 내보내기 .....	188
컬렉션 속성이 있는 구성 요소 사용 .....	189
 <b>색인</b> .....	 <b>191</b>



Flash CS3 Professional은 고성능 웹 환경을 제공하는 표준 제작 도구입니다. 구성 요소는 이러한 환경을 제공하는 풍부한 인터넷 응용 프로그램을 구성하는 단위입니다. 구성 요소는 Flash에서 제작하는 동안 설정되는 매개 변수와 런타임에 구성 요소를 사용자 정의하는 데 사용할 수 있는 ActionScript 메서드, 속성 및 이벤트를 가진 무비 클립입니다. 구성 요소는 개발자가 코드를 다시 사용하거나 공유할 수 있게 하고 복잡한 기능을 캡슐화하여 디자이너가 ActionScript를 사용하지 않고도 해당 기능을 사용하거나 사용자 정의할 수 있도록 디자인되었습니다.

Adobe Component Architecture 버전 2를 사용하여 만든 구성 요소를 사용하면 일관된 모양과 비헤이비어를 가진 강력한 응용 프로그램을 손쉽게 신속하게 만들 수 있습니다. 이 설명서에서는 버전 2 구성 요소를 사용하여 응용 프로그램을 구성하는 방법에 대해 설명합니다. 관련된 *ActionScript 2.0 구성 요소 언어 참조 설명서*에는 각 구성 요소의 API(Application Programming Interface)가 설명되어 있습니다. 여기서는 Flash 버전 2 구성 요소를 사용하는 시나리오와 단계별 샘플을 비롯하여 구성 요소 API에 대한 설명을 알파벳순으로 제공합니다.

Adobe에서 만든 구성 요소를 사용하거나 다른 개발자가 만든 구성 요소를 다운로드하거나 구성 요소를 직접 만들 수 있습니다.

이 장에서 설명하는 단원은 다음과 같습니다.

이 설명서의 대상 .....	8
시스템 요구 사항 .....	8
설명서 .....	8
인쇄 규칙 .....	9
설명서에 사용된 용어 .....	9
추가 리소스 .....	9

# 이 설명서의 대상

이 설명서는 Flash 응용 프로그램을 제작할 때 구성 요소를 사용하여 개발 시간을 단축하고자 하는 개발자를 위해 작성되었습니다. Flash에서 응용 프로그램을 개발하고 ActionScript를 작성하는 데 이미 능숙한 사용자에게 적합합니다.

ActionScript 작성에 익숙하지 않으면 문서에 구성 요소를 추가하고 속성 관리자나 구성 요소 관리자에서 매개 변수를 설정하고 **비헤이비어** 패널을 사용하여 이벤트를 처리할 수 있습니다. 예를 들어, ActionScript 코드를 작성하지 않고 버튼을 클릭할 때 웹 브라우저에서 URL을 여는 **웹 페이지로 이동** 비헤이비어를 Button 구성 요소에 첨부할 수 있습니다.

더욱 강력한 응용 프로그램을 만들고자 하는 프로그래머는 동적으로 구성 요소를 만들고 ActionScript를 사용하여 런타임에 속성을 설정하고 메시지를 호출하며 리스너 이벤트를 사용하여 이벤트를 처리할 수 있습니다.

자세한 내용은 47페이지의 제3장, “구성 요소를 사용한 작업”을 참조하십시오.

# 시스템 요구 사항

Adobe 구성 요소를 사용하기 위해서는 Flash 시스템 요구 사항만 충족하면 됩니다.

버전 2 구성 요소를 사용하는 SWF 파일은 Flash Player 6(6.0.79.0) 이상으로 보아야 하며, ActionScript 2.0용으로 제작해야 합니다(파일 > 제작 설정의 **Flash** 탭에서 설정 가능).

# 설명서

이 설명서에서는 구성 요소를 사용하여 Flash 응용 프로그램을 개발하는 방법에 대해 자세하게 설명합니다. 이 설명서는 독자가 Flash와 ActionScript에 대한 일반적인 지식을 가지고 있다고 가정합니다. Flash 및 관련 제품에 대한 설명서는 별도로 제공됩니다.

이 문서는 PDF 파일이나 온라인 도움말 형태로 사용할 수 있습니다. 온라인 도움말을 보려면 Flash를 시작하고 **도움말 > ActionScript 2.0 구성 요소 사용 설명서**를 선택하십시오.

Flash에 대한 자세한 내용은 다음 문서를 참조하십시오.

- *Flash 사용 설명서*
- *Adobe Flash에서 ActionScript 2.0 학습*
- *ActionScript 2.0 언어 참조 설명서*
- *ActionScript 2.0 구성 요소 언어 참조 설명서*

## 인쇄 규칙

이 설명서에는 다음과 같은 인쇄 규칙이 사용됩니다.

- 기울임 글꼴은 대체할 값을 나타냅니다(예: 폴더 경로의 값).
- 코드 글꼴은 메서드 및 속성 이름을 포함한 `ActionScript` 코드를 나타냅니다.
- 코드 *기울임 글꼴*은 대체해야 하는 코드 항목을 나타냅니다(예: `ActionScript` 매개 변수).
- 굵은 글꼴은 사용자가 입력하는 값을 나타냅니다.

## 설명서에 사용된 용어

이 설명서에는 다음과 같은 용어가 사용됩니다.

**런타임** Flash Player에서 코드가 실행 중인 상태를 나타냅니다.

**제작하는 동안** Flash 제작 환경에서 작업 중인 상태를 나타냅니다.

## 추가 리소스

Flash 최신 정보, 전문 사용자의 경험담, 고급 항목, 예제, 팁 및 기타 업데이트는 정기적으로 업데이트되는 Adobe DevNet 웹 사이트([www.adobe.com/kr/devnet](http://www.adobe.com/kr/devnet))를 참조하십시오. Flash에 대한 최신 소식 및 최신 프로그램을 얻으려면 이 웹 사이트를 자주 방문하십시오.

TechNotes, 설명서 업데이트 및 Flash 커뮤니티의 추가 리소스에 대한 링크는 Adobe Flash Support Center([www.adobe.com/support/flash](http://www.adobe.com/support/flash))를 참조하십시오.

ActionScript 용어, 구문 및 사용 방법에 대한 자세한 내용은 *Adobe Flash에서 ActionScript 2.0 학습* 및 *ActionScript 2.0 언어 참조 설명서*를 참조하십시오.

구성 요소를 사용하는 방법에 대한 기본적인 내용은 Adobe On Demand Seminar, Using UI Components([www.adobe.com/events/main.jsp](http://www.adobe.com/events/main.jsp))를 참조하십시오.



Flash CS3 Professional 구성 요소는 매개 변수를 사용하여 모양과 비헤이비어를 수정할 수 있는 무비 클립입니다. 구성 요소는 라디오 버튼이나 체크 상자와 같은 간단한 사용자 인터페이스 컨트롤이거나 스크롤 창과 같이 내용을 포함할 수 있으며, 응용 프로그램에서 포커스를 받을 객체를 제어하는 데 사용되는 FocusManager와 같이 비시각적일 수도 있습니다.

ActionScript에 대한 고급 지식이 없는 사람도 누구나 구성 요소를 사용하여 복잡한 Flash 응용 프로그램을 만들 수 있습니다. 사용자 정의 버튼, 콤보 상자 및 목록을 직접 만드는 대신 **구성 요소** 패널에서 구성 요소를 드래그하는 방법으로 응용 프로그램에 기능을 추가할 수 있습니다. 또한 디자인 상의 필요에 맞도록 구성 요소의 모양과 느낌을 손쉽게 사용자 정의할 수 있습니다.

Adobe Component Architecture 버전 2를 사용하여 만든 구성 요소를 사용하면 일관된 모양과 비헤이비어를 가진 강력한 응용 프로그램을 손쉽게 신속하게 만들 수 있습니다. 버전 2 아키텍처에는 모든 구성 요소의 기반이 되는 클래스, 구성 요소 모양을 사용자 정의하는 데 사용할 수 있는 스타일과 스킨 메커니즘, 브로드캐스터/리스너 이벤트 모델, 깊이와 포커스 관리, 액세스 가능성 구현 등이 포함됩니다.

**인포**

버전 2 구성 요소를 제작할 경우 제작 설정을 ActionScript 2.0으로 설정해야 합니다(파일 > 제작 설정, Flash 탭). ActionScript 1.0 또는 ActionScript 3.0을 사용하여 제작한 버전 2 구성 요소는 올바르게 실행되지 않습니다.

각 구성 요소에는 Flash에서 제작하는 동안 설정할 수 있는 미리 정의된 매개 변수를 비롯하여 런타임에 매개 변수 및 추가 옵션을 설정하는 데 사용할 수 있는 **API**(Application Programming Interface), 즉 고유한 ActionScript 메서드, 속성 및 이벤트 집합도 들어 있습니다.

Flash를 포함한 전체 구성 요소 목록을 보려면 [12페이지](#)의 “구성 요소 설치”를 참조하십시오. 이러한 구성 요소 외에 Flash 커뮤니티의 멤버가 만든 구성 요소를 Adobe Exchange ([www.adobe.com/cfusion/exchange/index.cfm](http://www.adobe.com/cfusion/exchange/index.cfm))에서 다운로드할 수도 있습니다.

이 장에서 설명하는 단원은 다음과 같습니다.

구성 요소 설치.....	12
구성 요소 파일이 저장되는 위치 .....	14
구성 요소 파일 수정.....	15
구성 요소 사용의 장점.....	16
구성 요소 범주.....	16
버전 2 구성 요소 아키텍처 .....	17
버전 2 구성 요소 기능.....	18
컴파일된 클립 및 SWC 파일 .....	19
액세스 가능성 및 구성 요소.....	20

## 구성 요소 설치

Flash를 처음 시작하면 일부 Adobe 구성 요소가 이미 설치되어 있습니다. 이들 구성 요소는 구성 요소 패널에서 볼 수 있습니다.

<b>예</b> <b>10</b>	버전 2 구성 요소를 보려면 제작 설정을 ActionScript 2.0으로 설정해야 합니다(파일 > 제작 설정, Flash 탭). 버전 2 구성 요소에는 기본 제작 서정이 표시되지 않습니다.
-----------------------	--

Flash에는 다음과 같은 구성 요소가 들어 있습니다.

- Accordion
- Alert
- Button
- CheckBox
- ComboBox
- DataGrid
- DataHolder
- DataSet
- DateChooser
- DateField
- FLVPlayback
- Label
- List
- Loader
- Media
- Menu

- MenuBar
- NumericStepper
- ProgressBar
- RadioButton
- RDBMSResolver
- ScrollPane
- TextArea
- TextInput
- Tree
- UIScrollBar
- WebServiceConnector
- Window
- XMLConnector
- XUpdateResolver

Flash에는 또한 `DataBinding` 클래스가 들어 있습니다. 이 클래스에 액세스하려면 **윈도우 > 공용 라이브러리 > 클래스**를 선택합니다.

### Flash 구성 요소를 보려면:

1. Flash를 시작합니다.
2. 제작 설정을 ActionScript 2.0으로 설정합니다.  
(파일 > 제작 설정 > **Flash** 탭을 선택하고 **ActionScript** 드롭다운 메뉴에서 **ActionScript 2.0**을 선택합니다.)
3. 윈도우 > 구성 요소를 선택하여 구성 요소 패널을 엽니다.
4. 사용자 인터페이스를 선택하여 트리를 확장한 다음 설치되어 있는 구성 요소를 봅니다.  
추가 구성 요소는 Adobe Exchange([www.adobe.com/cfusion/exchange](http://www.adobe.com/cfusion/exchange))에서 다운로드할 수도 있습니다. Exchange에서 다운로드된 구성 요소를 설치하려면 [www.adobe.com/exchange/em\\_download/](http://www.adobe.com/exchange/em_download/)에서 Macromedia Extension Manager를 다운로드하여 설치합니다.

Flash의 구성 요소 패널에 구성 요소가 나타날 수 있습니다. Windows 또는 Macintosh 컴퓨터에서 구성 요소를 설치하려면 다음 단계를 수행하십시오.

## Windows 기반 컴퓨터나 Macintosh 컴퓨터에 구성 요소를 설치하려면:

1. Flash를 종료합니다.
2. 구성 요소가 들어 있는 SWC 또는 FLA 파일을 하드 디스크의 다음 폴더로 이동합니다.
  - Windows의 경우: C:\Program Files\Adobe\Adobe Flash CS3\language\Configuration\Components
  - Macintosh의 경우: Macintosh HD/Applications/Adobe Flash CS3/Configuration/Components
3. Flash를 시작합니다.
4. 제작 설정을 ActionScript 2.0으로 설정합니다.  
(파일 > 제작 설정 > **Flash** 탭을 선택하고 **ActionScript** 드롭다운 메뉴에서 **ActionScript 2.0**을 선택합니다.)
5. 구성 요소 패널이 아직 열려 있지 않은 경우 윈도우 > 구성 요소를 선택하여 구성 요소 패널에서 구성 요소를 봅니다.

## 구성 요소 파일이 저장되는 위치

Flash 구성 요소는 응용 프로그램 수준의 Configuration 폴더에 저장됩니다.

이  
FO

이 폴더에 대한 자세한 내용은 Flash 사용 설명서의 “Flash와 함께 설치되는 Configuration 폴더”를 참조하십시오.

구성 요소는 다음 위치에 설치됩니다.

- Windows 2000 또는 Windows XP: C:\Program Files\Adobe\Adobe Flash CS3\language\Configuration\Components
- Mac OS X: Macintosh HD/Applications/Adobe Flash CS3/Configuration/Components

# 구성 요소 파일 수정

구성 요소의 소스 ActionScript 파일은 다음 위치에 있습니다.

- Windows 2000 또는 Windows XP: C:\Program Files\Adobe\Adobe Flash CS3\language\First Run\Classes\mx
- Mac OS X: Macintosh HD/Applications/Adobe Flash CS3/First Run/Classes/mx

Flash를 처음 시작하면 First Run 디렉토리의 파일이 Documents and Settings 경로에 복사됩니다. Documents and Settings 경로는 다음과 같습니다.

- Windows 2000 또는 Windows XP: C:\Documents and Settings\username\Local Settings\Application Data\Adobe\Adobe Flash CS3\language\Configuration\Classes\mx
- Mac OS X: Username/Library/Application Support/Adobe/Adobe Flash CS3/language/Configuration/Classes/mx

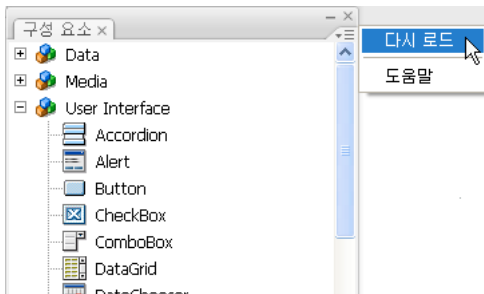
Flash가 시작될 때 Document and Settings 경로에 없는 파일은 First Run 디렉토리에서 Documents and Settings 경로로 자동으로 복사됩니다.

<b>예외</b>	소스 ActionScript 파일을 수정하려는 경우 Documents and Settings 경로에 있는 파일을 수정합니다. 수정 작업으로 인해 구성 요소가 "손상"되는 경우 Flash에서는 First Run 디렉토리에서 올바르게 작동하는 파일을 복사하여 사용자가 Flash를 닫고 다시 시작하면 원래의 기능을 복원합니다. 그러나 First Run 디렉토리에서 파일을 수정한 결과로 구성 요소가 "손상"되는 경우에는 Flash를 다시 설치하여 원본 파일을 올바르게 작동하는 파일로 복원해야 합니다.
-----------	--

구성 요소를 추가한 경우 구성 요소 패널을 새로 고쳐야 합니다.

## 구성 요소 패널의 내용을 새로 고치려면:

- 구성 요소 패널 메뉴에서 다시 로드를 선택합니다.



## 구성 요소 패널에서 구성 요소를 제거하려면:

- Configuration 폴더에서 MXP 또는 FLA 파일을 제거합니다.

## 구성 요소 사용의 장점

구성 요소를 사용하면 코딩 과정에서 응용 프로그램의 디자인 과정을 분리할 수 있습니다. 또한 다른 구성 요소를 만들 때 기존 코드를 다시 사용할 수 있으며, 다른 개발자가 만든 구성 요소를 다운로드한 후 설치하여 해당 코드를 다시 사용할 수도 있습니다.

구성 요소를 사용하면 코드 작성자는 디자이너가 응용 프로그램에 사용할 수 있는 기능을 만들 수 있습니다. 개발자는 자주 사용하는 기능을 구성 요소에 캡슐화할 수 있으며 디자이너는 속성 관리자나 구성 요소 관리자에서 매개 변수를 변경하여 구성 요소의 모양 및 비헤이비어를 사용자 정의할 수 있습니다.

Flash 개발자는 Adobe Exchange([www.adobe.com/go/exchange\\_kr](http://www.adobe.com/go/exchange_kr))를 사용하여 구성 요소를 교환할 수 있습니다. 구성 요소를 사용하면 복잡한 웹 응용 프로그램에서 각 요소를 처음부터 새로 구성할 필요 없이 원하는 구성 요소를 찾아 Flash 문서에 추가하는 방법으로 새 응용 프로그램을 만들 수 있습니다.

버전 2 아키텍처에 기반한 구성 요소는 스타일, 이벤트 처리, 스키닝, 포커스 관리, 깊이 관리 등의 핵심 기능을 공유합니다. 응용 프로그램에 첫 번째 버전 2 구성 요소를 추가하면 이 핵심 기능을 제공하기 위해 문서에 약 25KB가 추가됩니다. 이후에 다른 구성 요소를 추가하면 이들 구성 요소에 대해서도 처음의 25KB가 동일하게 다시 사용되므로 문서 크기는 많이 증가하지 않습니다. 구성 요소 업그레이드에 대한 자세한 내용은 [63페이지의 “버전 1 구성 요소를 버전 2 아키텍처로 업그레이드”](#)를 참조하십시오.

## 구성 요소 범주

Flash에 포함된 구성 요소는 다음과 같은 다섯 가지 범주로 구분됩니다. 괄호 안에 있는 ActionScript 소스 파일의 위치를 보면 속한 범주를 대략적으로 알 수 있습니다.

### ■ 데이터 구성 요소(mx.data.\*)

WebServiceConnector 및 XMLConnector 구성 요소와 같은 데이터 구성 요소는 데이터 소스에서 정보를 로드하고 조작하는 데 사용됩니다.

아이오

데이터 구성 요소의 소스 파일은 Flash와 함께 설치되지 않지만 지원되는 ActionScript 파일 중 일부는 설치됩니다.

### ■ FLVPlayback 구성 요소(mx.video.FLVPlayback)

FLVPlayback 구성 요소를 사용하면 Flash 응용 프로그램에 비디오 플레이어를 추가할 수 있습니다. 이 비디오 플레이어는 FVSS(Flash Video Streaming Service) 또는 Flash Media Server에서 HTTP를 통해 점진적 스트리밍 비디오를 재생할 수 있습니다.

### ■ 미디어 구성 요소(mx.controls.\*)

미디어 구성 요소를 사용하면 스트리밍 미디어를 재생하고 제어할 수 있습니다. MediaController, MediaPlayer 및 MediaDisplay는 미디어 구성 요소입니다.

- 사용자 인터페이스 구성 요소(mx.controls.\*)  
 RadioButton, CheckBox 및 TextInput 구성 요소와 같은 사용자 인터페이스 구성 요소(주로 “UI 구성 요소”라고 함)는 사용자가 응용 프로그램과 상호 작용하는 데 사용되는 컨트롤입니다.
- 관리자(mx.managers.\*)  
 관리자는 응용 프로그램에서 포커스나 깊이 같은 기능을 관리하는 데 사용되는 비시각적 구성 요소입니다. FocusManager, DepthManager, PopUpManager, StyleManager 및 SystemManager 구성 요소는 관리자 구성 요소입니다.
- 스크린(mx.screens.\*)  
 스크린 범주에는 Flash에서 양식과 슬라이드를 제어하는 데 사용되는 ActionScript 클래스가 포함됩니다.

구성 요소의 전체 목록을 보려면 *ActionScript 2.0 구성 요소 언어 참조 설명서*를 참조하십시오.

## 버전 2 구성 요소 아키텍처

속성 관리자나 구성 요소 관리자를 통해 구성 요소 매개 변수를 변경하여 구성 요소의 기본 기능을 활용할 수 있습니다. 그러나 구성 요소를 더 강력하게 제어하기 위해서는 해당 구성 요소의 API를 사용하고 구성 요소가 어떻게 구성되었는지 이해하는 것이 중요합니다.

Flash 구성 요소는 Adobe Component Architecture 버전 2를 사용하여 만들어졌습니다.

버전 2 구성 요소는 Flash Player 6(6.0.79.0) 이상과 ActionScript 2.0에서 지원됩니다. 이러한 구성 요소가 버전 1 아키텍처를 사용하여 작성한 구성 요소(Flash MX 2004 이전에 출시된 모든 구성 요소)와 항상 호환되는 것은 아닙니다. 또한 원래 버전 1 구성 요소는 Flash Player 7에서 지원되지 않습니다. 자세한 내용은 [63페이지의 “버전 1 구성 요소를 버전 2 아키텍처로 업그레이드”](#)를 참조하십시오.

**애프** Flash MX UI 구성 요소는 Flash Player 7 이상에서 작동하도록 업데이트되었습니다. 이러한 업데이트된 구성 요소는 여전히 버전 1 아키텍처를 기반으로 합니다. 업데이트된 구성 요소는 Adobe Flash Exchange([www.adobe.com/go/v1\\_components\\_kr](http://www.adobe.com/go/v1_components_kr))에서 다운로드할 수 있습니다.

버전 2 구성 요소는 컴파일된 클립 심볼(SWC)로 **구성 요소** 패널에 포함됩니다. 컴파일된 클립은 코드가 컴파일된 구성 요소 무비 클립입니다. 컴파일된 클립은 편집할 수 없지만 해당 매개변수는 구성 요소와 마찬가지로 속성 관리자와 구성 요소 관리자에서 변경할 수 있습니다. 자세한 내용은 [19페이지의 “컴파일된 클립 및 SWC 파일”](#)을 참조하십시오.

버전 2 구성 요소는 ActionScript 2.0으로 작성됩니다. 각 구성 요소는 하나의 클래스이며, ActionScript 패키지마다 각각의 클래스가 포함됩니다. 예를 들어, 한 라디오 버튼 구성 요소는 패키지 이름이 mx.controls인 RadioButton 클래스의 인스턴스일 수 있습니다. 예를 들어, 라디오 버튼 구성 요소는 패키지 이름이 mx.controls인 RadioButton 클래스의 인스턴스입니다. 패키지에 대한 자세한 내용은 *Adobe Flash에서 ActionScript 2.0 학습*의 “패키지”를 참조하십시오.

Adobe Component Architecture 버전 2로 만든 대부분의 UI 구성 요소는 UIObject 및 UIComponent 클래스의 하위 클래스이며 해당 클래스의 모든 속성, 메서드 및 이벤트를 상속합니다. 대부분의 구성 요소는 다른 구성 요소의 하위 클래스이기도 합니다. 각 구성 요소의 상속 경로는 *ActionScript 2.0 구성 요소 언어 참조 설명서*의 해당 구성 요소 항목에 설명되어 있습니다.

예제

클래스 계층의 FlashPaper 파일은 Flash 샘플 페이지([www.adobe.com/go/learn\\_fl\\_samples\\_kr](http://www.adobe.com/go/learn_fl_samples_kr))를 참조하십시오.

모든 구성 요소는 동일한 이벤트 모델, CSS 기반 스타일 및 내장 테마 및 스키닝 메커니즘을 사용합니다. 스타일 및 스키닝에 대한 자세한 내용은 79페이지의 제5장, “구성 요소 사용자 정의”를 참조하십시오. 이벤트 처리에 대한 자세한 내용은 47페이지의 제3장, “구성 요소를 사용한 작업”을 참조하십시오.

버전 2 구성 요소 아키텍처에 대한 자세한 내용은 121페이지의 제6장, “구성 요소 만들기”를 참조하십시오.

## 버전 2 구성 요소 기능

이 단원에서는 구성 요소를 사용하여 Flash 응용 프로그램을 만드는 개발자의 입장에서 버전 2 구성 요소의 기능을 버전 1 구성 요소와 비교해 간략하게 설명합니다. 버전 1 아키텍처와 버전 2 아키텍처에서 구성 요소를 만드는 차이에 대한 자세한 내용은 121페이지의 제6장, “구성 요소 만들기”를 참조하십시오.

**구성 요소 관리자**를 사용하면 Adobe Flash와 Dreamweaver에서 제작하는 동안 구성 요소 매개 변수를 변경할 수 있습니다. 자세한 내용은 52페이지의 “구성 요소 매개 변수 설정”을 참조하십시오.

**리스너 이벤트 모델**을 사용하면 리스너가 이벤트를 처리할 수 있습니다. 자세한 내용은 65페이지의 제4장, “구성 요소 이벤트 처리”를 참조하십시오. Flash 속성 관리자에는 clickHandler 매개 변수가 없으므로 ActionScript 코드를 작성하여 이벤트를 처리해야 합니다.

**스킨 속성**을 사용하면 런타임에 개별 스킨(예: 위쪽 및 아래쪽 화살표 또는 체크 상자의 체크 표시)을 로드할 수 있습니다. 자세한 내용은 93페이지의 “구성 요소 스키닝”을 참조하십시오.

**CSS 기반 스타일**을 사용하면 여러 응용 프로그램에 일관된 모양과 느낌을 적용할 수 있습니다. 자세한 내용은 [80페이지](#)의 “스타일을 사용하여 구성 요소 색상 및 텍스트 사용자 정의”를 참조하십시오.

**테마**를 사용하면 라이브러리에서 특정 구성 요소 집합으로 미리 디자인된 모양을 드래그할 수 있습니다. 자세한 내용은 [104페이지](#)의 “테마”를 참조하십시오.

**Halo 테마**는 버전 2 구성 요소가 사용하는 기본 테마입니다. 자세한 내용은 [104페이지](#)의 “테마”를 참조하십시오.

**관리자 클래스**를 사용하면 응용 프로그램에서 손쉬운 방법으로 포커스와 깊이를 처리할 수 있습니다. 자세한 내용은 [58페이지](#)의 “사용자 정의 포커스 탐색 기능 만들기” 및 [59페이지](#)의 “문서 내에서 구성 요소 심도 관리”를 참조하십시오.

**기본 클래스인 UIObject와 UICComponent**는 이들 클래스를 확장하는 구성 요소에 기본 메서드, 속성 및 이벤트를 제공합니다. 자세한 내용은 *ActionScript 2.0 구성 요소 언어 참조 설명서*의 “UICComponent 클래스” 및 “UIObject 클래스”를 참조하십시오.

**SWC 파일로 패키징**하면 구성 요소를 쉽게 배포하고 코드를 숨길 수 있습니다. 자세한 내용은 [121페이지](#)의 [제6장](#), “구성 요소 만들기”를 참조하십시오.

**내장된 데이터 바인딩**은 구성 요소 관리자를 통해 사용할 수 있습니다. 자세한 내용은 *Flash 사용 설명서*의 “데이터 통합”을 참조하십시오.

ActionScript 2.0을 사용하는 **쉽게 확장할 수 있는 클래스 계층 구조**를 사용하면 고유한 네임스페이스를 만들고 필요에 따라 클래스를 가져오고 하위 클래스를 만들어 쉽게 구성 요소를 확장할 수 있습니다. 자세한 내용은 [121페이지](#)의 [제6장](#), “구성 요소 만들기” 및 *ActionScript 2.0 언어 참조 설명서*를 참조하십시오.

예제 10

Flash 8 이상에는 9-슬라이스(“Scale-9”이라고 함), 고급 앤티앨리어싱 및 비트맵 캐싱을 비롯하여 v2 구성 요소에서 지원되지 않는 기능이 몇 가지 있습니다.

## 컴파일된 클립 및 SWC 파일

**컴파일된 클립**은 미리 컴파일된 Flash 심볼과 ActionScript 코드로 구성된 패키지로, 변경되지 않을 심볼과 코드를 다시 컴파일하지 않기 위해 사용합니다. Flash에서 무비 클립을 “컴파일”하여 컴파일된 클립으로 변환할 수도 있습니다. 예를 들어, 자주 변경되지 않는 ActionScript 코드가 많이 들어 있는 무비 클립을 컴파일된 클립으로 만들면 컴파일된 클립은 컴파일하기 전의 무비 클립과 동일하게 작동하지만 컴파일된 클립은 일반 무비 클립보다 더 빠르게 표시되고 제작됩니다. 컴파일된 클립은 편집할 수 없지만 속성 관리자와 구성 요소 관리자에 속성이 나타납니다.

Flash에 포함된 구성 요소는 FLA 파일이 아니라 SWC 파일로 패키징화된 컴파일된 클립입니다. SWC는 구성 요소를 배포하기 위한 Adobe 파일 포맷으로, 컴파일된 클립, 구성 요소의 ActionScript 클래스 파일, 구성 요소에 대해 설명하는 다른 파일이 포함되어 있습니다. SWC 파일에 대한 자세한 내용은 175페이지의 “구성 요소 내보내기 및 배포”를 참조하십시오.

SWC 파일을 First Run\Components 폴더에 추가하면 구성 요소 패널에 해당 구성 요소가 나타납니다. 구성 요소 패널에서 스테이지에 구성 요소를 추가하면 컴파일된 클립 심볼이 라이브러리에 추가됩니다.

#### 무비 클립을 컴파일하려면:

- 라이브러리 패널에서 무비 클립을 마우스 오른쪽 버튼으로 클릭(Windows)하거나 Control 키를 누른 상태에서 클릭(Macintosh)한 다음 **컴파일된 클립으로 변환**을 선택합니다.

#### SWC 파일을 내보내려면:

- 라이브러리 패널에서 무비 클립을 선택하고 마우스 오른쪽 버튼으로 클릭(Windows)하거나 Control 키를 누른 상태에서 클릭(Macintosh)한 다음 **SWC 파일 내보내기**를 선택합니다.

아이오

FLA 구성 요소는 Flash에서 계속 지원됩니다.

## 액세스 가능성 및 구성 요소

장애를 가진 사용자도 사용할 수 있는 웹 콘텐츠에 대한 필요성이 커지고 있습니다. 스크린 판독기 소프트웨어는 스크린의 내용을 말로 설명해 주므로, 시각 장애를 가진 사용자도 이 소프트웨어를 사용하여 Flash 응용 프로그램에 포함된 시각적인 정보에 액세스할 수 있습니다.

구성 요소를 만들 때 작성자는 구성 요소와 스크린 판독기 소프트웨어 사이의 통신을 가능하게 하는 ActionScript를 작성할 수 있습니다. 그러면 개발자는 Flash에서 구성 요소를 사용하여 응용 프로그램을 만들 때 액세스 가능성 패널을 사용하여 각 구성 요소 인스턴스를 구성할 수 있습니다.

Adobe에서 만든 대부분의 구성 요소는 액세스 가능성을 고려하여 디자인되었습니다. 특정 구성 요소가 액세스 가능한지 알아 보려면 *ActionScript 2.0 구성 요소 언어 참조 설명서*에서 해당 구성 요소 항목을 확인하십시오. Flash에서 응용 프로그램을 만들 때는 각 구성 요소에 대해 코드 행 하나(`mx.accessibility.ComponentNameAccImpl.enableAccessibility();`)를 추가한 다음 **액세스 가능성** 패널에서 액세스 가능성 매개 변수를 설정해야 합니다. 구성 요소에 대한 액세스 가능성은 모든 Flash 무비 클립의 액세스 가능성과 동일한 방식으로 작동합니다.

Adobe에서 만든 대부분의 구성 요소는 키보드를 사용하여 탐색할 수도 있습니다.

*ActionScript 2.0 구성 요소 언어 참조 설명서*의 각 구성 요소 항목은 키보드를 사용하여 구성 요소를 제어할 수 있는지 여부를 나타냅니다.

# 구성 요소를 사용하여 응용 프로그램 만들기

구성 요소는 Flash 응용 프로그램을 만들 때 사용할 수 있도록 미리 작성된 Flash 요소입니다. 구성 요소에는 사용자 인터페이스 컨트롤, 데이터 액세스 및 연결 메커니즘, 미디어 관련 요소가 포함됩니다. 구성 요소를 사용하면 각 요소와 비헤이비어를 처음부터 만들 필요가 없으므로 Flash 응용 프로그램을 만들 때 시간과 노력을 절약할 수 있습니다.

이 장에는 Flash CS3 Professional에서 제공하는 구성 요소를 사용하여 Flash 응용 프로그램을 만드는 방법을 보여 주는 자습 과정이 포함되어 있습니다. Flash 제작 환경에서 구성 요소를 사용하는 방법 및 구성 요소와 ActionScript 코드가 상호 작용하도록 만드는 방법을 배우게 됩니다.

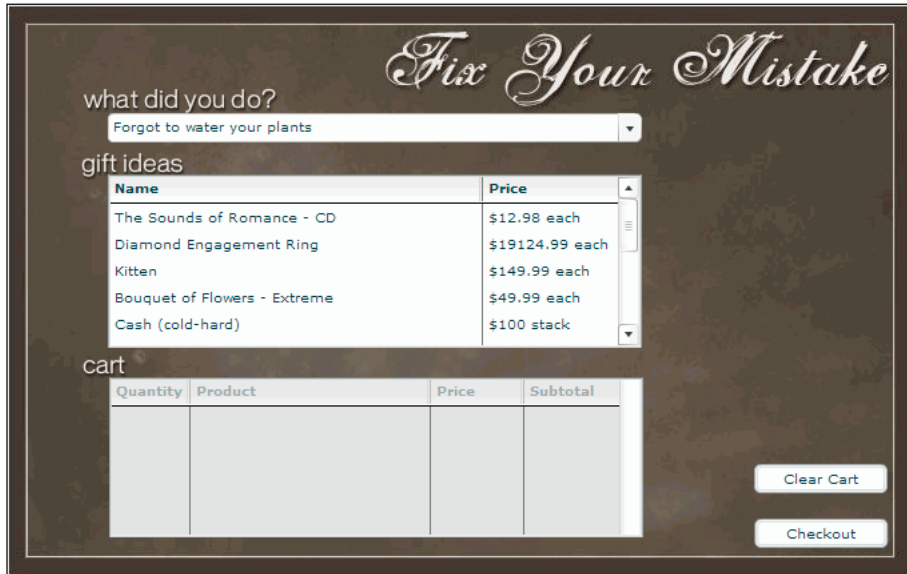
## 실수 만회 자습 과정에 대한 설명

이 자습 과정에서는 “실수 만회” 선물 서비스에 필요한 기본 온라인 쇼핑 응용 프로그램을 만드는 단계를 진행합니다. 이 서비스는 누군가에게 잘못된 행동을 했을 경우 이를 만회하기 위해 적당한 선물을 고를 수 있도록 도와주는 프로그램입니다. 이 응용 프로그램은 잘못된 심각도에 따라 적당한 품목을 고를 수 있도록 선물 목록을 필터링합니다. 사용자는 이 목록에서 원하는 품목을 장바구니에 추가한 다음 체크 아웃 페이지로 가서 지불, 배송 및 신용 카드 정보를 제공할 수 있습니다.

이 장에서 설명하는 단원은 다음과 같습니다.

실수 만회 자습 과정에 대한 설명.....	21
기본 페이지 작성.....	23
Gift Ideas를 표시하도록 데이터 구성 요소 바인딩.....	28
선물 세부 정보 표시.....	32
체크 아웃 화면 만들기.....	37
응용 프로그램 테스트.....	45
완성된 응용 프로그램 보기.....	45

응용 프로그램의 인터페이스를 만드는 데에는 ComboBox, DataGrid, TextArea, Button 등의 구성 요소가 사용됩니다. 인터페이스의 기본 페이지는 다음과 같습니다.



응용 프로그램은 ActionScript WebService 클래스를 사용하여 웹 서비스에 동적으로 연결하고 콤보 상자에 나타나는 잘못된 행동 목록(problems.xml)을 검색합니다. 또한 ActionScript를 사용하여 사용자와 응용 프로그램 간의 상호 작용을 처리합니다.

응용 프로그램은 데이터 구성 요소를 사용하여 인터페이스를 다른 데이터 소스로 연결합니다. 또한 선물 목록을 가져오기 위해 XMLConnector 구성 요소를 사용하여 XML 데이터 파일(products.xml)에 연결하고, DataSet 구성 요소를 사용하여 데이터를 필터링한 다음 데이터 격자로 보냅니다.

이 자습 과정을 진행하려면 Flash 제작 환경에 어느 정도 익숙해야 하고 ActionScript를 사용해 본 경험이 있어야 합니다. 또한 제작 환경에서 패널, 도구, 타임라인 및 라이브러리를 사용해 본 경험이 있어야 합니다. 이 자습 과정에서는 샘플 응용 프로그램을 만드는 데 필요한 모든 ActionScript가 제공되지만 스크립팅 개념을 이해하고 직접 응용 프로그램을 만들려면 ActionScript를 작성해 본 경험이 있어야 합니다.

완성된 응용 프로그램의 실행 버전을 보려면 [45페이지의 “완성된 응용 프로그램 보기”](#)를 참조하십시오.

샘플 응용 프로그램은 데모용이므로 실제 사용되는 응용 프로그램처럼 완전하지 않다는 점에 유의하시기 바랍니다.

# 기본 페이지 작성

다음 단계에 따라 기본 틀을 제공하는 시작 페이지에 구성 요소를 추가하여 응용 프로그램의 기본 페이지를 만듭니다. 그런 다음 구성 요소를 사용자 정의하는 **ActionScript** 코드를 추가하고, 응용 프로그램의 구성 요소를 조작할 수 있도록 **ActionScript** 클래스를 가져오고, 잘못된 행동 목록으로 콤보 상자를 채우기 위해 웹 서비스에 액세스합니다. 추가된 코드는 웹 서비스로부터 결과를 수신할 수 있도록 `dataProvider` 속성을 설정하여 콤보 상자를 채웁니다.

1. `first_app_start.fla` 파일을 엽니다. 샘플 `.fla` 파일은 Flash 샘플 페이지([www.adobe.com/go/learn\\_fl\\_samples\\_kr](http://www.adobe.com/go/learn_fl_samples_kr))를 참조하십시오.

이 파일에는 다음과 같은 시작 페이지가 포함되어 있습니다.



`start_app.fla` 파일에는 검정 배경 이미지와 텍스트 제목이 있는 **Background** 레이어, 응용 프로그램 섹션용 텍스트 레이블이 있는 **Text** 레이어, 첫 번째 프레임(홈)과 열 번째 프레임(체크 아웃)의 레이블이 있는 **Labels** 레이어가 포함되어 있습니다.

2. **파일 > 다른 이름으로 저장**을 선택합니다. 파일 이름을 변경하고 하드 디스크에 저장합니다.
3. 타임라인에서 **Labels** 레이어를 선택하고 레이어 삽입 버튼을 클릭하여 새 레이어를 추가합니다. 새 레이어의 이름을 **Form**으로 지정합니다. 이 레이어에 구성 요소 인스턴스를 배치합니다.

4. **Form** 레이어가 선택되어 있는지 확인합니다. **구성 요소** 패널(윈도우 > 구성 요소)에서 ComboBox 구성 요소를 찾습니다. ComboBox 인스턴스를 스테이지로 드래그하여 **What Did You Do?** 텍스트 아래에 놓습니다. 속성 관리자(윈도우 > 속성 > 속성)에서 인스턴스 이름으로 **problems\_cb**를 입력합니다. 폭으로 **400**(픽셀)을 입력합니다.  $x$  위치로 **76.0**을 입력하고  $y$  위치로 **82.0**을 입력합니다.

<b>이 요</b>	ComboBox 구성 요소 심볼이 라이브러리(윈도우 > 라이브러리)에 추가됩니다. 구성 요소 인스턴스를 스테이지로 드래그하면 구성 요소의 컴파일된 클립 심볼이 라이브러리에 추가됩니다. Flash의 모든 심볼과 마찬가지로 라이브러리 심볼을 스테이지로 드래그하여 구성 요소의 추가 인스턴스를 만들 수 있습니다.
----------------	---

5. DataGrid 구성 요소의 인스턴스를 **구성 요소** 패널에서 스테이지로 드래그합니다. **Gift Ideas** 텍스트 아래에 놓습니다. 인스턴스 이름으로 **products\_dg**를 입력합니다. 폭으로 **400**(픽셀)을 입력하고 높이로 **130**을 입력합니다.  $x$  위치로 **76.0**을 입력하고  $y$  위치로 **128.0**을 입력합니다.
6. DataSet 구성 요소의 인스턴스를 **구성 요소** 패널에서 스테이지 옆으로 드래그합니다. DataSet 구성 요소는 런타임에 응용 프로그램에 나타나지 않습니다. DataSet 아이콘은 Flash 제작 환경에서 사용하는 자리 표시자일 뿐입니다. 인스턴스 이름으로 **products\_ds**를 입력합니다.

**구성 요소** 패널에서 스테이지 옆으로 XMLConnector 구성 요소의 인스턴스를 드래그합니다. DataSet 구성 요소처럼 XMLConnector 구성 요소도 런타임에 응용 프로그램에 나타나지 않습니다. 인스턴스 이름으로 **products\_xmlcon**을 입력합니다. **속성 관리자**에서 **매개 변수** 탭을 클릭하고 URL 속성의 값으로 **http://www.flash-mx.com/mm/firstapp/products.xml**을 입력합니다. **direction** 속성에 대한 값을 클릭하여 콤보 상자를 활성화하고 아래쪽 화살표를 클릭한 다음 목록에서 **receive**를 선택합니다.

<b>이 요</b>	구성 요소 관리자(윈도우 > 구성 요소 관리자)를 사용하여 구성 요소의 매개 변수를 설정할 수도 있습니다. 구성 요소 관리자와 속성 관리자의 매개 변수 탭은 동일하게 작동합니다.
----------------	---

- URL은 응용 프로그램의 **Gift Ideas** 섹션에 나타나는 제품에 대한 데이터가 있는 외부 XML 파일을 지정합니다. 데이터 바인딩을 사용하여 XMLConnector, DataSet 및 DataGrid 구성 요소를 함께 바인딩하는 방법은 자습 과정의 뒷부분에서 설명합니다. DataSet 구성 요소는 외부 XML 파일의 데이터를 필터링하고 DataGrid 구성 요소는 필터링된 데이터를 표시합니다.
7. Button 구성 요소의 인스턴스를 **구성 요소** 패널에서 스테이지로 드래그합니다. 스테이지 오른쪽 아래 모서리에 놓습니다. 인스턴스 이름으로 **checkout\_button**을 입력합니다. **매개 변수** 탭을 클릭하고 label 속성의 값으로 **체크 아웃**을 입력합니다.  $x$  및  $y$  좌표로 **560.3**과 **386.0**을 각각 입력합니다.

## 구성 요소 클래스 가져오기

각 구성 요소는 해당 메서드와 속성을 정의하는 `ActionScript` 클래스 파일과 연관됩니다. 이 단원에서는 응용 프로그램의 구성 요소와 연관된 클래스를 가져오는 `ActionScript` 코드를 추가합니다. 이들 구성 요소 중 일부는 이미 스테이지에 인스턴스를 추가했습니다. 다른 구성 요소에 대해서는 자습 과정의 뒷부분에서 `ActionScript`를 추가하여 동적으로 인스턴스를 만듭니다.

`import` 문은 클래스 이름에 대한 참조를 만들고 구성 요소에 대한 `ActionScript` 작성 과정을 더욱 쉽게 만듭니다. `import` 문을 사용하면 패키지 이름을 포함하는 전체 이름 대신 클래스 이름으로 해당 클래스를 참조할 수 있습니다. 예를 들어, `import` 문으로 `ComboBox` 클래스 파일에 대한 참조를 만들면 `instanceName:mx.controls.ComboBox` 대신 `instanceName:ComboBox` 구문을 사용하여 콤보 상자의 인스턴스를 참조할 수 있습니다.

`패키지`는 클래스 파일을 포함하고 지정된 클래스 경로 디렉토리에 있는 디렉토리입니다. 와일드카드 문자를 사용하여 패키지의 모든 클래스에 대한 참조를 만들 수 있습니다. 예를 들어, `mx.controls.*` 구문을 사용하면 컨트롤 패키지의 모든 클래스에 대한 참조가 만들어 집니다. 와일드카드를 사용하면 검토할 때 응용 프로그램에서 사용하지 않은 클래스가 삭제되므로 크기가 더 커지지 않습니다.

이 자습 과정에 사용된 응용 프로그램의 경우 다음과 같은 패키지와 개별 클래스가 필요합니다.

**UI 구성 요소 컨트롤 패키지** 이 패키지에는 `ComboBox`, `DataGrid`, `Loader`, `TextInput`, `Label`, `NumericStepper`, `Button` 및 `CheckBox`를 비롯한 사용자 인터페이스 컨트롤 구성 요소의 클래스가 포함되어 있습니다.

**UI 구성 요소 컨테이너 패키지** 이 패키지에는 `Accordion`, `ScrollPane` 및 `Window`를 비롯한 사용자 인터페이스 컨테이너 구성 요소의 클래스가 포함되어 있습니다. 컨트롤 패키지와 마찬가지로 와일드카드를 사용하여 이 패키지에 대한 참조를 만들 수 있습니다.

**DataGridColumn 클래스** 이 클래스를 사용하면 `DataGrid` 인스턴스에 열을 추가하고 그 모양을 제어할 수 있습니다.

**WebService 클래스** 이 클래스는 문제 또는 잘못된 행동 목록으로 `ComboBox` 인스턴스를 채웁니다. 또한 이 클래스의 경우 `클래스` 공용 라이브러리에서 `WebServiceClasses` 항목을 가져와야 합니다. 이 항목에는 응용 프로그램의 SWF 파일을 컴파일하고 생성하는 데 필요한 컴파일된 클립(SWC) 파일이 포함되어 있습니다.

**Cart 클래스** 이 자습 과정과 함께 제공되는 사용자 정의 클래스이며 나중에 만들 장바구니의 기능을 정의합니다. `Cart` 클래스 파일의 코드를 조사하려면 응용 프로그램 FLA 및 SWF 파일이 포함된 `component_application` 폴더에 있는 `cart.as` 파일을 엽니다.

이러한 클래스를 가져오기 위해 `Actions` 레이어를 만들고 기본 타임라인의 첫 번째 프레임에 `ActionScript` 코드를 추가합니다. 이 자습 과정의 나머지 단계에서 응용 프로그램에 추가할 모든 코드는 `Actions` 레이어에 배치해야 합니다.

1. 클래스 라이브러리에서 `WebServiceClasses` 항목을 가져오려면 **윈도우 > 공용 라이브러리 > 클래스**를 선택합니다.

2. 클래스 라이브러리에서 응용 프로그램의 라이브러리로 `WebServiceClasses` 항목을 드래그합니다.

클래스 라이브러리에서 항목을 가져오는 것은 라이브러리에 구성 요소를 추가하는 것과 비슷합니다. 즉, 라이브러리에 클래스의 SWC 파일이 추가됩니다. 응용 프로그램에서 클래스를 사용하려면 라이브러리에 SWC 파일이 있어야 합니다.

3. 타임라인에서 **Form** 레이어를 선택하고 레이어 삽입 버튼을 클릭합니다. 새 레이어의 이름을 **Actions**로 지정합니다.

4. **Actions** 레이어를 선택한 상태에서 프레임 1을 선택하고 F9 키를 눌러 **액션** 패널을 엽니다.

5. **액션** 패널에 다음 코드를 입력하여 재생하는 동안 응용 프로그램이 반복되지 않도록 하는 `stop()` 함수를 만듭니다.

```
stop();
```

6. **Actions** 레이어에서 프레임 1을 선택한 상태로 **액션** 패널에서 다음 코드를 추가하여 클래스를 가져옵니다.

```
// 필요한 클래스를 가져옵니다 .
import mx.services.Webservice;
import mx.controls.*;
import mx.containers.*;
import mx.controls.gridclasses.DataGridColumn;
// 사용자 정의 Cart 클래스를 가져옵니다 .
import Cart;
```

## 구성 요소 인스턴스의 데이터 유형 설정

이제 앞서 스테이지로 드래그한 각 구성 요소 인스턴스에 데이터 유형을 지정합니다.

ActionScript 2.0은 변수를 만들 때 데이터 유형을 지정함을 의미하는 엄격한 데이터 유형 지정을 사용합니다. 엄격한 데이터 유형 지정은 **액션** 패널에서 변수에 코드 힌트를 사용할 수 있게 합니다.

■ **액션** 패널에서 다음 코드를 추가하여 이미 만든 구성 요소 인스턴스 네 개의 데이터 유형을 지정합니다.

```
/* 스테이지에 있는 데이터 유형 인스턴스 , 다른 인스턴스도 Cart 클래스에서
런타임에 추가될 수 있습니다 .*/
var problems_cb:ComboBox;
var products_dg:DataGrid;
var cart_dg:DataGrid;
var products_xmlcon:mx.data.components.XMLConnector;
```

**예제 10**

여기서 지정하는 인스턴스 이름은 구성 요소를 스테이지로 드래그할 때 지정한 인스턴스 이름과 일치해야 합니다.

## 구성 요소 모양 사용자 정의

각 구성 요소에는 강조 색상, 글꼴, 글꼴 크기를 비롯하여 모양을 사용자 정의하는 데 사용되는 스타일 속성과 메서드가 있습니다. 개별적으로 구성 요소 인스턴스의 스타일을 설정할 수도 있고 전역적으로 스타일을 설정하여 응용 프로그램의 모든 구성 요소 인스턴스에 적용할 수도 있습니다. 이 자습 과정에서는 전역적으로 스타일을 설정합니다.

- 다음 코드를 추가하여 스타일을 설정합니다.

```
// problems_cb ComboBox 의 전역 스타일과 여유 수치를 정의합니다 .
_global.style.setStyle("themeColor", "haloBlue");
_global.style.setStyle("fontFamily", "Verdana");
_global.style.setStyle("fontSize", 10);
_global.style.setStyle("openEasing",
    mx.transitions.easing.Bounce.easeOut);
```

이 코드는 구성 요소의 테마 색상(선택한 항목에 대한 강조 색상), 글꼴 및 글꼴 크기를 설정하며 ComboBox 제목 막대를 클릭할 때 드롭 다운 목록이 나타나고 사라지는 방법을 지정하는 ComboBox 여유 값도 설정합니다.

## 콤보 상자에 잘못된 행동 표시

이 단원에서는 잘못된 행동(예: 감박 있고 화분에 물을 주지 않음 등) 목록이 포함된 웹 서비스로 연결하는 코드를 추가합니다. WSDL(Web Service Description Language) 파일은 [www.flash-mx.com/mm/firstapp/problems.cfc?WSDL](http://www.flash-mx.com/mm/firstapp/problems.cfc?WSDL)에 있습니다. WSDL이 어떻게 구성되어 있는지 보려면 WSDL 페이지로 이동하십시오.

ActionScript 코드는 표시할 웹 서비스 결과를 ComboBox 인스턴스에 전달합니다. 함수가 심각도에 따라 잘못된 행동을 정렬합니다. 서비스의 작동이 중지되었거나 함수가 없는 경우와 같이 웹 서비스에서 반환된 결과가 없으면 출력 패널에 오류 메시지가 나타납니다.

- 액션 패널에 다음 코드를 추가합니다.

```
/* 잘못된 행동을 검색하는 데 사용할 웹 서비스를 정의합니다 .
이 서비스는 problems_cb ComboBox 인스턴스에 바인딩됩니다 . */
var problemService:WebService = new WebService("http://www.flash-mx.com/
mm/firstapp/problems.cfc?WSDL");
var myProblems:Object = problemService.getProblems();

/* 웹 서비스에서 결과를 받으면 열 레이블을 위해 사용할 필드를 설정합니다 .
데이터 공급자를 웹 서비스에서 반환된 결과로 설정합니다 . */
myProblems.onResult = function(wsdlResults:Array) {
    problems_cb.labelField = "name";
    problems_cb.dataProvider = wsdlResults.sortOn("severity",
Array.NUMERIC);
};

/* 원격 웹 서비스에 연결할 수 없는 경우 출력 패널에
오류 메시지를 표시합니다 . */
```

```

myProblems.onFault = function(error:Object) {
    trace("error:");
    for (var prop in error) {
        trace(" "+prop+" -> "+error[prop]);
    }
};

```

**만  
민**

Ctrl+S를 눌러 작업을 저장한 다음 Ctrl+Enter를 누르거나 컨트롤 > 무비 테스트를 선택하여 응용 프로그램을 테스트합니다. 여기서 콤보 상자가 잘못된 행동 목록으로 채워지고 Gift Ideas용으로 만든 빈 데이터 격자가 체크 아웃 버튼과 함께 나타나야 합니다.

## Gift Ideas를 표시하도록 데이터 구성 요소 바인딩

자습 과정의 시작 부분에서 스테이지에 DataSet, XMLConnector 구성 요소를 추가했습니다. 또한 products\_xmlcon이라는 XMLConnector 인스턴스의 URL 속성을 응용 프로그램의 **Gift Ideas** 섹션에 들어갈 제품 정보가 포함된 XML 파일의 위치로 설정했습니다.

이제 응용 프로그램에서 XML 데이터를 사용할 수 있도록 Flash 제작 환경에서 데이터 바인딩 기능을 사용하여 XMLConnector, DataSet 및 DataSet 구성 요소를 함께 바인딩합니다.

구성 요소를 바인딩할 때 DataSet 구성 요소는 사용자가 What Did You Do? 섹션에서 선택하는 잘못된 행동의 심각도에 따라 XML 파일의 제품 목록을 필터링합니다. DataSet 구성 요소는 목록을 표시합니다.

## XML 데이터 소스 표시를 위해 스키마 사용

XMLConnector 구성 요소로 외부 XML 데이터 소스에 연결할 때는 XML 문서의 구조를 스키마 형식으로 표현하는 스키마를 지정해야 합니다. 스키마는 XMLConnector 구성 요소에 XML 데이터 소스를 읽는 방법을 알려 줍니다. 가장 쉽게 스키마를 지정하는 방법은 연결할 XML 파일의 복사본을 가져와서 스키마로 사용하는 것입니다.

1. 웹 브라우저를 열고 [www.flash-mx.com/mm/firstapp/products.xml](http://www.flash-mx.com/mm/firstapp/products.xml)(XMLConnector URL 매개 변수에 대해 설정한 위치)로 이동합니다.
2. 파일 > 다른 이름으로 저장을 선택합니다.
3. 작업 중인 FLA 파일과 같은 위치에 products.xml을 저장합니다.
4. 기본 타임라인에서 프레임 1을 선택합니다.
5. 스테이지 옆에 있는 products\_xmlcon(XMLConnector) 인스턴스를 선택합니다.
6. 구성 요소 관리자에서 스키마 탭을 클릭하고 스키마 탭 오른쪽, 스크롤 창 위에서 가져오기 버튼을 클릭합니다. 열기 대화 상자에서 3단계에서 가져온 products.xml 파일을 찾고 열기를 클릭합니다. 스키마 탭의 스크롤 창에 products.xml 파일의 스키마가 나타납니다.

스키마 탭의 위쪽 창에서 image 요소를 선택합니다. 아래쪽 창에서 data type을 선택하고 값 필드를 <empty>에서 String으로 변경합니다. description 요소에 대해 이 단계를 반복합니다.

## 잘못된 행동과 일치하도록 Gift Ideas 필터링

구성 요소 관리자의 바인딩 탭을 사용하여 XMLConnector, DataSet 및 DataGrid 구성 요소 인스턴스를 서로 바인딩합니다.

1. 스테이지에서 products\_xmlcon(XMLConnector) 인스턴스를 선택한 상태로 구성 요소 관리자에서 **바인딩** 탭을 클릭합니다.
2. **바인딩 추가** 버튼을 클릭합니다.
3. **바인딩 추가** 대화 상자에서 results.products.product array 항목을 선택하고 **확인**을 클릭합니다.
4. **바인딩** 탭의 바인딩 속성 창(속성 이름-값 쌍을 보여 주는 아래쪽 창)에서 **바인딩 대상** 항목을 클릭합니다.
5. **바인딩 대상** 항목의 값 열에서 돋보기 아이콘을 클릭하여 **바인딩 대상** 대화 상자를 엽니다.
6. **바인딩 대상** 대화 상자의 **구성 요소 경로** 창에서 DataSet <products\_ds> 인스턴스를 선택합니다. **스키마 위치** 창에서 dataProvider:array를 선택합니다. **확인**을 클릭합니다.
7. **바인딩** 탭의 바인딩 속성 창에서 **방향** 항목을 클릭합니다. 값 열의 팝업 메뉴에서 **밖으로**를 선택합니다.

이 옵션은 데이터가 양방향으로 전달되거나 DataSet 인스턴스에서 XMLConnector 인스턴스로 전달되는 것이 아니라 products\_xmlcon 인스턴스에서 products\_ds 인스턴스로 전달됨을 의미합니다.

8. 스테이지에서 products\_ds 인스턴스를 선택합니다. 구성 요소 관리자의 **바인딩** 탭에 있는 바인딩 목록(**바인딩** 탭의 위쪽 창)에 구성 요소의 데이터 공급자가 나타납니다. **바인딩 속성** 창에서 **바인딩 대상** 매개 변수는 products\_ds 인스턴스가 products\_xmlcon 인스턴스에 바인딩되어 있고 바인딩 방향이 **안으로**임을 나타냅니다.

다음은 데이터 세트에 의해 필터링된 데이터가 데이터 격자에 표시되도록 DataSet 인스턴스를 DataGrid 인스턴스에 바인딩하는 단계입니다.

9. products\_ds 인스턴스를 선택한 상태에서 **바인딩** 탭에서 **바인딩 추가** 버튼을 클릭합니다.
10. **바인딩 추가** 대화 상자에서 dataProvider: array 항목을 선택하고 **확인**을 클릭합니다.
11. **바인딩** 탭의 **바인딩** 목록에서 dataProvider: array 항목이 선택되어 있는지 확인합니다.
12. 바인딩 속성 창에서 **바인딩 대상** 항목을 클릭합니다.
13. **바인딩 대상** 항목의 값 열에서 돋보기 아이콘을 클릭하여 **바인딩 대상** 대화 상자를 엽니다.
14. **바인딩 대상** 대화 상자의 **구성 요소 경로** 창에서 products\_dg(DataGrid) 인스턴스를 선택합니다. **스키마 위치** 창에서 dataProvider:array를 선택합니다. **확인**을 클릭합니다.

## Gift Ideas 섹션에 열 추가

이제 응용 프로그램의 Gift Ideas 섹션에 있는 데이터 격자에 제품 정보와 가격을 표시하기 위한 열을 추가할 수 있습니다.

- Actions 레이어를 선택합니다. **액션** 패널에서 다음 코드를 추가하여 이름 열과 가격 열을 만들어 구성하고 DataGrid 인스턴스에 추가합니다.

```
// products_dg DataGrid 인스턴스에 데이터 격자 열과 해당 기본 폭을
// 정의합니다 .
var name_dgc:DataGridColumn = new DataGridColumn("name");
name_dgc.headerText = "Name";
name_dgc.width = 280;

// DataGrid 에 열을 추가합니다 .
products_dg.addColumn(name_dgc);
var price_dgc:DataGridColumn = new DataGridColumn("price");
price_dgc.headerText = "Price";
price_dgc.width = 100;

// 토탈입에 열의 테이블을 설정하는 데 사용될 함수를 정의합니다 .
price_dgc.labelFunction = function(item:Object) {
    if (item != undefined) {
        return "$"+item.price+ " "+item.priceQualifier;
    }
};
products_dg.addColumn(price_dgc);
```

## XML 커넥터 트리거

이제 XMLConnector 인스턴스가 원격 products.xml 파일의 내용을 로드, 파싱 및 바인딩하는 코드 행을 추가합니다. 이 파일은 앞서 만든 XMLConnector 인스턴스의 URL 속성에 대해 입력한 URL에 있습니다. 이 파일에는 응용 프로그램의 **Gift Ideas** 섹션에 나타날 제품 정보가 포함되어 있습니다.

- **액션** 패널에서 다음 코드를 추가합니다.

```
products_xmlcon.trigger();
```

## Gift Ideas를 필터링하는 이벤트 리스너 추가

이 단원에서는 사용자가 What Did You Do? 섹션(problems\_cb ComboBox 인스턴스)에서 잘못된 행동을 선택하면 이를 감지하는 이벤트 리스너를 추가합니다. 이 리스너에는 사용자가 선택한 잘못된 행동에 따라 Gift Ideas 목록을 필터링하는 함수가 포함되어 있습니다. 사소한 잘못된 행동을 선택하면 CD나 꽃과 같이 수수한 선물 목록이, 심각한 잘못된 행동을 선택하면 호화로운 선물이 표시됩니다.

이벤트 리스너를 사용한 작업에 대한 자세한 내용은 *Adobe Flash에서 ActionScript 2.0 학습의 “이벤트 리스너 사용”*을 참조하십시오.

### ■ 액션 패널에 다음 코드를 추가합니다.

```
/* problems_cb ComboBox 인스턴스에 대한 리스너를 정의합니다 .
리스너에서 DataSet( 및 DataGrid)에 있는 제품을 필터링합니다 .
필터링 기준은 ComboBox 에서 현재 선택된 항목의 심각도입니다 . */
var cbListener:Object = new Object();
cbListener.change = function(evt:Object) {
    products_ds.filtered = false;
    products_ds.filtered = true;
    products_ds.filterFunc = function(item:Object) {
        // 현재 항목의 심각도가 ComboBox 에서 선택한 항목보다 크거나 같으면
        // true 를 반환합니다 .
        return (item.severity>=evt.target.selectedItem.severity);
    };
};

// ComboBox 에 리스너를 추가합니다 .
problems_cb.addEventListener("change", cbListener);
```

change() 함수의 시작 부분에서 filtered 속성을 재설정(false로 설정했다가 다시 true로 설정)하면 사용자가 반복해서 What Did You Do? 선택 사항을 변경해도 함수가 제대로 작동하게 됩니다.

filterFunc() 함수는 선물 배열에서 지정한 항목이 사용자가 콤보 상자에서 선택한 심각도 범위 내에 있는지 확인합니다. 선물이 선택한 심각도 범위 내에 있으면 DataSet 인스턴스에 바인딩된 DataGrid 인스턴스에 표시됩니다.

코드의 마지막 행은 리스너를 problems\_cb ComboBox 인스턴스에 등록합니다.

## 장바구니 추가

이제 사용자 정의 Cart 클래스의 인스턴스를 만든 다음 이를 초기화하는 코드를 추가합니다.

- 액션 패널에 다음 코드를 추가합니다.

```
var myCart:Cart = new Cart(this);  
myCart.init();
```

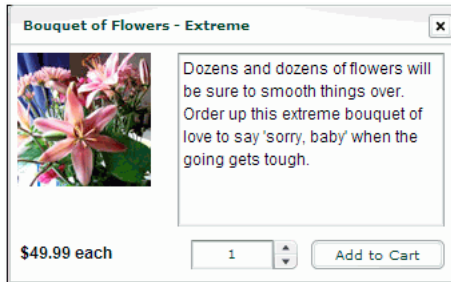
이 코드는 Cart 클래스의 init() 메서드를 사용하여 스테이지에 DataGrid 인스턴스를 추가하고 열을 정의하며 스테이지에 DataGrid 인스턴스를 배치합니다. 또한 Button 구성 요소 인스턴스를 추가하고 배치하며 버튼의 Alert 핸들러를 추가합니다. Cart 클래스 init() 메서드의 코드를 보려면 Cart.as 파일을 여십시오.

팁

Ctrl+S를 눌러 작업을 저장한 다음 Ctrl+Enter를 누르거나 컨트롤 -> 무비 테스트를 선택하여 응용 프로그램을 테스트합니다. 콤보 상자에서 잘못된 행동을 선택하면 선택한 잘못된 행동에 해당하는 선물의 일부가 Gift Ideas에 대해 만든 데이터 격자에 표시됩니다.

## 선물 세부 정보 표시

응용 프로그램에서 사용자가 Gift Ideas 섹션의 제품을 클릭하면 팝업 윈도우가 나타납니다. 이 팝업 윈도우에는 텍스트 설명, 이미지, 가격 등의 제품 정보를 표시하는 구성 요소 인스턴스가 포함되어 있습니다. 이 팝업 윈도우를 만들기 위해 무비 클립 심볼을 만들고 Loader, TextArea, Label, NumericStepper, Button 구성 요소의 인스턴스를 추가합니다. Bouquet of Flowers Extreme의 제품 정보 윈도우는 다음과 같습니다.



나중에 각 제품에 대해 이 무비 클립의 인스턴스를 동적으로 만드는 ActionScript를 추가합니다. 이 무비 클립의 인스턴스는 앞서 라이브러리에 추가한 Window 구성 요소에 표시됩니다. 구성 요소 인스턴스는 외부 XML 파일의 요소로 채워집니다.

1. 구성 요소 패널에서 Window 구성 요소의 인스턴스를 라이브러리로 드래그합니다.

이제 라이브러리에 Window 구성 요소 심볼이 추가되었습니다. ActionScript를 사용하여 Window 구성 요소의 인스턴스를 만드는 방법은 자습 과정의 뒷부분에서 설명합니다.

2. 라이브러리 패널(윈도우 > 라이브러리)에서 제목 막대 오른쪽의 옵션 메뉴를 클릭하고 새 심볼을 선택합니다.
3. 새 심볼 생성 대화 상자에서 이름 값으로 **ProductForm**을 입력하고 유형에서 무비 클립을 선택합니다.
4. 고급 버튼을 클릭합니다. 링크에서 **ActionScript에 내보내기**를 선택한 다음 첫 프레임으로 내보내기를 선택된 채로 두고 확인을 클릭합니다. 새 심볼의 문서 윈도우가 심볼 편집 모드로 열립니다.

라이브러리에는 있지만 스테이지에는 없는 무비 클립 심볼의 경우에는 **ActionScript에 내보내기**를 선택해야 ActionScript를 사용하여 채울 수 있습니다. 첫 프레임으로 내보내기를 선택하면 첫 번째 프레임을 로드하자마자 무비 클립을 사용할 수 있게 됩니다. 사용자가 Gift Ideas 섹션에 있는 제품을 클릭할 때마다 동적으로 무비 클립 인스턴스를 생성하는 ActionScript를 추가하는 방법은 자습 과정의 뒷부분에서 설명합니다.

5. 새 심볼의 타임라인에서 **레이어 1**을 선택하고 이름을 **Components**로 변경합니다.
6. Loader 구성 요소의 인스턴스를 구성 요소 패널에서 스테이지로 드래그합니다. *x, y* 좌표로 **5, 5**를 각각 입력합니다. 인스턴스 이름으로 **image\_ldr**을 입력합니다. 속성 관리자에서 매개 변수 탭을 클릭합니다. autoLoad에는 false를 선택하고 scaleContent에는 false를 선택합니다.

Loader 구성 요소 인스턴스는 제품 이미지를 표시하는 데 사용됩니다. autoLoad를 false로 설정하면 이미지가 자동으로 로드되지 않습니다. scaleContent를 false로 설정하면 이미지 크기가 조절되지 않습니다. 사용자가 Gift Ideas 섹션에서 선택한 제품에 따라 동적으로 이미지를 로드하는 코드를 추가하는 과정은 자습 과정의 뒷부분에서 설명합니다.

7. 구성 요소 패널에서 스테이지로 TextArea 구성 요소의 인스턴스를 드래그합니다. Loader 구성 요소 옆에 놓습니다. *x, y* 좌표로 **125, 5**를 각각 입력합니다. 인스턴스 이름으로 **description\_ta**를 입력합니다. 폭을 **200**으로, 높이를 **130**으로 설정합니다. 속성 관리자에서 매개 변수 탭을 클릭합니다. editable에는 false를, html에는 true를 선택합니다. wordWrap에는 true를 선택합니다.

TextArea 구성 요소 인스턴스는 선택한 제품의 텍스트 설명을 표시하는 데 사용됩니다. 선택한 설정은 사용자가 텍스트를 편집할 수 없고 HTML 태그로 텍스트 포맷을 지정할 수 있으며 텍스트 영역 크기에 맞추어 자동 줄 바꿈하도록 지정합니다.

8. 구성 요소 패널에서 스테이지로 Label 구성 요소의 인스턴스를 드래그합니다. Loader 구성 요소 아래에 놓습니다. *x, y* 좌표로 **5, 145**로 설정합니다. 인스턴스 이름으로 **price\_lbl**을 입력합니다. 속성 관리자에서 매개 변수 탭을 클릭합니다. autoSize에는 left를 선택하고 html에는 true를 선택합니다.

Label 구성 요소 인스턴스는 제품 가격과 가격 한정자("1개"나 "12개"와 같이 지정된 가격이 나타내는 제품 수량)를 표시합니다.

9. 구성 요소 패널에서 스테이지로 NumericStepper 구성 요소의 인스턴스를 드래그합니다. TextArea 구성 요소 아래에 놓습니다.  $x, y$  좌표를 **135, 145**로 설정합니다. 인스턴스 이름으로 **quantity\_ns**를 입력합니다. 속성 관리자에서 **매개 변수** 탭을 클릭합니다. **minimum**에 대해 **1**을 입력합니다.

**minimum**을 1로 설정하면 사용자가 적어도 하나의 제품을 선택해야 장바구니에 항목을 추가할 수 있습니다.

10. Button 구성 요소의 인스턴스를 구성 요소 패널에서 스테이지로 드래그합니다. NumericStepper 구성 요소 옆에 놓습니다.  $x, y$  좌표를 **225, 145**로 설정합니다. 인스턴스 이름으로 **addToCart\_button**을 입력합니다. 속성 관리자에서 **매개 변수** 탭을 클릭합니다. **label**에 **Add to Cart**를 입력합니다.

## 선물 세부 정보 표시를 트리거하는 이벤트 리스너 추가

이제 `products_dg` DataGrid 인스턴스에 이벤트 리스너를 추가하여 각 제품에 대한 정보를 표시합니다. 사용자가 **Gift Ideas** 섹션에서 제품을 클릭하면 팝업 윈도우가 나타나면서 제품 정보를 보여 줍니다.

■ 기본 타임라인의 액션 패널에서 다음 코드를 추가합니다.

```
// DataGrid의 행이 변경될 때 이를 감지하는 DataGrid 리스너를 만듭니다.
var dgListener:Object = new Object();
dgListener.change = function(evt:Object) {
    // DataGrid에서 현재 행이 변경될 때 제품 정보를 표시하는 새 팝업 윈도우를
    // 시작합니다.
    myWindow = mx.managers.PopUpManager.createPopUp(_root,
    mx.containers.Window, true, {title:evt.target.selectedItem.name,
    contentPath:"ProductForm", closeButton:true});
    // 팝업 윈도우의 크기를 설정합니다.
    myWindow.setSize(340, 210);
    // 사용자가 닫기 버튼을 클릭할 때 팝업 윈도우를 닫는 리스너를
    // 정의합니다.
    var closeListener:Object = new Object();
    closeListener.click = function(evt) {
        evt.target.deletePopUp();
    };
    myWindow.addEventListener("click", closeListener);
};
products_dg.addEventListener("change", dgListener);
```

이 코드는 `dgListener`라는 새 이벤트 리스너를 만들고 앞서 라이브러리에 추가한 `Window` 구성 요소의 인스턴스를 만듭니다. 새 윈도우의 제목은 제품 이름으로 설정됩니다. 윈도우의 내용 경로는 `ProductForm` 무비 클립으로 설정되고 윈도우의 크기는 340 x 210픽셀로 설정됩니다.

이 코드는 사용자가 정보를 본 후 윈도우를 닫는 데 사용되는 닫기 버튼도 추가합니다.

## ProductForm 무비 클립에 코드 추가

이제 방금 만든 ProductForm 무비 클립에 ActionScript를 추가합니다. ActionScript는 선택한 선물 정보로 무비 클립에 있는 구성 요소를 채우고, 선택한 제품을 장바구니에 추가하는 이벤트 리스너를 Add to Cart 버튼에 추가합니다.

이벤트 리스너를 사용한 작업에 대한 자세한 내용은 *Adobe Flash에서 ActionScript 2.0 학습의 “이벤트 리스너 사용”*을 참조하십시오.

1. ProductForm 무비 클립의 타임라인에서 새 레이어를 만들고 이름을 **Actions**으로 지정합니다. **Actions** 레이어의 첫 번째 프레임을 선택합니다.
2. 액션 패널에 다음 코드를 추가합니다.

```
// 객체를 만들어서 DataGrid 에서 선택한 제품 항목을 참조합니다 .
var thisProduct:Object = this._parent._parent.products_dg.selectedItem;
// 선택한 제품의 데이터로 description_ta TextArea 및 price_lbl Label
// 인스턴스를 채웁니다 .
description_ta.text = thisProduct.description;
price_lbl.text = "<b>${"+thisProduct.price+"
    "+thisProduct.priceQualifier+"</b>";
// 응용 프로그램 디렉토리에서 제품 이미지를 로드합니다 .
image_ldr.load(thisProduct.image);
```

**어**  
**이**

위 코드에는 용도를 설명하는 주석이 포함되어 있습니다. 사용자 자신이나 나중에 코드를 보는 다른 사람이 용도를 쉽게 이해할 수 있도록 사용자가 작성하는 모든 ActionScript 코드에 위와 같은 주석을 넣는 것이 좋습니다.

먼저 코드는 이후의 코드에서 선택한 제품을 참조하는 변수를 정의합니다. thisProduct 변수를 사용하면 this.\_parent.\_parent.products\_dg.selectedItem 경로를 사용하여 지정한 제품을 참조하지 않아도 됩니다.

그런 다음 코드는 thisProduct 객체의 description, price 및 priceQualifier 속성을 사용하여 TextArea와 Label 인스턴스를 채웁니다. 이들 속성은 자습 과정의 시작 부분에서 products\_xmlcon XMLConnector 인스턴스에 링크한 products.xml 파일의 요소에 해당됩니다. XMLConnector, DataSet 및 DataGrid 구성 요소 인스턴스를 함께 바인딩하는 방법은 자습 과정의 뒷부분에서 설명합니다. XML 파일의 요소는 다른 두 구성 요소 인스턴스를 채웁니다.

마지막으로 코드는 thisProduct 객체의 image 속성을 사용하여 제품 이미지를 Loader 구성 요소에 로드합니다.

- 이제 사용자가 Add to Cart 버튼을 클릭할 때 장바구니에 제품을 추가하는 이벤트 리스너를 추가합니다. 응용 프로그램의 기본 타임라인에 **ActionScript**를 추가하여 **Cart** 클래스 인스턴스를 만드는 방법은 자습 과정의 뒷부분에서 설명합니다. 다음 코드를 추가합니다.

```
var cartListener:Object = new Object();
cartListener.click = function(evt:Object) {
    var tempObj:Object = new Object();
    tempObj.quantity = evt.target._parent.quantity_ns.value;
    tempObj.id = thisProduct.id;
    tempObj.productObj = thisProduct;
    var theCart = evt.target._parent._parent._parent.myCart;
    theCart.addProduct(tempObj.quantity, thisProduct);
};
addToCart_button.addEventListener("click", cartListener);
```

- 구문 확인** 버튼(스크립트 창 위의 파랑 체크 표시)을 클릭하여 코드에 구문 오류가 있는지 확인합니다.

응용 프로그램에 코드를 추가할 때 구문을 자주 확인해야 합니다. 코드에 오류가 있으면 **출력** 패널에 나열됩니다. 구문 확인을 실행하면 현재 스크립트의 구문만 확인되고 FLA 파일에 있는 다른 스크립트의 구문은 확인되지 않습니다. 자세한 내용은 *Flash 사용 설명서*의 “스크립트 디버깅”을 참조하십시오.

- 문서 윈도우의 왼쪽 위에 있는 화살표 버튼을 클릭하거나 **보기 > 문서 편집**을 선택하여 심볼 편집 모드를 종료하고 기본 타임라인으로 돌아옵니다.

**만**

Ctrl+S를 눌러 작업을 저장한 다음 Ctrl+Enter를 누르거나 컨트롤 > 무비 테스트를 선택하여 응용 프로그램을 테스트합니다. 이때 선택한 선물을 클릭하면 설명, 가격 및 원하는 수량을 선택할 수 있는 숫자 스택퍼와 함께 선물 이미지가 표시되는 윈도우가 나타납니다.

# 체크 아웃 화면 만들기

사용자가 기본 화면에서 **체크 아웃** 버튼을 클릭하면 **체크 아웃** 화면이 나타납니다. **체크 아웃** 화면은 사용자가 결제, 배송 및 신용 카드 정보를 입력할 수 있는 양식을 제공합니다. 체크 아웃 화면은 다음과 같습니다.



체크 아웃 인터페이스는 응용 프로그램의 프레임 10에 있는 키프레임에 배치된 구성 요소로 구성됩니다. Accordion 구성 요소를 사용하여 체크 아웃 인터페이스를 만듭니다. Accordion 구성 요소는 한 번에 하나씩 표시되는 자식 시퀀스가 포함된 탐색기입니다. 또한 사용자가 기본 화면으로 돌아갈 수 있도록 Button 구성 요소 인스턴스를 추가하여 **뒤로** 버튼을 만듭니다.

Accordion 인스턴스의 자식으로 사용할 무비 클립을 만들어 Billing Information, Shipping Information 및 Credit Card Information 창을 표시하는 방법은 자습 과정의 뒷부분에서 설명합니다.

1. 응용 프로그램의 기본 타임라인에서 재생 헤드를 **체크 아웃**이라는 프레임 10으로 이동합니다. **Form** 레이어가 선택되어 있는지 확인합니다.
2. **Form** 레이어의 프레임 10에 빈 키프레임을 삽입합니다. 즉, 프레임을 선택하고 **삽입 > 타임라인 > 빈 키프레임**을 선택합니다.
3. 새 키프레임을 선택한 채 **구성 요소** 패널에서 스테이지로 Accordion 구성 요소의 인스턴스를 드래그합니다. 속성 관리자에 인스턴스 이름으로 **checkout\_acc**를 입력합니다. 폭을 **300**픽셀로, 높이를 **200**픽셀로 설정합니다.

4. 구성 요소 패널에서 스테이지의 오른쪽 아래 모서리로 Button 구성 요소의 인스턴스를 드래그합니다. 속성 관리자에 인스턴스 이름으로 **back\_button**을 입력합니다. **매개 변수** 탭을 클릭하고 label 속성의 값으로 **뒤로**를 입력합니다.

## Billing Information, Shipping Information 및 Credit Card Information 창

**Billing Information, Shipping Information** 및 **Credit Card Information** 창은 Accordion 구성 요소 인스턴스에 표시되는 무비 클립 인스턴스로 만듭니다. 각 창은 중첩된 두 무비 클립으로 구성됩니다.

부모 무비 클립에는 스크롤 가능 영역에 내용을 표시하는 데 사용되는 ScrollPane 구성 요소가 포함되어 있고, 자식 무비 클립에는 사용자가 이름, 주소 등의 개인 데이터를 입력할 수 있는 Label과 TextInput 구성 요소가 포함되어 있습니다. 사용자가 각 정보 필드를 스크롤할 수 있도록 ScrollPane 구성 요소를 사용하여 자식 무비 클립을 표시해야 합니다.

## Billing Information 창 만들기

먼저 **Billing Information** 양식 필드를 표시할 두 무비 클립, 즉 ScrollPane 구성 요소 인스턴스가 있는 부모 무비 클립과 Label 및 TextArea 구성 요소 인스턴스가 있는 자식 무비 클립을 만듭니다.

1. 라이브러리 패널(윈도우 > 라이브러리)에서 제목 막대 오른쪽의 옵션 메뉴를 클릭하고 새 심볼을 선택합니다.
2. 새 심볼 생성 대화 상자에서 이름 값으로 **checkout1\_mc**를 입력하고 유형에서 무비 클립을 선택합니다.
3. 고급 버튼을 클릭합니다. 링크에서 **ActionScript에 내보내기**를 선택한 다음 첫 프레임으로 내보내기를 선택된 채로 두고 확인을 클릭합니다.  
새 심볼의 문서 윈도우가 심볼 편집 모드로 열립니다.
4. ScrollPane 구성 요소의 인스턴스를 스테이지로 드래그합니다.
5. 속성 관리자에 인스턴스 이름으로 **checkout1\_sp**를 입력합니다. W 및 H 값을 **300, 135**로 설정합니다. x 및 y 좌표를 **0, 0**으로 설정합니다.
6. 매개 변수 탭을 클릭합니다. contentPath 속성을 **checkout1\_sub\_mc**로 설정합니다.  
checkout1\_sub\_mc 무비 클립은 스크롤 창 내부에 나타나고 Label과 TextInput 구성 요소를 포함합니다. 이제 이 무비 클립을 만듭니다.
7. 라이브러리 옵션 메뉴에서 새 심볼을 선택합니다.
8. 새 심볼 생성 대화 상자에서 이름 값으로 **checkout1\_sub\_mc**를 입력하고 유형에서 무비 클립을 선택합니다.

9. 고급 버튼을 클릭합니다. 링크에서 **ActionScript**에 **내보내기**를 선택한 다음 **첫 프레임으로 내보내기**를 선택된 채로 두고 **확인**을 클릭합니다.
- 새 심볼의 문서 윈도우가 심볼 편집 모드로 열립니다.
10. Label 구성 요소의 인스턴스 여섯 개를 스테이지로 드래그합니다. 또는 하나의 인스턴스를 스테이지로 드래그하고, 스테이지에서 Ctrl 키(Windows) 또는 Option 키(Macintosh)를 누른 상태에서 클릭하고 드래그하여 복사본을 만들 수 있습니다. 다음과 같이 인스턴스의 이름을 지정하고 배치합니다.
- 첫 번째 인스턴스에 대해 인스턴스 이름으로 **firstname\_lbl**을 입력하고  $x$  및  $y$  좌표를 **5, 5**로 설정합니다. **매개 변수** 탭을 클릭하고 **text**의 값으로 **이름**을 입력합니다.
  - 두 번째 인스턴스에 대해 인스턴스 이름으로 **lastname\_lbl**을 입력하고  $x$  및  $y$  좌표를 **5, 35**로 설정합니다. **매개 변수** 탭을 클릭하고 **text**의 값으로 **성**을 입력합니다.
  - 세 번째 인스턴스에 대해 인스턴스 이름으로 **country\_lbl**을 입력하고  $x$  및  $y$  좌표를 **5, 65**로 설정합니다. **매개 변수** 탭을 클릭하고 **text**의 값으로 **국가**를 입력합니다.
  - 네 번째 인스턴스에 대해 인스턴스 이름으로 **province\_lbl**을 입력하고  $x$  및  $y$  좌표를 **5, 95**로 설정합니다. **매개 변수** 탭을 클릭하고 **text**의 값으로 **시/도**를 입력합니다.
  - 다섯 번째 인스턴스에 대해 인스턴스 이름으로 **city\_lbl**을 입력하고  $x$  및  $y$  좌표를 **5, 125**로 설정합니다. **매개 변수** 탭을 클릭하고 **text**의 값으로 **구/군/시**를 입력합니다.
  - 여섯 번째 인스턴스에 대해 인스턴스 이름으로 **postal\_lbl**을 입력하고  $x$  및  $y$  좌표를 **5, 155**로 설정합니다. **매개 변수** 탭을 클릭하고 **text**의 값으로 **우편 번호**를 입력합니다.
11. TextInput 구성 요소의 인스턴스 여섯 개를 스테이지로 드래그합니다. 각 Label 인스턴스 바로 오른쪽에 TextInput 인스턴스를 놓습니다. 예를 들어 첫 번째 TextInput 인스턴스의  $x, y$  좌표는 **105, 5**여야 합니다. 다음과 같이 TextInput 인스턴스의 이름을 지정합니다.
- 첫 번째 인스턴스의 이름을 **billingFirstName\_ti**로 지정합니다.
  - 두 번째 인스턴스의 이름을 **billingLastName\_ti**로 지정합니다.
  - 세 번째 인스턴스의 이름을 **billingCountry\_ti**로 지정합니다.
  - 네 번째 인스턴스의 이름을 **billingProvince\_ti**로 지정합니다.
  - 다섯 번째 인스턴스의 이름을 **billingCity\_ti**로 지정합니다.
  - 여섯 번째 인스턴스의 이름을 **billingPostal\_ti**로 지정합니다.
- 창 가장자리에 너무 가까워 스크롤 창의 내용이 잘리는 경우가 있습니다. 다음은 Label과 TextInput 인스턴스가 제대로 표시되도록 checkout1\_sub\_mc 무비 클립에 흰색 사각형을 추가하는 단계입니다.
12. 타임라인에서 레이어 추가 버튼을 클릭합니다. 새 레이어를 기존 레이어 아래에 드래그합니다. 사각형이 있는 레이어가 아래에 있어야 사각형이 구성 요소 표시를 방해하지 않습니다.
13. 새 레이어의 **프레임 1**을 선택합니다.

14. 도구 패널에서 **사각형** 도구를 선택합니다. **회색상**은 **없음**으로, **채움 색상**은 흰색으로 설정합니다.  
 도구 패널에서 **회색상** 컨트롤을 클릭하고 **없음** 버튼(빨간선이 지나가는 흰색 견본)을 클릭합니다. **채움 색상** 컨트롤을 클릭하고 흰색 견본을 클릭합니다.
15. 드래그하여 Label과 TextInput 인스턴스의 아래쪽 및 오른쪽 모서리를 넘어가는 사각형을 만듭니다.

## Shipping Information 창 만들기

**Shipping Information** 창의 무비 클립은 Billing Information 창의 무비 클립과 비슷합니다. 사용자가 Billing Information 창에 입력한 것과 같은 데이터로 Shipping Information 양식 필드를 채울 수 있도록 하기 위해 CheckBox 구성 요소도 추가합니다.

1. 앞서 설명한 지침(38페이지의 “Billing Information 창 만들기” 참조)에 따라 Credit Card Information 창의 무비 클립을 만듭니다. 이름을 지정할 때 다음과 같은 차이점에 주의합니다.
  - 첫 번째 무비 클립의 경우 심볼 이름으로 **checkout2\_mc**를, 인스턴스 이름으로 **checkout2\_sp**를 입력합니다. 속성 관리자의 **메개 변수** 탭에서 **contentPath** 속성을 **checkout2\_sub\_mc**로 설정합니다.
  - 두 번째 무비 클립의 경우 심볼 이름으로 **checkout2\_sub\_mc**를 입력합니다.
  - TextInput 인스턴스의 경우 인스턴스 이름에서 “**billing**”을 “**shipping**”으로 변경합니다.
2. **checkout2\_sub\_mc** 무비 클립을 심볼 편집 모드로 연 상태에서 CheckBox 구성 요소의 인스턴스를 스테이지로 드래그하여 첫 번째 Label 인스턴스 바로 위에 놓습니다.  
 다른 구성 요소 인스턴스와 함께 이 인스턴스도 레이어 1에 배치해야 합니다.
3. 속성 관리자에 인스턴스 이름으로 **sameAsBilling\_ch**를 입력합니다.
4. **메개 변수** 탭을 클릭합니다. **label** 속성을 **결제 정보와 동일**로 설정합니다.

## Credit Card Information 창 만들기

**Credit Card Information** 창의 무비 클립도 **Billing Information** 및 **Shipping Information** 창의 무비 클립과 비슷합니다. 하지만 **Credit Card Information** 창의 중첩된 무비 클립에는 다른 두 창과 다소 다른 필드가 있는데 여기에는 신용 카드 번호와 다른 카드 데이터가 들어갑니다.

1. **Billing Information** 지침의 1-9단계(38페이지의 “**Billing Information** 창 만들기” 참조)에 따라 **Credit Card Information** 창의 무비 클립을 만듭니다. 이름을 지정할 때 다음과 같은 차이점에 주의합니다.
  - 첫 번째 무비 클립의 경우 심볼 이름으로 **checkout3\_mc**를, 인스턴스 이름으로 **checkout3\_sp**를 입력합니다. 속성 관리자의 **매개 변수** 탭에서 **contentPath** 속성을 **checkout3\_sub\_mc**로 설정합니다.
  - 두 번째 무비 클립의 경우 심볼 이름으로 **checkout3\_sub\_mc**를 입력합니다.
2. **Label** 구성 요소의 인스턴스 네 개를 스테이지로 드래그합니다. 다음과 같이 인스턴스의 이름을 지정하고 배치합니다.
  - 첫 번째 인스턴스에 대해 인스턴스 이름으로 **ccName\_lbl**을 입력하고  $x$  및  $y$  좌표를 **5, 5**로 설정합니다. **매개 변수** 탭을 클릭하고 **text**의 값으로 **신용 카드 소유자 이름**을 입력합니다.
  - 두 번째 인스턴스에 대해 인스턴스 이름으로 **ccType\_lbl**을 입력하고  $x$  및  $y$  좌표를 **5, 35**로 설정합니다. **매개 변수** 탭을 클릭하고 **text**의 값으로 **카드 종류**를 입력합니다.
  - 세 번째 인스턴스에 대해 인스턴스 이름으로 **ccNumber\_lbl**을 입력하고  $x$  및  $y$  좌표를 **5, 65**로 설정합니다. **매개 변수** 탭을 클릭하고 **text**의 값으로 **카드 번호**를 입력합니다.
  - 네 번째 인스턴스에 대해 인스턴스 이름으로 **ccExp\_lbl**을 입력하고  $x$  및  $y$  좌표를 **5, 95**로 설정합니다. **매개 변수** 탭을 클릭하고 **text**의 값으로 **만료일**을 입력합니다.
3. **TextInput** 구성 요소의 인스턴스를 스테이지로 드래그하여 **ccName\_lbl** 인스턴스 오른쪽에 놓습니다. 새 인스턴스의 이름을 **ccName\_ti**로 지정합니다.  $x$  및  $y$  좌표를 **105, 5**로 설정합니다. 폭을 **140**으로 설정합니다.
4. **ComboBox** 구성 요소의 인스턴스를 스테이지로 드래그하여 **ccType\_lbl** 인스턴스 오른쪽에 놓습니다. 새 인스턴스의 이름을 **ccType\_cb**로 지정합니다.  $x$  및  $y$  좌표를 **105, 35**로 설정합니다. 폭을 **140**으로 설정합니다.
5. **TextInput** 구성 요소의 다른 인스턴스를 스테이지로 드래그하여 **ccNumber\_lbl** 인스턴스 오른쪽에 놓습니다. 새 인스턴스의 이름을 **ccNumber\_ti**로 지정합니다.  $x$  및  $y$  좌표를 **105, 65**로 설정합니다. 폭을 **140**으로 설정합니다.

6. ComboBox 구성 요소의 인스턴스 두 개를 스테이지로 드래그하여 ccExp\_1b1 인스턴스 오른쪽에 하나를 놓고 그 오른쪽에 다른 하나를 놓습니다. 새 인스턴스의 이름을 **ccMonth\_cb**로 지정합니다. 폭을 **60**으로, *x* 및 *y* 좌표를 **105, 95**로 설정합니다. 두 번째 새 인스턴스 이름을 **ccYear\_cb**로 지정합니다. 폭을 **70**으로, *x* 및 *y* 좌표를 **175, 95**로 설정합니다.
7. Button 구성 요소의 인스턴스를 스테이지로 드래그하여 양식 아래쪽의 ccMonth\_cb 인스턴스 아래에 놓습니다. 새 인스턴스의 이름을 **checkout\_button**으로 지정합니다. *x* 및 *y* 좌표를 **125, 135**로 설정합니다. 속성 관리자의 **매개 변수** 탭에서 label 속성을 **체크 아웃**으로 설정합니다.
8. Billing Information 지침의 14-15단계(38페이지의 “Billing Information 창 만들기” 참조)에 따라 양식 아래쪽에 사각형을 추가합니다.

## 체크 아웃 버튼에 이벤트 리스너 추가

이제 사용자가 체크 아웃 버튼을 클릭할 때 체크 아웃 화면을 표시하는 코드를 추가합니다.

- 액션 패널에서 타임라인의 프레임 1에 추가한 ActionScript 뒤에 다음 코드를 추가합니다.

```
// 체크 아웃 버튼을 클릭하면 "체크 아웃" 프레임 레이블로 이동합니다.
var checkoutBtnListener:Object = new Object();
checkoutBtnListener.click = function(evt:Object) {
    evt.target._parent.gotoAndStop("checkout");
};
checkout_button.addEventListener("click", checkoutBtnListener);
```

이 코드는 사용자가 **체크 아웃** 버튼을 클릭할 때 재생 헤드가 타임라인에서 **체크 아웃** 레이블로 이동하도록 지정합니다.

## 체크 아웃 화면의 코드 추가

이제 기본 타임라인의 프레임 10에서 응용 프로그램의 **체크 아웃** 화면에 코드를 추가할 수 있습니다. 이 코드는 앞서 Accordion 구성 요소와 다른 구성 요소로 만든 **Billing Information**, **Shipping Information** 및 **Credit Card Information** 창에 사용자가 입력하는 데이터를 처리합니다.

1. 타임라인의 **Actions** 레이어에서 프레임 10을 선택하고 **삽입 > 타임라인 > 빈 키프레임**을 선택하여 빈 키프레임을 삽입합니다.
2. F9 키를 눌러 **액션** 패널을 엽니다.
3. 액션 패널에 다음 코드를 추가합니다.

```
stop();
import mx.containers.*;

// 스테이지에 Accordion 구성 요소를 정의합니다.
var checkout_acc:Accordion;
```

4. 이제 사용자의 결제 정보를 받아들이기 위해 **Accordion** 구성 요소 인스턴스에 첫 번째 자식을 추가합니다. 다음 코드를 추가합니다.

```
// Accordion 구성 요소의 자식을 정의합니다 .
var child1 = checkout_acc.createChild("checkout1_mc", "child1_mc",
  {label:"1. Billing Information"});
var thisChild1 = child1.checkout1_sp.spContentHolder;
```

첫 번째 행은 **Accordion** 구성 요소의 `createChild()` 메서드를 호출하고 인스턴스 이름이 `child1_mc`이고 레이블이 “1. Billing Information”인 앞서 만든 `checkout1_mc` 무비 클립 심볼의 인스턴스를 만듭니다. 코드의 두 번째 행은 포함된 **ScrollPane** 구성 요소 인스턴스의 단축키를 만듭니다.

5. 다음과 같이 **Accordion** 인스턴스의 두 번째 자식을 만들어 배송 정보를 받아들입니다.

```
/* Accordion 에 두 번째 자식을 추가합니다 .
sameAsBilling_ch CheckBox 에 대한 이벤트 리스너를 추가합니다 .
첫 번째 자식의 양식 값을 두 번째 자식에 복사합니다 . */
var child2 = checkout_acc.createChild("checkout2_mc", "child2_mc",
  {label:"2. Shipping Information"});
var thisChild2 = child2.checkout2_sp.spContentHolder;
var checkboxListener:Object = new Object();
checkboxListener.click = function(evt:Object) {
  if (evt.target.selected) {
    thisChild2.shippingFirstName_ti.text =
thisChild1.billingFirstName_ti.text;
    thisChild2.shippingLastName_ti.text =
thisChild1.billingLastName_ti.text;
    thisChild2.shippingCountry_ti.text =
thisChild1.billingCountry_ti.text;
    thisChild2.shippingProvince_ti.text =
thisChild1.billingProvince_ti.text;
    thisChild2.shippingCity_ti.text = thisChild1.billingCity_ti.text;
    thisChild2.shippingPostal_ti.text =
thisChild1.billingPostal_ti.text;
  }
};
thisChild2.sameAsBilling_ch.addEventListener("click", checkboxListener);
```

코드의 처음 두 행은 **Billing Information** 자식을 만드는 코드와 비슷합니다. 즉, 인스턴스 이름이 `child2_mc`이고 레이블이 “2. Shipping Information”인 `checkout2_mc` 무비 클립 심볼의 인스턴스를 만듭니다. 코드의 두 번째 행은 포함된 **ScrollPane** 구성 요소 인스턴스의 단축키를 만듭니다.

코드의 세 번째 행부터는 **CheckBox** 인스턴스에 이벤트 리스너를 추가합니다. 사용자가 체크 상자를 클릭하면 사용자가 **Billing Information** 창에 입력한 데이터가 배송 정보로 사용됩니다.

6. 이제 **Accordion** 인스턴스의 세 번째 자식을 만들어 신용 카드 정보를 받아들입니다.

```
// 세 번째 Accordion 자식을 정의합니다.
var child3 = checkout_acc.createChild("checkout3_mc", "child3_mc",
    {label:"3. Credit Card Information"});
var thisChild3 = child3.checkout3_sp.spContentHolder;
```

7. 다음 코드를 추가하여 신용 카드 만료일 및 종류에 대한 **ComboBox** 인스턴스를 만들고 각각을 정적으로 정의된 배열로 채웁니다.

```
/* 스테이지에 있는 세 개의 ComboBox 인스턴스에 값을 설정합니다.
ccMonth_cb, ccYear_cb and ccType_cb */
thisChild3.ccMonth_cb.labels = ["01", "02", "03", "04", "05", "06",
    "07", "08", "09", "10", "11", "12"];
thisChild3.ccYear_cb.labels = [2004, 2005, 2006, 2007, 2008, 2009,
    2010];
thisChild3.ccType_cb.labels = ["VISA", "MasterCard", "American Express",
    "Diners Club"];
```

8. 마지막으로 다음 코드를 추가하여 **체크 아웃** 버튼과 **뒤로** 버튼에 이벤트 리스너를 추가합니다. 사용자가 **체크 아웃** 버튼을 클릭하면 리스너 객체는 **Billing Information, Shipping Information** 및 **Credit Card Information** 창에서 서버에 보내지는 **LoadVars** 객체로 양식 필드를 복사합니다. **LoadVars** 클래스를 사용하면 객체의 모든 변수를 지정한 URL에 보낼 수 있습니다. 사용자가 **뒤로** 버튼을 클릭하면 응용 프로그램이 기본 화면으로 돌아갑니다.

```
/* checkout_button Button 인스턴스에 대한 리스너를 만듭니다.
이 리스너는 사용자가 Checkout 버튼을 클릭하면 모든 양식 변수를 서버에 보냅니다. */
var checkoutListener:Object = new Object();
checkoutListener.click = function(evt:Object){
    evt.target.enabled = false;
    /* 원격 서버에 변수를 보내고 결과를 받는 두 개의 LoadVars 객체 인스턴스를
    만듭니다. */
    var response_lv:LoadVars = new LoadVars();
    var checkout_lv:LoadVars = new LoadVars();
    checkout_lv.billingFirstName = thisChild1.billingFirstName_ti.text;
    checkout_lv.billingLastName = thisChild1.billingLastName_ti.text;
    checkout_lv.billingCountry = thisChild1.billingCountry_ti.text;
    checkout_lv.billingProvince = thisChild1.billingProvince_ti.text;
    checkout_lv.billingCity = thisChild1.billingCity_ti.text;
    checkout_lv.billingPostal = thisChild1.billingPostal_ti.text;
    checkout_lv.shippingFirstName = thisChild2.shippingFirstName_ti.text;
    checkout_lv.shippingLastName = thisChild2.shippingLastName_ti.text;
    checkout_lv.shippingCountry = thisChild2.shippingCountry_ti.text;
    checkout_lv.shippingProvince = thisChild2.shippingProvince_ti.text;
    checkout_lv.shippingCity = thisChild2.shippingCity_ti.text;
    checkout_lv.shippingPostal = thisChild2.shippingPostal_ti.text;
    checkout_lv.ccName = thisChild3.ccName_ti.text;
    checkout_lv.ccType = thisChild3.ccType_cb.selectedItem;
    checkout_lv.ccNumber = thisChild3.ccNumber_ti.text;
    checkout_lv.ccMonth = thisChild3.ccMonth_cb.selectedItem;
```

```

checkout_lv.ccYear = thisChild3.ccYear_cb.selectedItem;

/* 변수를 checkout_lv LoadVars 에서 서버의 원격 스크립트로 보냅니다.
결과를 response_lv 인스턴스에 저장합니다. */
checkout_lv.sendAndLoad("http://www.flash-mx.com/mm/firstapp/
cart.cfm", response_lv, "POST");
response_lv.onLoad = function(success:Boolean) {
    evt.target.enabled = true;
};
};
thisChild3.checkout_button.addEventListener("click", checkoutListener);
cart_mc._visible = false;
var backListener:Object = new Object();
backListener.click = function(evt:Object) {
    evt.target._parent.gotoAndStop("home");
}
back_button.addEventListener("click", backListener);

```

## 응용 프로그램 테스트

축하합니다! 지금까지 응용 프로그램을 만드는 방법을 배웠습니다. Ctrl+S를 눌러 작업을 저장한 다음 Ctrl+Enter를 누르거나 **컨트롤 > 무비 테스트**를 선택하여 응용 프로그램을 테스트합니다.

## 완성된 응용 프로그램 보기

자습 과정을 성공적으로 완료하지 못한 경우라도 완성된 응용 프로그램의 실행 버전을 볼 수 있습니다. 샘플 파일은 Flash 샘플 페이지([www.adobe.com/go/learn\\_fl\\_samples\\_kr](http://www.adobe.com/go/learn_fl_samples_kr))를 참조하십시오. 다음과 같은 샘플을 사용할 수 있습니다.

- 시작 Flash(FLA) 파일, first\_app\_start fla
- 완성된 파일, first\_app fla

응용 프로그램의 FLA 파일을 보려면 components\_application 폴더에서 first\_app fla 파일을 엽니다.

이 파일을 자신이 만든 파일과 비교하면 오류를 찾는 데 도움이 될 것입니다.

응용 프로그램을 만드는 데 사용된 그래픽 파일 및 다른 예셋과 함께 응용 프로그램에 사용된 모든 구성 요소가 라이브러리에 나타납니다. 일부 구성 요소는 스테이지에 인스턴스로 나타나고, 일부는 ActionScript 코드에서 참조되고 런타임 전에는 나타나지 않습니다.



이 장에서는 여러 Adobe Flash(FLA) 파일과 ActionScript 클래스 파일을 사용하여 문서에 구성 요소를 추가하고 속성을 설정하는 방법을 배웁니다. 또한 코드 힌트 사용, 사용자 정의 포커스 탐색 기능 만들기, 구성 요소 심도 관리, Adobe Component Architecture 버전 2로 버전 1 구성 요소 업그레이드 등의 몇 가지 고급 항목에 대해서도 설명합니다.

이 장에 사용된 샘플은 Flash 샘플 페이지([www.adobe.com/go/learn\\_fl\\_samples\\_kr](http://www.adobe.com/go/learn_fl_samples_kr))를 참조하십시오. 다음과 같은 샘플을 사용할 수 있습니다.

- TipCalculator.fla
- TipCalculator.swf

이 장에서 설명하는 항목은 다음과 같습니다.

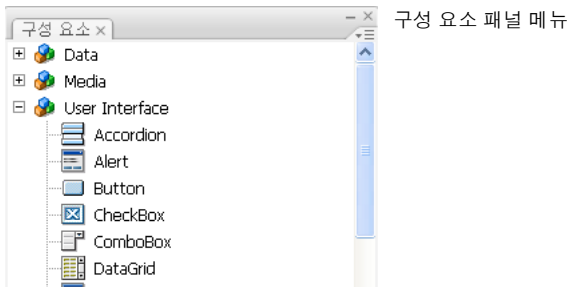
구성 요소 패널.....	48
Flash 문서에 구성 요소 추가 .....	48
라이브러리 패널의 구성 요소 .....	52
구성 요소 매개 변수 설정.....	52
구성 요소 크기 조절.....	54
Flash 문서에서 구성 요소 삭제 .....	55
코드 힌트 사용.....	55
스크린에서 ActionScript 사용 .....	56
사용자 정의 포커스 탐색 기능 만들기 .....	58
문서 내에서 구성 요소 심도 관리.....	59
실시간 미리 보기의 구성 요소.....	60
구성 요소에 프리로더 사용 .....	61
구성 요소 로드.....	62
버전 1 구성 요소를 버전 2 아키텍처로 업그레이드.....	63

# 구성 요소 패널

사용자 수준 구성/Components 디렉토리에 있는 모든 구성 요소가 **구성 요소** 패널에 표시됩니다. 이 디렉토리에 대한 자세한 내용은 14페이지의 “구성 요소 파일이 저장되는 위치”를 참조하십시오.

## 구성 요소 패널을 표시하려면:

- 윈도우 > 구성 요소를 선택합니다.



## Flash를 시작한 후에 설치된 구성 요소를 표시하려면:

1. ActionScript 2.0에 맞는 제작 옵션을 설정합니다.  
파일 > 제작 설정 > 플래시 탭을 선택하고 ActionScript 드롭 다운 메뉴에서 **ActionScript 2.0**을 선택합니다.
2. 윈도우 > 구성 요소를 선택합니다.
3. 구성 요소 패널 팝업 메뉴에서 **다시 로드**를 선택합니다.

# Flash 문서에 구성 요소 추가

구성 요소 패널에서 스테이지로 구성 요소를 드래그하면 컴파일된 클립(SWC) 심볼이 **라이브러리** 패널에 추가됩니다. SWC 심볼이 라이브러리에 추가되면 여러 인스턴스를 스테이지로 드래그할 수 있습니다. `UIObject.createClassObject()` ActionScript 메서드를 사용하여 런타임에 구성 요소를 문서에 추가할 수도 있습니다. 자세한 내용은 *ActionScript 2.0 구성 요소 언어 참조 설명서*를 참조하십시오.

**예외** Menu 및 Alert 구성 요소만이 예외인데, `UIObject.createClassObject()`를 사용하여 인스턴스화할 수 없습니다. 이들 구성 요소는 대신 `show()` 메서드를 사용합니다.

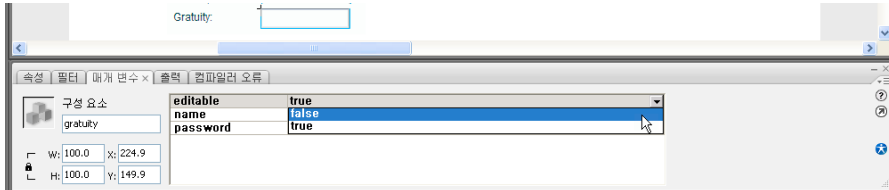
## 제작하는 동안 구성 요소 추가

구성 요소 패널을 사용하여 문서에 구성 요소를 추가한 후 라이브러리 패널에서 스테이지로 구성 요소를 드래그하여 문서에 구성 요소의 추가 인스턴스를 추가할 수 있습니다. 그런 다음 속성 관리자의 매개 변수 탭이나 구성 요소 관리자의 매개 변수 탭에서 추가 인스턴스에 대한 속성을 설정할 수 있습니다.

### 구성 요소 패널을 사용하여 Flash 문서에 구성 요소를 추가하려면:

1. ActionScript 2.0에 맞는 제작 옵션을 설정합니다.  
파일 > 제작 설정 > 플래시 탭을 선택하고 ActionScript 드롭 다운 메뉴에서 **ActionScript 2.0**을 선택합니다.
2. 윈도우 > 구성 요소를 선택합니다.
3. 다음 중 하나를 수행합니다.
  - 구성 요소 패널에서 스테이지로 구성 요소를 드래그합니다.
  - 구성 요소 패널에서 구성 요소를 두 번 클릭합니다.
4. 구성 요소가 FLA 파일(설치되어 있는 모든 버전 2 구성 요소는 SWC 파일임)이고 동일한 구성 요소의 다른 인스턴스에 대한 스킨 또는 추가하려는 구성 요소와 스킨을 공유하는 구성 요소의 스킨을 편집한 경우 다음 중 하나를 수행합니다.
  - 편집한 스킨을 유지하고 이 스킨을 새 구성 요소에 적용하려면 **기존 항목 교체 안함**을 선택합니다.
  - 모든 스킨을 기본 스킨으로 교체하려면 **기존 항목 교체**를 선택합니다. 새 구성 요소와 모든 이전 버전의 구성 요소나 자신의 스킨을 공유하는 구성 요소는 기본 스킨을 사용하게 됩니다.
5. 스테이지에서 구성 요소를 선택합니다.
6. 윈도우 > 속성 > 속성을 선택합니다.
7. 속성 관리자에서 구성 요소 인스턴스의 이름을 입력합니다.
8. 매개 변수 탭을 클릭한 다음 인스턴스에 대한 매개 변수를 지정합니다.

다음 그림에서는 TipCalculator.fla 샘플 파일에 있는 TextInput 구성 요소에 대한 속성 관리자를 보여 줍니다. TipCalculator.fla 샘플 파일에 액세스하려면 Flash 샘플 페이지 ([www.adobe.com/go/learn\\_fl\\_samples\\_kr](http://www.adobe.com/go/learn_fl_samples_kr))를 참조하십시오.



자세한 내용은 52페이지의 “구성 요소 매개 변수 설정”을 참조하십시오.

9. 폭 및 높이 값을 편집하여 구성 요소의 크기를 원하는 대로 변경합니다.  
특정 유형의 구성 요소 크기를 조절하는 방법에 대한 자세한 내용은 *ActionScript 2.0 구성 요소 언어 참조 설명서*에서 해당 구성 요소 항목을 참조하십시오.
10. 구성 요소의 색상과 텍스트 서식을 변경하려면 다음 중 하나 이상을 수행합니다.
  - 모든 구성 요소에 사용할 수 있는 `setStyle()` 메서드를 사용하여 구성 요소 인스턴스의 특정 스타일 속성 값을 설정하거나 변경합니다. 자세한 내용은 *ActionScript 2.0 구성 요소 언어 참조 설명서*의 “`UIObject.setStyle()`”를 참조하십시오.
  - 모든 버전 2 구성 요소에 할당된 전역 스타일 선언에서 여러 속성을 편집합니다.
  - 특정 구성 요소 인스턴스의 사용자 정의 스타일 선언을 만듭니다.  
자세한 내용은 80페이지의 “스타일을 사용하여 구성 요소 색상 및 텍스트 사용자 정의”를 참조하십시오.
11. 구성 요소 모양을 사용자 정의하려면 다음 중 하나를 수행합니다.
  - 테마를 적용합니다(104페이지의 “테마” 참조).
  - 구성 요소 스킨을 편집합니다(93페이지의 “구성 요소 스킨링” 참조).

## 런타임에 ActionScript로 구성 요소 추가

이 단원의 내용을 이해하려면 ActionScript에 대한 중급 또는 고급 지식이 필요합니다.

대부분의 구성 요소가 `UIObject` 클래스에서 상속하는 `createClassObject()` 메서드를 사용하여 동적으로 Flash 응용 프로그램에 구성 요소를 추가합니다. 예를 들어, 웹 포털의 홈 페이지와 같이 사용자가 구성된 환경 설정을 기반으로 페이지 레이아웃을 만드는 구성 요소를 추가할 수 있습니다.

Flash와 함께 설치되는 버전 2 구성 요소는 패키지 디렉토리에 있습니다. 자세한 내용은 *Adobe Flash에서 ActionScript 2.0 학습*의 “패키지”를 참조하십시오. 제작하는 동안 스테이지에 구성 요소를 추가한 경우 해당 인스턴스 이름(예: `myButton`)을 사용하여 구성 요소를 참조할 수 있습니다. 하지만 런타임에 ActionScript로 응용 프로그램에 구성 요소를 추가한 경우에는 정식 클래스 이름(예: `mx.controls.Button`)을 지정하거나 `import` 문을 사용하여 패키지를 가져와야 합니다.

예를 들어, `Alert` 구성 요소를 참조하는 ActionScript 코드를 작성하기 위해 다음과 같이 `import` 문을 사용하여 클래스를 참조할 수 있습니다.

```
import mx.controls.Alert;
Alert.show("The connection has failed", "Error");
```

또는 다음과 같이 전체 패키지 경로를 사용할 수 있습니다.

```
mx.controls.Alert.show("The connection has failed", "Error");
```

자세한 내용은 *Adobe Flash*에서 *ActionScript 2.0* 학습의 “클래스 파일 가져오기”를 참조하십시오.

ActionScript 메시지를 사용하면 동적으로 추가한 구성 요소에 대해 추가 매개 변수를 설정할 수 있습니다. 자세한 내용은 *ActionScript 2.0 구성 요소 언어 참조 설명서*를 참조하십시오.

<b>정보</b>	런타임에 구성 요소를 문서에 추가하려면 SWF 파일을 컴파일할 때 라이브러리에 해당 구성 요소가 있어야 합니다. 라이브러리에 구성 요소를 추가하려면 구성 요소 패널에 있는 구성 요소 아이콘을 라이브러리로 드래그합니다. 또한 ActionScript를 사용하여 동적으로 인스턴스화된 구성 요소가 포함된 무비 클립을 다른 무비 클립으로 로드하는 경우, SWF 파일을 컴파일할 때 부모 무비 클립의 라이브러리에 해당 구성 요소가 있어야 합니다.
-----------	--

### ActionScript를 사용하여 Flash 문서에 구성 요소를 추가하려면:

1. 구성 요소 패널에서 현재 문서의 라이브러리로 구성 요소를 드래그합니다.

<b>정보</b>	구성 요소는 기본적으로 첫 프레임으로 내보내도록 설정됩니다(마우스 오른쪽 버튼으로 클릭(Windows)하거나 Control 키를 누른 상태에서 클릭(Macintosh)한 다음 링크 메뉴 옵션을 선택하여 첫 프레임으로 내보내기 설정을 확인). 구성 요소가 포함된 응용 프로그램에 대한 프리로더를 사용하려면 내보내기 프레임을 변경하고 61페이지의 “구성 요소에 프리로더 사용”에서 자세한 내용을 참조하십시오.
-----------	--

2. 타임라인에서 구성 요소를 추가할 프레임을 선택합니다.
3. 아직 열려 있지 않은 경우 **액션** 패널을 엽니다.
4. `createClassObject()`를 호출하여 런타임에 구성 요소 인스턴스를 만듭니다.  
이 메시지는 단독으로 호출하거나 모든 구성 요소 인스턴스에서 호출할 수 있습니다. `createClassObject()` 메시지는 구성 요소 클래스 이름, 새 인스턴스의 인스턴스 이름, 심도, 런타임에 속성을 설정하는 데 사용할 수 있는 선택적 초기화 객체 등의 매개 변수를 사용합니다.  
다음 예와 같이 클래스 이름 매개 변수에서 클래스 패키지를 지정할 수 있습니다.  

```
createClassObject(mx.controls.CheckBox, "cb", 5, {label:"Check Me"});
```

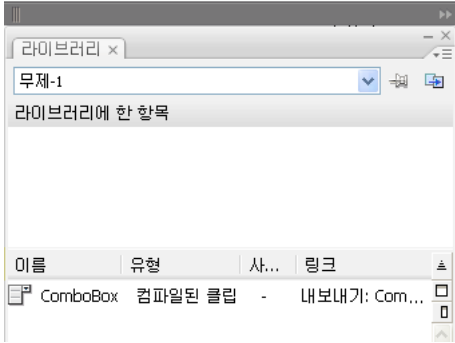
  
또는 다음 예와 같이 클래스 패키지를 가져올 수 있습니다.  

```
import mx.controls.CheckBox;  
createClassObject(CheckBox, "cb", 5, {label:"Check Me"});
```

  
자세한 내용은 65페이지의 제4장, “구성 요소 이벤트 처리” 및 *ActionScript 2.0 구성 요소 언어 참조 설명서*의 “UIObject.createClassObject()”를 참조하십시오.

# 라이브러리 패널의 구성 요소

문서에 구성 요소를 추가하면 라이브러리 패널에 해당 구성 요소가 컴파일된 클립(SWC 파일) 심볼로 표시됩니다.



라이브러리 패널의 ComboBox 구성 요소

구성 요소 아이콘을 라이브러리에서 스테이지로 드래그하여 구성 요소 인스턴스를 추가할 수 있습니다.

컴파일된 클립에 대한 자세한 내용은 [19페이지](#)의 “컴파일된 클립 및 SWC 파일”을 참조하십시오.

## 구성 요소 매개 변수 설정

각 구성 요소에는 모양과 비헤이비어를 변경하는 데 사용할 수 있는 매개 변수가 있습니다. 매개 변수는 속성 관리자 와 구성 요소 관리자 에 표시되는 속성입니다. 가장 일반적으로 사용되는 속성은 제작 매개 변수로 표시되지만 그 외 다른 속성은 `ActionScript`를 사용하여 설정해야 합니다. 제작하는 동안에 설정할 수 있는 모든 매개 변수를 `ActionScript`로 설정할 수도 있습니다. `ActionScript`로 매개 변수를 설정하면 제작하는 동안에 설정된 모든 값이 재정의됩니다.

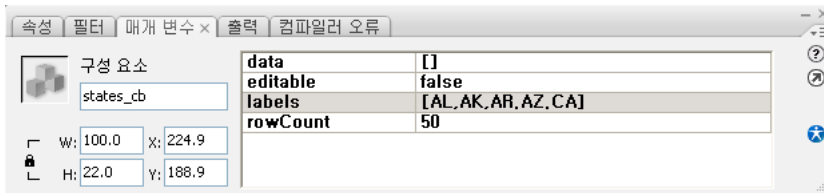
모든 버전 2 사용자 인터페이스(UI) 구성 요소는 `UIObject` 및 `UIComponent` 클래스의 속성과 메서드를 상속합니다. “`UIObject.setSize()`”, “`UIObject.setStyle()`”, “`UIObject.x`” 및 “`UIObject.y`” 등은 모든 구성 요소에서 사용하는 속성과 메서드입니다. 각 구성 요소에는 고유한 속성과 메서드도 들어 있으며, 그 중 일부는 제작 매개 변수로 사용됩니다. 예를 들어, `ProgressBar` 구성 요소에는 `percentComplete` 속성(“`ProgressBar.percentComplete`”)이 있고 `NumericStepper` 구성 요소에는 `nextValue` 및 `previousValue` 속성(“`NumericStepper.nextValue`”, “`NumericStepper.previousValue`”)이 있습니다.

사용하는 패널에 관계없이 구성 요소 관리자나 속성 관리자를 사용하여 구성 요소 인스턴스의 매개 변수를 설정할 수 있습니다.

### 속성 관리자에 구성 요소의 인스턴스 이름을 입력하려면:

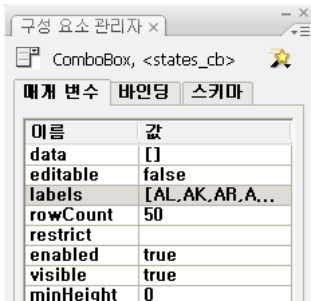
1. 윈도우 > 속성 > 속성을 선택합니다.
2. 스테이지에서 구성 요소 인스턴스를 선택합니다.
3. 구성 요소라는 단어 아래의 텍스트 상자에 인스턴스 이름을 입력합니다.

인스턴스 이름에 구성 요소의 종류를 나타내는 접미어를 추가하는 것이 좋습니다. 그러면 ActionScript 코드를 보다 쉽게 읽을 수 있습니다. 이 예에서 구성 요소가 각 지역을 나열하는 콤보 상자이기 때문에 인스턴스 이름은 **states\_cb**입니다.



### 구성 요소 관리자에 구성 요소 인스턴스의 매개 변수를 입력하려면:

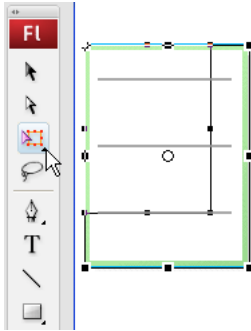
1. 윈도우 > 구성 요소 관리자를 선택합니다.
2. 스테이지에서 구성 요소 인스턴스를 선택합니다.
3. 매개 변수를 입력하려면 매개 변수 탭을 클릭합니다.



4. 구성 요소의 바인딩 또는 스키마를 입력하거나 보려면 해당 탭을 클릭합니다.

# 구성 요소 크기 조절

자유 변형 도구나 `setSize()` 메서드를 사용하면 구성 요소 인스턴스의 크기를 조절할 수 있습니다.



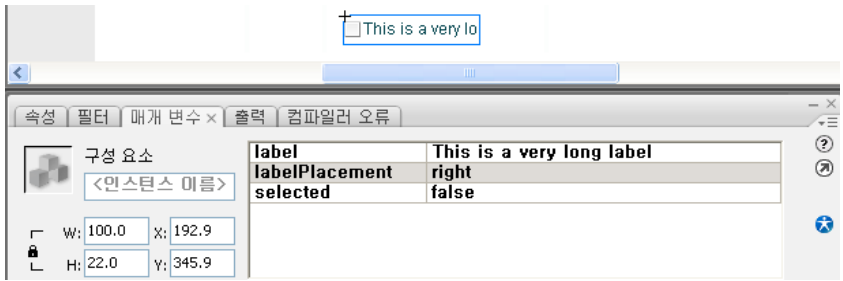
자유 변형 도구로 스테이지의 Menu 구성 요소 크기 조절

모든 구성 요소 인스턴스에서 `setSize()` 메서드를 호출하여 크기를 조절할 수 있습니다 (*ActionScript 2.0 구성 요소 언어 참조 설명서*의 “`UIObject.setSize()`” 참조). 다음 코드는 `TextArea` 구성 요소 크기를 200 x 300 픽셀로 조절합니다.

```
myTextArea.setSize(200, 300);
```

**예외** ActionScript의 `_width` 및 `_height` 속성을 사용하여 구성 요소의 폭과 높이를 조정하면 구성 요소 크기는 조절되지만 구성 요소 내용의 레이아웃은 동일하게 유지됩니다. 따라서 무비를 재생할 때 구성 요소가 제대로 표시되지 않을 수도 있습니다.

구성 요소는 레이블에 맞게 자동으로 크기가 조절되지 않습니다. 문서에 추가한 구성 요소 인스턴스가 작아서 레이블을 표시할 수 없으면 레이블 텍스트가 잘립니다. 사용자가 레이블에 맞게 구성 요소 크기를 조절해야 합니다.



CheckBox 구성 요소의 잘린 레이블

구성 요소 크기 조절에 대한 자세한 내용은 *ActionScript 2.0 구성 요소 언어 참조 설명서*에서 해당 구성 요소 항목을 참조하십시오.

## Flash 문서에서 구성 요소 삭제

Flash 문서에서 구성 요소 인스턴스를 삭제하려면 컴파일된 클립 아이콘을 삭제하여 라이브러리에서 구성 요소를 삭제해야 합니다. 스테이지에서 구성 요소를 삭제하는 것만으로는 부족합니다.

### 문서에서 구성 요소를 삭제하려면:

1. 라이브러리 패널에서 컴파일된 클립 심볼(SWC)을 선택합니다.
2. 라이브러리 패널의 아래쪽에 있는 **삭제** 버튼을 클릭하거나 **라이브러리 옵션** 메뉴에서 **삭제**를 선택합니다.
3. 삭제 대화 상자에서 **삭제**를 클릭하여 선택 항목을 삭제합니다.

## 코드 힌트 사용

ActionScript 2.0을 사용할 경우 구성 요소 클래스를 포함한 내장 클래스에 기반한 변수에 대해 고정 유형을 사용할 수 있습니다. 이렇게 변수를 입력하면 ActionScript 편집기는 해당 변수에 대한 코드 힌트를 표시합니다. 예를 들어, 다음을 입력한다고 가정합니다.

```
import mx.controls.CheckBox;  
var myCheckBox:CheckBox;  
myCheckBox.
```

변수를 `CheckBox` 유형으로 지정했으므로 `myCheckBox` 뒤에 마침표를 입력하자마자 `CheckBox` 구성 요소에 사용할 수 있는 메서드와 속성 목록이 표시됩니다. 자세한 내용은 *Adobe Flash*에서 *ActionScript 2.0 학습*의 “데이터 유형 및 고정 데이터 유형 지정” 및 *Flash 사용 설명서*의 “코드 힌트를 사용하려면”을 참조하십시오.

# 스크린에서 ActionScript 사용

스크린, 슬라이드 및 양식은 Flash CS3에서 제공되지 않습니다. Flash 8에서 제작된 기존 프로젝트 파일을 연 경우에는 이러한 기능을 사용할 수 있지만, Flash CS3에서는 ActionScript 2.0과 ActionScript 3.0 모두 스크린, 슬라이드 또는 양식 응용 프로그램을 새로 만들 수 없습니다.

문서에서 스크린을 제어하려면 ActionScript를 사용하여 스크린의 삽입, 제거, 이름 변경, 순서 변경 및 기타 작업을 수행합니다.

ActionScript에서는 스크린을 제어할 때 스크린의 인스턴스 이름, 클래스 이름 및 등록 포인트를 사용합니다. 자세한 내용은 56페이지의 “스크린 인스턴스 이름, 클래스 이름 및 등록 포인트”를 참조하십시오. 또한 ActionScript는 스크린 매개 변수를 사용합니다. 자세한 내용은 *Flash 사용 설명서*의 “스크린 매개 변수 설정”을 참조하십시오.

<b>예제</b>	스크린과 무비 클립이 ActionScript와 상호 작용하는 방식은 서로 비슷하지만 몇 가지 중요한 차이점이 있습니다. 자세한 내용은 57페이지의 “스크린과 ActionScript의 상호 작용 방식”을 참조하십시오.
-----------	--

자세한 내용은 *ActionScript 2.0 구성 요소 언어 참조 설명서*의 “Screen 클래스”, “Form 클래스” 및 “Slide 클래스”를 참조하십시오.

## 스크린 인스턴스 이름, 클래스 이름 및 등록 포인트

스크린 이름에 따라 해당 스크린의 인스턴스 이름과 클래스 이름이 자동으로 생성됩니다. 여러 가지 방식으로 ActionScript에서 스크린을 다룰 때 이런 식별 레이블이 필요합니다. 스크린의 동작 방식을 조정하려면 스크린의 등록 포인트를 변경합니다. 다음과 같은 방법으로 기능을 사용할 수 있습니다.

- 인스턴스 이름은 스크린에 지정된 고유 이름으로, ActionScript에서 스크린을 대상으로 할 때 사용됩니다. 속성 관리자에서 인스턴스 이름을 변경합니다. 인스턴스 이름은 [스크린 요약] 창의 스크린 이름 및 스크린의 링크 식별자와 같습니다. 인스턴스 이름을 업데이트하면 스크린 이름 및 링크 식별자도 업데이트됩니다. 자세한 내용은 *Flash 사용 설명서*의 “스크린 속성 및 매개 변수 설정”을 참조하십시오.

<b>예제</b>	무비 클립, 버튼 및 그래픽을 비롯한 심볼 인스턴스에도 인스턴스 이름이 있습니다. 자세한 내용은 <i>Flash 사용 설명서</i> 의 “심볼, 인스턴스 및 라이브러리 에셋 사용”을 참조하십시오.
-----------	---

- 클래스 이름은 스크린이 지정된 ActionScript 클래스를 나타냅니다. 기본적으로 슬라이드 스크린은 mx.screens.Slide 클래스에 지정되고 양식 스크린은 mx.screens.Form 클래스에 지정됩니다. 스크린에서 사용할 수 있는 메서드 및 속성을 수정하려면 스크린을 다른 클래스에 지정합니다. ActionScript 클래스에 대한 자세한 내용은 *Adobe Flash에서 ActionScript 2.0 학습*의 “클래스”를 참조하십시오.

- 속성 관리자의  $x$  및  $y$  좌표 필드와 등록 포인트 격자에는 등록 포인트가 표시됩니다. 자세한 내용은 *Flash 사용 설명서*의 “스크린 속성 및 매개 변수 설정”을 참조하십시오. 스크린 내용을 좀더 쉽게 조작할 수 있도록 등록 포인트를 이동할 수 있습니다. 예를 들어, 스크린의 가운데에 회전하는 모양을 만들려면 스크린 등록 포인트의 위치를 스크린 가운데로 변경하고 이 등록 포인트 주위로 스크린을 회전하면 됩니다.

## 스크린과 ActionScript의 상호 작용 방식

스크린과 중첩 무비 클립은 ActionScript와 상호 작용하는 방식이 서로 비슷합니다. 자세한 내용은 *Flash 사용 설명서*의 “중첩된 무비 클립”을 참조하십시오. 단, 몇 가지 차이점이 있습니다.

스크린에서 ActionScript를 사용하는 경우 다음과 같은 지침이 적용됩니다.

- **스크린 요약** 창에서 스크린을 선택하고 ActionScript를 추가하면 스크립트가 객체 액션으로 해당 스크린에 직접 추가됩니다. 이는 ActionScript가 무비 클립에 직접 추가되는 것과 같습니다. 스크린 간의 탐색 기능 만들기 같은 간단한 코드에는 객체 액션을 사용하고, 좀더 복잡한 코드에는 외부 ActionScript 파일을 사용합니다.
- 최상의 결과를 얻으려면 ActionScript를 추가하기 전에 문서 구조를 구성하고 스크린 이름을 완성합니다. 스크린 이름을 변경하면 인스턴스 이름도 자동으로 변경되므로 작성하는 ActionScript 코드의 인스턴스 이름을 업데이트해야 합니다.
- 스크린의 타임라인에 프레임 액션을 추가하려면 스크린을 선택하고 기본적으로 축소되어 있는 타임라인을 확장한 다음, 타임라인에서 첫 번째 프레임을 선택합니다. 스크린의 복잡한 코드에는 프레임 액션 대신 외부 ActionScript 파일을 사용합니다.
- 스크린 기반 문서의 기본 타임라인은 보거나 조작할 수 없습니다. 그러나 대상 경로에 `_root`를 사용하여 기본 타임라인을 대상으로 지정할 수는 있습니다.
- 각 스크린은 해당 클래스에 따라 자동으로 ActionScript와 연관됩니다. 자세한 내용은 *Flash 사용 설명서*의 “슬라이드 스크린 및 양식 스크린”을 참조하십시오. 스크린이 지정되어 있는 클래스를 변경하고 속성 관리자에서 스크린의 일부 매개 변수를 설정할 수 있습니다. 자세한 내용은 *Flash 사용 설명서*의 “스크린 속성 및 매개 변수 설정”을 참조하십시오.
- ActionScript를 사용하여 스크린을 제어하려면 Screen 클래스, Slide 클래스 및 Form 클래스를 사용합니다.
- 대화형 작업을 만들려면 가급적 구성 요소를 사용합니다. 단일 FLA 파일에 구성 요소 인스턴스는 125개까지만 추가합니다.
- 슬라이드 간의 탐색 기능을 만들려면 `rootSlide`를 사용합니다. 예를 들어, 현재 슬라이드를 가져오려면 `rootSlide.currentSlide`를 사용합니다.
- `on(reveal)` 또는 `on(hide)` 핸들러 내부에서 슬라이드 탐색을 수행하려고 하지 마십시오.

- 스크린을 제어하는 ActionScript 코드에 on(keydown) 또는 on(keyup) 이벤트를 추가하지 마십시오.

ActionScript로 스크린을 제어하는 방법에 대한 자세한 내용은 *ActionScript 2.0 구성 요소 언어 참조 설명서*의 “Screen 클래스”, “Form 클래스” 및 “Slide 클래스”를 참조하십시오.

Object 클래스 및 onClipEvent() 이벤트 핸들러에 대한 자세한 내용은 *ActionScript 2.0 언어 참조 설명서*의 Object 및 onClipEvent 핸들러를 참조하십시오.

## 스크린에서 구성 요소 사용

스크린에서 구성 요소를 사용하여 Flash에 복잡하고 구조화된 응용 프로그램을 만들 수 있습니다. 데이터를 표시하고 비선형 사용자 대화형 작업을 수행할 수 있는 구조화된 응용 프로그램을 만드는 경우 구성 요소를 양식과 함께 사용하면 특히 유용합니다. 예를 들어, 양식을 사용하여 컨테이너 구성 요소를 채울 수 있습니다.

스크린에서 구성 요소를 사용하는 경우에는 포커스 관리자를 사용하여 구성 요소 간의 사용자 정의 탐색 기능을 만들 수 있습니다. 포커스 관리자에서는 사용자가 Tab 키를 눌러 응용 프로그램을 탐색할 때 구성 요소가 포커스를 받는 순서를 지정합니다. 예를 들어, Tab 키를 눌러 필드를 탐색하고 Return 키(Macintosh) 또는 Enter 키(Windows)를 눌러 양식을 제출하도록 양식 응용 프로그램을 사용자 정의할 수 있습니다.

포커스 관리자에 대한 자세한 내용은 *ActionScript 2.0 구성 요소 언어 참조 설명서*의 58페이지의 “사용자 정의 포커스 탐색 기능 만들기” 및 “FocusManager 클래스”를 참조하십시오.

액세스 가능성 패널을 사용하여 탭 순서를 정할 수도 있습니다. 자세한 내용은 *Flash 사용 설명서*의 “탭 순서와 읽기 순서 보거나 만들기”를 참조하십시오.

## 사용자 정의 포커스 탐색 기능 만들기

사용자가 Tab 키를 눌러 Flash 응용 프로그램 내에서 이동하거나 응용 프로그램 내에서 클릭하면 FocusManager 클래스는 입력 포커스를 받을 구성 요소를 결정합니다(자세한 내용은 *ActionScript 2.0 구성 요소 언어 참조 설명서*의 “FocusManager 클래스” 참조). 응용 프로그램에 FocusManager 인스턴스를 추가하거나 코드를 따로 작성하지 않아도 포커스 관리자가 활성화됩니다.

RadioButton 객체가 포커스를 받으면 포커스 관리자는 해당 객체 및 groupName 값이 동일한 모든 객체를 검사한 다음 selected 속성이 true로 설정되어 있는 객체에 포커스를 설정합니다.

각 모달 Window 구성 요소에 포커스 관리자의 인스턴스가 포함되어 있으므로 해당 윈도우 내에서만 자체적으로 탭이 설정됩니다. 따라서 사용자가 실수로 Tab 키를 눌러도 다른 윈도우의 구성 요소로 이동하지 않습니다.

응용 프로그램에서 포커스 탐색 기능을 만들려면 버튼을 비롯하여 포커스를 받을 모든 구성 요소에 `tabIndex` 속성을 설정해야 합니다. 사용자가 Tab 키를 누르면 `FocusManager` 클래스는 `tabIndex` 값이 `tabIndex`의 현재 값보다 큰 활성화된 객체를 찾습니다. `FocusManager` 클래스가 가장 높은 `tabIndex` 속성에 도달하면 0으로 돌아갑니다. 예를 들어, 다음 코드에서 `comment` 객체(`TextArea` 구성 요소)가 먼저 포커스를 받은 다음 `okButton` 인스턴스가 포커스를 받습니다.

```
var comment:mx.controls.TextArea;
var okButton:mx.controls.Button;
comment.tabIndex = 1;
okButton.tabIndex = 2;
```

**액세스 가능성** 패널을 사용하여 탭 인덱스 값을 할당할 수도 있습니다.

스태이지에 탭 인덱스 값을 가진 객체가 없으면 포커스 관리자는 심도 레벨( $z$  순서)을 사용합니다. 심도 레벨은 주로 구성 요소를 스테이지로 드래그하는 순서에 따라 설정되지만 **수정 > 정렬 > 맨 앞으로 가져오기/맨 뒤로 보내기** 명령을 사용하여 최종  $z$  순서를 결정할 수도 있습니다.

응용 프로그램에서 구성 요소에 포커스를 설정하려면 “`FocusManager.setFocus()`”를 호출합니다.

사용자가 Enter(Windows) 또는 Return(Macintosh) 키를 누를 때 포커스를 받는 버튼을 만들려면 다음 코드와 같이 “`FocusManager.defaultPushButton`” 속성을 원하는 버튼의 인스턴스로 설정합니다.

```
focusManager.defaultPushButton = okButton;
```

`FocusManager` 클래스(API)는 기본 Flash Player 포커스 사각형을 무시하고 모서리가 둥근 사용자 정의 포커스 사각형을 그립니다.

Flash 응용 프로그램에서 포커스 체계를 만드는 방법에 대한 자세한 내용은 *ActionScript 2.0 언어 및 구성 요소 참조 설명서*의 `FocusManager` 클래스를 참조하십시오.

## 문서 내에서 구성 요소 심도 관리

응용 프로그램에서 다른 객체의 앞이나 뒤에 구성 요소를 배치하려면 *ActionScript 2.0 구성 요소 언어 참조 설명서*의 `DepthManager` 클래스를 사용해야 합니다. `DepthManager` 클래스의 메서드를 사용하면 사용자 인터페이스 구성 요소를 적절한 *상대* 순서로 배치할 수 있습니다. 예를 들어, 콤보 상자를 열 때 해당 내용을 다른 구성 요소 앞에 표시하거나 삽입점을 모든 항목 앞에 표시하거나 대화 상자를 내용 위에 표시하는 것 등이 해당됩니다.

`Depth Manager`는 크게 두 가지 목적으로 사용됩니다. 즉, 문서 내에서의 상대적인 심도 할당을 관리하고, 루트 타임라인에서 커서나 도구 설명 같은 시스템 수준의 서비스에 대해 예약된 심도를 관리하기 위해 사용됩니다.

DepthManager를 사용하려면 DepthManager의 메서드를 호출합니다.

다음 코드는 button 구성 요소 아래에(중첩되는 경우 버튼 “아래”에 나타날 제작한 SWF 파일에) loader 구성 요소 인스턴스를 배치합니다.

```
loader.setDepthBelow(button);
```

예제 10

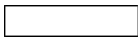
문서 내에서 레이어 및 수정 > 정렬 메뉴 옵션을 사용하여 상대 깊이를 관리할 수도 있습니다. 구성 요소는 레이어 및 정렬을 사용하여 런타임 깊이 관리할 때 무비 클립과 동일한 규칙을 따릅니다.

## 실시간 미리 보기의 구성 요소

기본적으로 활성화되어 있는 실시간 미리 보기 기능을 사용하면 제작된 Flash 내용에 구성 요소가 표시되는 모양을 스테이지에서 실제 크기로 미리 볼 수 있습니다. 실시간 미리 보기에 반영되는 매개 변수는 구성 요소마다 다릅니다. 실시간 미리 보기에 반영되는 구성 요소 매개 변수에 대한 자세한 내용은 *ActionScript 2.0 구성 요소 언어 참조 설명서*에서 각 구성 요소 항목을 참조하십시오.



실시간 미리 보기가 활성화된 Button 구성 요소



실시간 미리 보기가 비활성화된 Button 구성 요소

실시간 미리 보기 상태에서는 구성 요소가 작동하지 않습니다. 구성 요소의 기능을 테스트하려면 **컨트롤 > 무비 테스트**를 사용해야 합니다.

### 실시간 미리 보기를 활성화하거나 비활성화하려면:

- **컨트롤 > 실시간 미리 보기 활성화**를 선택합니다. 이 옵션 옆에 있는 체크 상자가 선택되어 있으면 실시간 미리 보기가 활성화되어 있다는 것을 나타냅니다.

# 구성 요소에 프리로더 사용

미리 로드하는 작업은 사용자가 SWF 파일과 상호 작용하기 전에 SWF 파일에 대한 데이터 일부를 로드하는 것입니다. 기본적으로 구성 요소와 클래스는 구성 요소가 포함된 문서의 첫 프레임으로 내보내도록 설정됩니다. 구성 요소와 클래스는 첫 번째로 로드되는 데이터임으로 진행률 막대를 구현하거나 애니메이션을 로드할 때 문제가 발생할 수 있습니다. 특히 구성 요소와 클래스를 진행률 막대보다 먼저 로드할 수 있지만 진행률 막대를 통해 클래스를 포함한 모든 데이터의 로드 진행률을 나타내야 합니다. 그러므로 SWF 파일의 다른 부분 이후, 그리고 구성 요소를 사용하기 이전에 클래스를 로드합니다.

이렇게 하려면 구성 요소가 포함된 응용 프로그램의 사용자 정의 프리로더를 만들 때 제작 설정에서 모든 클래스를 구성 요소가 포함된 클래스로 내보내도록 설정합니다. 예셋을 첫 프레임으로 내보내도록 설정한 모든 구성 요소 목록을 Halo 및 Sample 테마에서 확인하려면 113페이지의 “내보내기 설정 변경”을 참조하십시오.

## 모든 클래스에 대해 내보내기 프레임을 변경하려면:

1. 파일 > 제작 설정을 선택합니다.
2. 제작 설정 대화 상자의 Flash 탭에서 ActionScript 버전이 ActionScript 2.0으로 설정되어 있는지 확인합니다.
3. ActionScript 버전의 오른쪽에 있는 설정 버튼을 클릭합니다.
4. ActionScript 2.0 설정의 클래스용 내보내기 프레임 텍스트 상자에서 구성 요소가 처음 나타나는 프레임의 번호를 변경합니다.

클래스를 로드하려고 선택한 프레임에 재생 헤드가 도달해야 클래스를 사용할 수 있습니다. 구성 요소가 제대로 작동하려면 클래스가 필요하므로 클래스를 로드하도록 지정된 프레임 이후에 구성 요소를 로드해야 합니다. 프레임 3에 내보내는 경우 재생 헤드가 프레임 3에 도달하여 데이터를 로드할 때까지 해당 클래스에서 어떤 것도 사용할 수 없습니다.

구성 요소를 사용하는 파일을 미리 로드하려면 SWF 파일에서도 구성 요소를 미리 로드해야 합니다. 이 작업을 수행하려면 SWF 파일의 다른 프레임에 내보내도록 구성 요소를 설정해야 합니다.

<b>예</b> <b>10</b>	ActionScript를 사용하여 스테이지에 구성 요소를 추가하는 경우 추가하려는 구성 요소의 인스턴스를 스테이지 주위의 붙여넣기 영역으로 드래그해야 합니다. Flash에서 이 구성 요소는 사용하지 않은 라이브러리 항목과 구분되며, 사용자의 응용 프로그램에서 사용되는 구성 요소로 인식됩니다. Flash에서 사용하지 않은 라이브러리 항목은 SWF 파일에 추가되지 않습니다.
-----------------------	--

### 구성 요소를 내보낼 프레임을 변경하려면:

1. 윈도우 > 라이브러리를 선택하여 라이브러리 패널을 엽니다.
2. 라이브러리에서 구성 요소를 마우스 오른쪽 버튼으로 클릭(Windows)하거나 **Control** 키를 누른 상태에서 클릭(Macintosh)합니다.
3. 컨텍스트 메뉴에서 링크를 선택합니다.
4. 첫 프레임으로 내보내기 선택을 해제합니다.
5. 확인을 클릭합니다.
6. 파일 > 제작 설정을 선택합니다.
7. Flash 탭을 선택하고 설정 버튼을 클릭합니다.
8. 클래스용 내보내기 프레임 텍스트 상자에 숫자를 입력하고 확인을 클릭합니다. 클래스가 이 프레임에 로드됩니다.
9. 확인을 클릭하여 제작 설정 대화 상자를 닫습니다.

구성 요소가 첫 번째 프레임에 로드되지 않는 경우 SWF 파일의 첫 번째 프레임에 사용자 정의 진행률 막대를 만들 수 있습니다. 7단계에서 지정한 프레임에 클래스가 로드될 때까지 ActionScript에서 구성 요소를 참조하거나 구성 요소를 스테이지에 포함하지 하십시오.



ActionScript 클래스 이후에 구성 요소를 내보내야 합니다.

## 구성 요소 로드

버전 2 구성 요소를 SWF 파일이나 Loader 구성 요소로 로드하는 경우 구성 요소가 올바르게 작동하지 않을 수 있습니다. 이러한 구성 요소로는 Alert, ComboBox, DateField, Menu, MenuBar, Window가 있습니다.

loadMovie()를 호출하거나 Loader 구성 요소에 로드할 때 `_lockroot` 속성을 사용합니다. Loader 구성 요소를 사용할 경우에는 다음 코드를 추가합니다.

```
myLoaderComponent.content._lockroot = true;
```

loadMovie()에 대한 호출에서 무비 클립을 사용할 경우에는 다음 코드를 추가합니다.

```
myMovieClip._lockroot = true;
```

로더 무비 클립에서 `_lockroot`를 true로 설정하지 않으면 로더는 자체 라이브러리에만 액세스할 수 있고 로드된 무비 클립의 라이브러리에는 액세스할 수 없습니다.

`_lockroot` 속성은 Flash Player 7에서 지원됩니다. 이 속성에 대한 자세한 내용은 *ActionScript 2.0 언어 참조 설명서*의 `_lockroot` (MovieClip.\_lockroot property)를 참조하십시오.

# 버전 1 구성 요소를 버전 2 아키텍처로 업그레이드

버전 2 구성 요소는 이벤트 [[www.w3.org/TR/DOM-Level-3-Events/events.html](http://www.w3.org/TR/DOM-Level-3-Events/events.html)], 스타일, getter/setter 정책 등과 관련된 여러 웹 표준을 준수하도록 작성되었으며 Macromedia Flash MX의 버전 1 구성 요소 및 Macromedia Flash MX 2004 이전에 출시된 DRK에 제공된 버전 1 구성 요소와는 매우 다릅니다. 버전 2 구성 요소는 버전 1 구성 요소와는 다른 API를 사용하며 ActionScript 2.0으로 작성되었습니다. 따라서 응용 프로그램에서 버전 1 구성 요소와 버전 2 구성 요소를 함께 사용하면 예상치 못한 결과가 발생할 수 있습니다. 버전 2 이벤트 처리, 스타일 및 메서드가 아닌 속성에 대한 getter/setter 액세스를 사용하도록 버전 1 구성 요소를 업그레이드하는 방법에 대한 자세한 내용은 [121페이지의 제6장](#), “구성 요소 만들기”를 참조하십시오.

버전 1 구성 요소가 포함된 Flash 응용 프로그램은 Flash Player 6 또는 Flash Player 6(6.0.65.0) 용으로 제작된 경우에만 Flash Player 6과 Flash Player 7에서 제대로 작동합니다. Flash Player 7 이상용으로 제작된 경우에도 작동하도록 응용 프로그램을 업그레이드하려면 고정 데이터 유형을 사용하도록 코드를 변환해야 합니다. 자세한 내용은 *Adobe Flash에서 ActionScript 2.0 학습*의 “사용자 정의 클래스 파일 작성”을 참조하십시오.



모든 구성 요소에는 사용자가 구성 요소와 상호 작용할 때 브로드캐스팅되는 이벤트(예: click 및 change 이벤트)나 구성 요소에 중요한 무언가가 발생할 때 브로드캐스팅되는 이벤트(예: load 이벤트)가 있습니다. 이벤트를 처리하려면 이벤트가 발생할 때 실행되는 ActionScript 코드를 작성해야 합니다.

각 구성 요소는 자체 이벤트 집합을 브로드캐스팅합니다. 이 집합에는 구성 요소가 상속 받은 모든 클래스의 이벤트가 포함되어 있습니다. 즉, UIObject 클래스와 UIComponent 클래스가 버전 2 아키텍처의 기본 클래스이기 때문에 미디어 구성 요소를 제외한 모든 구성 요소는 UIObject 클래스와 UIComponent 클래스에서 이벤트를 상속 받습니다. 구성 요소가 브로드캐스팅하는 이벤트 목록을 보려면 *Action Script 2.0 구성 요소 언어 참조 설명서*에서 각 구성 요소 항목과 조상 클래스 항목을 참조하십시오.

이 장에서는 간단한 Flash 응용 프로그램인 TipCalculator의 여러 버전을 사용하여 구성 요소 이벤트를 처리하는 방법을 배웁니다. TipCalculator 샘플은 Flash 샘플 페이지 ([www.adobe.com/go/learn\\_fl\\_samples\\_kr](http://www.adobe.com/go/learn_fl_samples_kr))를 참조하십시오.

이 장에서 설명하는 단원은 다음과 같습니다.

리스너를 사용하여 이벤트 처리 .....	66
이벤트 위임 .....	74
이벤트 객체 .....	77
on() 이벤트 핸들러 사용 .....	78

# 리스너를 사용하여 이벤트 처리

버전 2 구성 요소 아키텍처에는 브로드캐스터/리스너 이벤트 모델이 있습니다. **브로드캐스터**를 **디스패처**라고 하기도 합니다. 다음은 모델에 대해 알아야 하는 주요 사항입니다.

- 구성 요소 클래스의 인스턴스가 모든 이벤트를 브로드캐스팅합니다. 구성 요소 인스턴스는 **브로드캐스터**입니다.
- **리스너**는 함수 또는 객체일 수 있습니다. 리스너가 객체인 경우에는 콜백 함수가 정의되어 있어야 합니다. 리스너가 이벤트를 **처리**합니다. 즉, 이벤트가 발생하면 콜백 함수가 실행됩니다.
- 리스너를 브로드캐스터에 등록하려면 브로드캐스터에서 `addEventListener()` 메서드를 호출합니다. 다음 구문을 사용합니다.

```
componentInstance.addEventListener("eventName",  
    listenerObjectORFunction);
```

- 구성 요소 인스턴스 하나에 리스너를 여러 개 등록할 수 있습니다.

```
myButton.addEventListener("click", listener1);  
myButton.addEventListener("click", listener2);
```

- 여러 구성 요소 인스턴스에 리스너를 하나만 등록할 수 있습니다.

```
myButton.addEventListener("click", listener1);  
myButton2.addEventListener("click", listener1);
```

- 핸들러 함수에 이벤트 객체가 전달됩니다.

이벤트를 브로드캐스팅하는 인스턴스와 이벤트 유형에 대한 정보를 검색하기 위해 함수 본문에 이벤트 객체를 사용할 수 있습니다. 자세한 내용은 [77페이지의 “이벤트 객체”](#)를 참조하십시오.

- 리스너 객체는 “`EventDispatcher.removeEventListener()`”를 사용하여 명시적으로 제거하기 전까지 활성 상태를 유지합니다. 예를 들면 다음과 같습니다.

```
myComponent.removeEventListener("change", listenerObj);
```

## 리스너 객체 사용

리스너 객체를 사용하기 위해 `this` 키워드를 사용하여 현재 객체를 리스너로 지정하거나 이미 응용 프로그램에 있는 객체를 사용하거나 새 객체를 만들 수 있습니다.

- 대부분의 경우에는 `this`를 사용합니다.  
범위에 이벤트가 브로드캐스팅될 때 응답해야 할 구성 요소가 포함되기 때문에 대개 현재 객체(`this`)를 리스너로 사용하는 것이 가장 쉽습니다.
- 기존 객체를 사용하는 것이 편하면 그렇게 합니다.  
예를 들어, 이벤트에 응답하는 구성 요소가 양식에 포함된 경우 Flash 양식 응용 프로그램에서 양식을 리스너 객체로 사용할 수 있습니다. 양식 타임라인의 프레임에 코드를 배치합니다.
- 많은 구성 요소가 하나의 이벤트(예: `click` 이벤트)를 브로드캐스팅할 경우 특정 리스너 객체만 응답하도록 하려면 새 리스너 객체를 사용합니다.

`this` 객체를 사용할 경우에는 처리할 이벤트와 같은 이름으로 함수를 정의합니다. 구문은 다음과 같습니다.

```
function eventName(evtObj:Object){  
    // 여기에 코드를 입력하십시오.  
};
```

새 리스너 객체를 사용할 경우에는 다음과 같이 객체를 생성하고 이벤트와 같은 이름으로 속성을 정의하며 이벤트가 브로드캐스팅될 때 실행되는 콜백 함수에 속성을 할당해야 합니다.

```
var listenerObject:Object = new Object();  
listenerObject.eventName = function(evtObj:Object){  
    // 여기에 코드를 입력하십시오.  
};
```

기존 객체를 사용할 경우에는 다음과 같이 새 객체를 만들지 않은 채 새 리스너 객체와 같은 구문을 사용합니다.

```
existingObject.eventName = function(evtObj:Object){  
    // 여기에 코드를 입력하십시오.  
};
```

**만  
년**

`evtObj` 매개 변수는 이벤트가 트리거되어 콜백 함수에 전달될 때 자동으로 생성되는 객체입니다. 이 이벤트 객체의 속성에는 이벤트에 대한 정보가 들어 있습니다. 자세한 내용은 [77페이지의 “이벤트 객체”](#)를 참조하십시오.

마지막으로 이벤트를 브로드캐스팅하는 구성 요소 인스턴스에서 `addEventListener()` 메서드를 호출합니다. `addEventListener()` 메서드는 두 개의 매개 변수, 즉 이벤트 이름을 나타내는 문자열과 리스너 객체에 대한 참조를 사용합니다.

```
componentInstance.addEventListener("eventName", listenerObject);
```

다음은 복사하여 붙여 넣을 수 있는 전체 코드입니다. 기울임체로 된 모든 코드를 실제 값으로 바꾸어야 합니다. `listenerObject` 및 `evtObj` 또는 다른 모든 유효한 식별자를 사용할 수 있지만 `eventName`을 이벤트 이름으로 변경해야 합니다.

```
var listenerObject:Object = new Object();
listenerObject.eventName = function(evtObj:Object){
    // 여기에 입력된 코드는 이벤트가
    // 트리거될 때 실행됩니다.
};
componentInstance.addEventListener("eventName", listenerObject);
```

다음 코드에서는 `this` 키워드를 리스너 객체로 사용합니다.

```
function eventName(evtObj:Object){
    // 여기에 입력된 코드는 이벤트가
    // 트리거될 때 실행됩니다.
}
componentInstance.addEventListener("eventName", this);
```

모든 구성 요소 인스턴스에서 `addEventListener()`를 호출할 수 있습니다. 이것은 `EventDispatcher` 클래스에서 모든 구성 요소에 혼합됩니다. “mix-in”은 다른 클래스의 비헤이비어를 강화하는 특정 기능을 제공하는 클래스입니다. 자세한 내용은 *ActionScript 2.0 구성 요소 언어 참조 설명서*의 “`EventDispatcher.addEventListener()`”를 참조하십시오.

구성 요소에서 브로드캐스팅하는 이벤트에 대한 자세한 내용은 *ActionScript 2.0 구성 요소 언어 참조 설명서*에서 해당 구성 요소 항목을 참조하십시오. 예를 들어, `Button` 구성 요소 이벤트는 `Button` 구성 요소 섹션(또는 [도움말 > ActionScript 2.0 구성 요소 언어 참조 설명서 > Button 구성 요소 > Button 클래스 > Button 클래스의 이벤트 요약](#))에 나열됩니다.

### Flash(FLA) 파일에서 리스너 객체를 등록하려면:

1. Flash에서 **파일 > 새로 만들기**를 선택하고 새 Flash 문서를 만듭니다.
2. `Button` 구성 요소를 **구성 요소** 패널에서 스테이지로 드래그합니다.
3. 속성 관리자에서 인스턴스 이름으로 **myButton**을 입력합니다.
4. `TextInput` 구성 요소를 **구성 요소** 패널에서 스테이지로 드래그합니다.
5. 속성 관리자에서 인스턴스 이름으로 **myText**를 입력합니다.
6. 타임라인에서 **프레임 1**을 선택합니다.
7. 윈도우 > **액션**을 선택합니다.
8. **액션** 패널에 다음 코드를 입력합니다.

```
var myButton:mx.controls.Button;
var myText:mx.controls.TextInput;

function click(evt){
    myText.text = evt.target;
```

```
}
```

```
myButton.addEventListener("click", this);
```

이벤트 객체 evt의 target 속성은 이벤트를 브로드캐스팅하는 인스턴스에 대한 참조입니다. 이 코드는 TextInput 구성 요소에 target 속성의 값을 표시합니다.

### 클래스 (AS) 파일에서 리스너 객체를 등록하려면:

1. 47페이지의 “구성 요소를 사용한 작업”에 지정된 위치에서 TipCalculator.fla 파일을 엽니다.
2. 47페이지의 “구성 요소를 사용한 작업”에 지정된 위치에서 TipCalculator.as 파일을 엽니다.
3. FLA 파일에서 form1을 선택하고 속성 관리자에서 클래스 이름 TipCalculator를 확인합니다. 이것은 양식과 클래스 파일 간의 링크입니다. 이 응용 프로그램의 모든 코드는 TipCalculator.as 파일에 있습니다. 양식에서는 클래스로 정의된 속성과 비헤이비어가 코드에 할당되었다고 간주합니다.
4. AS 파일에서 25행인 public function onLoad():Void로 스크롤합니다.  
onLoad() 함수는 양식이 Flash Player에 로드될 때 실행됩니다. 함수 본문에서 subtotal TextInput 인스턴스와 percentRadio15, percentRadio18 및 percentRadio20의 세 RadioButton 인스턴스는 addEventListener() 메서드를 호출하여 리스너를 이벤트에 등록합니다.
5. 27행인 subtotal.addEventListener("change", this)를 확인합니다.  
addEventListener()를 호출할 때 두 개의 매개 변수를 전달해야 합니다. 첫 번째는 브로드캐스팅되는 이벤트 이름을 나타내는 문자열(이 경우 "change")입니다. 두 번째는 이벤트를 처리하는 객체나 함수에 대한 참조입니다. 이 경우 매개 변수는 클래스 파일 인스턴스(객체)를 나타내는 this 키워드입니다. 그런 다음 Flash에서 객체에 이벤트와 같은 이름의 함수가 있는지 확인합니다.
6. 63행인 public function change(event:Object):Void를 확인합니다.  
이것은 subtotal TextInput 인스턴스가 변경될 때 실행되는 함수입니다.
7. TipCalculator.fla를 선택하고 **컨트롤 > 무비 테스트**를 선택하여 파일을 테스트합니다.

## handleEvent 콜백 함수 사용

handleEvent 함수를 지원하는 리스너 객체를 사용할 수도 있습니다. 브로드캐스팅되는 이벤트 이름에 상관없이 리스너 객체의 handleEvent 메서드가 호출됩니다. 여러 이벤트를 처리하려면 if..else 또는 switch 문을 사용해야 합니다. 예를 들어, 다음 코드는 if..else 문을 사용하여 click 및 change 이벤트를 처리합니다.

```
// handleEvent 함수를 정의합니다.
// 함수에 evt를 이벤트 객체 매개 변수로 전달합니다.

function handleEvent(evt){
  // 이벤트가 click인지 확인합니다.
  if (evt.type == "click"){
    // 이벤트가 click이면 작업을 수행합니다.
  } else if (evt.type == "change"){
    // 이벤트가 change이면 다른 작업을 수행합니다.
  }
};

// 다른 두 구성 요소 인스턴스에
// 리스너 객체를 등록합니다.
// "this" 객체에 함수가 정의되어 있기 때문에
// 리스너는 this입니다.

instance.addEventListener("click", this);
instance2.addEventListener("change", this);
```

## 리스너 함수 사용

handleEvent 구문과는 달리 여러 리스너 함수가 서로 다른 이벤트를 처리할 수 있습니다. 따라서 myHandler에서 if 및 else if를 체크 인하는 대신에 다음과 같이 change 이벤트에 대해서는 myChangeHandler를, scroll 이벤트에 대해서는 myScrollHandler를 정의하여 등록할 수 있습니다.

```
myList.addEventListener("change", myChangeHandler);
myList.addEventListener("scroll", myScrollHandler);
```

리스너 함수를 사용하려면 먼저 다음과 같이 함수를 정의해야 합니다.

```
function myFunction(evtObj:Object){
  // 여기에 코드를 입력하십시오.
}
```

**참  
고**

evtObj 매개 변수는 이벤트가 트리거되어 함수에 전달될 때 자동으로 생성되는 객체입니다. 이 이벤트 객체의 속성에는 이벤트에 대한 정보가 들어 있습니다. 자세한 내용은 [77페이지의 “이벤트 객체”](#)를 참조하십시오.

그런 다음 이벤트를 브로드캐스팅하는 구성 요소 인스턴스에서 `addEventListener()` 메서드를 호출합니다. `addEventListener()` 메서드는 두 개의 매개 변수, 즉 이벤트 이름을 나타내는 문자열과 함수에 대한 참조를 사용합니다.

```
componentInstance.addEventListener("eventName", myFunction);
```

모든 구성 요소 인스턴스에서 `addEventListener()`를 호출할 수 있습니다. 이것은 `EventDispatcher` 클래스에서 모든 UI 구성 요소에 포함됩니다. 자세한 내용은 *ActionScript 2.0 구성 요소 언어 참조 설명서*의 “`EventDispatcher.addEventListener()`”를 참조하십시오.

구성 요소에서 브로드캐스팅하는 이벤트에 대한 자세한 내용은 *ActionScript 2.0 구성 요소 언어 참조 설명서*에서 해당 구성 요소 항목을 참조하십시오.

### Flash(FLA) 파일에서 리스너 객체를 등록하려면:

1. Flash에서 **파일 > 새로 만들기**를 선택하고 새 Flash 문서를 만듭니다.
2. List 구성 요소를 **구성 요소 패널**에서 스테이지로 드래그합니다.
3. 속성 관리자에서 인스턴스 이름으로 **myList**를 입력합니다.
4. 타임라인에서 **프레임 1**을 선택합니다.
5. 윈도우 > **액션**을 선택합니다.
6. 액션 패널에 다음 코드를 입력합니다.

```
// 변수를 선언합니다 .
var myList:mx.controls.List;
var myHandler:Function;

// 목록에 항목을 추가합니다 .
myList.addItem("Bird");
myList.addItem("Dog");
myList.addItem("Fish");
myList.addItem("Cat");
myList.addItem("Ape");
myList.addItem("Monkey");

// myHandler 함수를 정의합니다 .
function myHandler(eventObj:Object){

    // eventObj 매개 변수를 사용하여
    // 이벤트 유형을 캡처합니다 .
    if (eventObj.type == "change"){
        trace("The list changed");
    } else if (eventObj.type == "scroll"){
        trace("The list was scrolled");
    }
}

// myList 에 myHandler 함수를 등록합니다 .
```

```
// 항목을 선택 (change 이벤트 트리거) 하거나
// 목록을 스크롤하면 myHandler 가 실행됩니다 .
myList.addEventListener("change", myHandler);
myList.addEventListener("scroll", myHandler);
```

**참고**

이벤트 객체 evt의 type 속성은 이벤트 이름에 대한 참조입니다.

7. 컨트롤 > 무비 테스트를 선택한 다음 목록에서 항목을 선택하고 목록을 스크롤하여 출력 패널에서 결과를 확인합니다.

**CAUTION**

리스너 함수에서 this 키워드는 타임라인이나 함수가 정의된 클래스 대신에 addEventListener()를 호출하는 구성 요소 인스턴스를 나타냅니다. 하지만 Delegate 클래스를 사용하여 리스너 함수를 다른 범위에 위임할 수 있습니다. 자세한 내용은 74페이지의 “이벤트 위임”을 참조하십시오. 함수 범위를 지정하는 예제를 보려면 다음 단원을 참조하십시오.

## 리스너의 범위

범위는 함수가 실행되는 객체를 나타냅니다. 함수 내의 모든 변수 참조를 해당 객체의 속성으로 인식합니다. Delegate 클래스를 사용하여 리스너의 범위를 지정할 수 있습니다. 자세한 내용은 74페이지의 “이벤트 위임”을 참조하십시오.

앞서 설명한 것과 같이 addEventListener()를 호출하여 리스너를 구성 요소 인스턴스에 등록합니다. 이 메서드는 두 개의 매개 변수, 즉 이벤트 이름을 나타내는 문자열과 객체 또는 함수에 대한 참조를 사용합니다. 다음 표는 각 매개 변수 유형의 범위 목록입니다.

리스너 유형	범위
객체	리스너 객체
함수	이벤트를 브로드캐스팅하는 구성 요소 인스턴스

addEventListener()에 객체를 전달하면 객체의 범위에서 해당 객체에 할당된 콜백 함수나 객체에 정의된 함수가 호출됩니다. 즉, this 키워드는 콜백 함수 내에서 사용될 때 다음과 같이 리스너 객체를 나타냅니다.

```
var lo:Object = new Object();
lo.click = function(evt){
    // this 는 객체 lo 를 나타냅니다 .
    trace(this);
}
myButton.addEventListener("click", lo);
```

하지만 `addEventListener()`에 함수를 전달하면 `addEventListener()`를 호출하는 구성 요소 인스턴스의 범위에서 함수가 호출됩니다. 즉, `this` 키워드는 함수 내에서 사용될 때 브로드캐스팅하는 구성 요소 인스턴스를 참조합니다. 따라서 클래스 파일에서 함수를 정의할 경우 문제가 발생합니다. `this`가 클래스 인스턴스를 가리키지 않기 때문에 예상된 경로를 사용하여 클래스 파일의 속성과 메서드에 액세스할 수 없습니다. 이 문제를 해결하려면 `Delegate` 클래스를 사용하여 함수를 올바른 범위에 위임합니다. 자세한 내용은 74페이지의 “이벤트 위임”을 참조하십시오.

다음 코드에서는 클래스 파일에서 `addEventListener()`에 전달되는 함수의 범위 지정을 보여 줍니다. 이 코드를 사용하려면 `Cart.as`라는 `ActionScript(AS)` 파일에 복사하고 `Button` 구성 요소 `myButton`과 `DataGrid` 구성 요소 `myGrid`를 사용하여 `Flash(FLA)` 파일을 만듭니다. 스테이지에서 두 구성 요소를 모두 선택하고 `F8` 키를 눌러 `Cart`라는 새 심볼로 변환한 다음 `Cart` 심볼의 링크 속성에서 `Cart` 클래스에 할당합니다.

```
class Cart extends MovieClip {

    var myButton:mx.controls.Button;
    var myGrid:mx.controls.DataGrid;

    function myHandler(eventObj:Object){

        // eventObj 매개 변수를 사용하여
        // 이벤트 유형을 캡처합니다 .
        if (eventObj.type == "click"){

            /* this 의 값을 출력 패널로 보냅니다 .
            myHandler 가 리스너 객체에 정의되지 않은 함수이기 때문에
            this 는 myHandler 가 등록된 구성 요소 인스턴스 (myButton) 에
            대한 참조입니다 . 또한 this 는 Cart 클래스의 인스턴스를 참조하지
            않기 때문에 myGrid 는 정의되지 않습니다 .
            버튼을 누를 때 레이블 및 / 또는 아이콘을 오프셋하는 데 사용할 숫자
            */
            trace("this: " + this);
            trace("myGrid: " + myGrid);
        }
    }

    // myButton 에 myHandler 함수를 등록합니다 .
    // 버튼을 클릭하면 myHandler 가 실행됩니다 .

    function onLoad():Void{
        myButton.addEventListener("click", myHandler);
    }
}
```

# 이벤트 위임

Delegate 클래스를 스크립트나 클래스로 가져와서 이벤트를 특정 범위와 함수에 위임할 수 있습니다(*ActionScript 2.0 구성 요소 언어 참조 설명서*의 “DeltaItem 클래스” 참조). Delegate 클래스를 가져오려면 다음 구문을 사용합니다.

```
import mx.utils.Delegate;
compInstance.addEventListener("eventName", Delegate.create(scopeObject,
    function));
```

*scopeObject* 매개 변수는 지정한 *function* 매개 변수가 호출되는 범위를 지정합니다.

Delegate.create() 호출의 일반적인 용도는 다음 두 가지입니다.

- 같은 이벤트를 두 함수에 전달합니다.  
자세한 내용은 다음 단원을 참조하십시오.
- 포함하는 클래스의 범위 내에서 함수를 호출합니다.  
함수를 addEventListener()에 매개 변수로 전달하면 함수가 선언된 객체가 아니라 해당 함수가 브로드캐스터 구성 요소 인스턴스의 범위에서 호출됩니다. 자세한 내용은 [76 페이지의 “함수의 범위 위임”](#)을 참조하십시오.

## 함수에 이벤트 위임

Delegate.create() 호출은 이름이 같은 이벤트를 브로드캐스팅하는 구성 요소가 두 개 있는 경우에 유용합니다. 예를 들어, 체크 박스와 버튼이 있는 경우 click 이벤트를 브로드캐스팅하고 있는 구성 요소를 확인하기 위해 eventObject.target 속성에서 가져오는 정보에 대해 switch 문을 사용해야 합니다.

다음 코드를 사용하려면 스테이지에 myCheckBox\_chb라는 체크 상자와 myButton\_btn이라는 버튼을 배치합니다. 두 인스턴스를 모두 선택하고 F8 키를 눌러 새 심볼을 만듭니다. 대화 상자가 기본 모드이면 **고급**을 클릭하고 **ActionScript에 내보내기**를 선택합니다. **AS 2.0** 클래스 텍스트 상자에 **Cart**를 입력합니다. 속성 관리자에서 새 심볼의 인스턴스 이름을 원하는 대로 설정합니다. 이제 심볼이 Cart 클래스와 연관되며 심볼의 인스턴스는 이 클래스의 인스턴스가 됩니다.

```
import mx.controls.Button;
import mx.controls.CheckBox;

class Cart {
    var myCheckBox_chb:CheckBox;
    var myButton_btn:Button;

    function onLoad() {
        myCheckBox_chb.addEventListener("click", this);
```

```

        myButton_btn.addEventListener("click", this);
    }

    function click(eventObj:Object) {
        switch(eventObj.target) {
            case myButton_btn:
                // 출력 패널에 브로드캐스터 인스턴스
                // 이름과 이벤트 유형을 보냅니다 .
                trace(eventObj.target + ": " + eventObj.type);
                break;
            case myCheckBox_chb:
                trace(eventObj.target + ": " + eventObj.type);
                break;
        }
    }
}

```

다음 코드는 **Delegate**를 사용하도록 수정된 같은 클래스 파일(**Cart.as**)입니다.

```

import mx.utils.Delegate;
import mx.controls.Button;
import mx.controls.CheckBox;

class Cart {
    var myCheckBox_chb:CheckBox;
    var myButton_btn:Button;

    function onLoad() {
        myCheckBox_chb.addEventListener("click", Delegate.create(this,
        chb_onClick));
        myButton_btn.addEventListener("click", Delegate.create(this,
        btn_onClick));
    }
}

```

// 두 개의 개별 함수가 이벤트를 처리합니다 .

```

function chb_onClick(eventObj:Object) {
    // 출력 패널에 브로드캐스터 인스턴스
    // 이름과 이벤트 유형을 보냅니다 .
    trace(eventObj.target + ": " + eventObj.type);
    // 출력 패널에 FLA 파일의 Cart
    // 클래스와 연결된 심볼의 절대 경로를
    // 보냅니다 .
    trace(this)
}

function btn_onClick(eventObj:Object) {
    trace(eventObj.target + ": " + eventObj.type);
}
}

```

## 함수의 범위 위임

`addEventListener()` 메서드는 두 개의 매개 변수, 즉 이벤트 이름과 리스너에 대한 참조가 필요합니다. 리스너는 객체일 수도 있고 함수일 수도 있습니다. 객체를 전달하면 객체의 범위에서 객체에 할당된 콜백 함수가 호출됩니다. 하지만 함수를 전달하면 `addEventListener()`를 호출하는 구성 요소 인스턴스의 범위에서 함수가 호출됩니다. 자세한 내용은 [72페이지의 “리스너의 범위”](#)를 참조하십시오.

브로드캐스터 인스턴스의 범위에서 함수가 호출되기 때문에 함수 본문의 `this` 키워드는 함수를 포함하는 클래스가 아니라 브로드캐스터 인스턴스를 가리킵니다. 따라서 해당 함수를 포함하는 클래스의 속성과 메서드에 액세스할 수 없습니다. 이 경우 `Delegate` 클래스를 사용하여 함수의 범위를 포함하는 클래스에 위임하면 포함하는 클래스의 속성과 메서드에 액세스할 수 있습니다.

다음 예제에서는 변형된 `Cart.as` 클래스 파일을 사용하여 이전 단원과 동일한 방법을 따릅니다.

```
import mx.controls.Button;
import mx.controls.CheckBox;

class Cart {

    var myCheckBox_chb:CheckBox;
    var myButton_btn:Button;

    // chb_onClick 함수에서 액세스할
    // 변수를 정의합니다.
    var i:Number = 10

    function onLoad() {
        myCheckBox_chb.addEventListener("click", chb_onClick);
    }

    function chb_onClick(eventObj:Object) {
        // i 변수에 액세스하고 10을 출력할 수
        // 있을 것으로 예상하겠지만 i가
        // 정의된 Cart 인스턴스로 함수 범위가
        // 지정되지 않았기 때문에
        // 출력 패널에 undefined가
        // 보내집니다.
        trace(i);
    }
}
```

`Cart` 클래스의 속성과 메서드에 액세스하려면 다음과 같이 `Delegate.create()`를 `addEventListener()`의 두 번째 매개 변수로 호출합니다.

```
import mx.utils.Delegate;
```

```

import mx.controls.Button;
import mx.controls.CheckBox;

class Cart {
    var myCheckBox_chb:CheckBox;
    var myButton_btn:Button;
    // chb_onClick 함수에서 액세스할
    // 변수를 정의합니다.
    var i:Number = 10

    function onLoad() {
        myCheckBox_chb.addEventListener("click", Delegate.create(this,
        chb_onClick));
    }

    function chb_onClick(eventObj:Object) {
        // Cart 인스턴스로 함수 범위가
        // 지정되었기 때문에 출력 패널에
        // 10을 보냅니다.
        trace(i);
    }
}

```

## 이벤트 객체

이벤트 객체는 `ActionScript` 객체 클래스의 인스턴스이며 이벤트에 대한 정보를 포함하는 다음 속성이 있습니다.

속성	설명
<code>type</code>	이벤트 이름을 나타내는 문자열입니다.
<code>target</code>	이벤트를 브로드캐스팅하는 구성 요소 인스턴스에 대한 참조입니다.

이벤트에 추가 속성이 있으면 구성 요소 사전의 해당 이벤트 항목에 나열됩니다.

이벤트 객체는 이벤트가 트리거되어 리스너 객체의 콜백 함수나 리스너 함수에 전달될 때 자동으로 생성됩니다.

함수 내에 이벤트 객체를 사용하여 브로드캐스팅된 이벤트의 이름이나 이벤트를 브로드캐스팅한 구성 요소의 인스턴스 이름에 액세스할 수 있습니다. 인스턴스 이름에서 다른 구성 요소 속성에 액세스할 수 있습니다. 예를 들어, 다음 코드는 `evtObj` 이벤트 객체의 `target` 속성을 사용하여 `myButton` 인스턴스의 `label` 속성에 액세스하고 해당 값을 출력 패널에 보냅니다.

```

var myButton:mx.controls.Button;
var listener:Object;

```

```

listener = new Object();

listener.click = function(evtObj){
    trace("The " + evtObj.target.label + " button was clicked");
}
myButton.addEventListener("click", listener);

```

## on() 이벤트 핸들러 사용

버튼이나 무비 클립에 핸들러를 할당할 때와 마찬가지로 구성 요소 인스턴스에 on() 이벤트 핸들러를 할당할 수 있습니다. on() 이벤트 핸들러는 간단한 테스트에 유용할 수 있지만 모든 응용 프로그램에서는 이벤트 리스너가 사용됩니다. 자세한 내용은 [66페이지의 “리스너를 사용하여 이벤트 처리”](#)를 참조하십시오.

구성 요소(액션 패널의 *instance* 구성 요소에 할당된)에 첨부한 on() 핸들러 안의 this 키워드를 사용하면 this가 구성 요소 인스턴스를 참조합니다. 예를 들어, 다음 코드를 myButton이라는 Button 구성 요소 인스턴스에 바로 첨부하면 출력 패널에 “\_level0.myButton”이 표시됩니다.

```

on(click){
    trace(this);
}

```

### on() 이벤트 핸들러를 사용하려면:

1. 사용자 인터페이스 구성 요소를 스테이지로 드래그합니다.  
예를 들어, Button 구성 요소를 스테이지로 드래그합니다.
2. 스테이지에서 구성 요소를 선택하고 액션 패널을 엽니다.
3. on() 핸들러를 액션 패널에 다음과 같이 추가합니다.

```

on(event){
    //your statements go here
}

```

예를 들면 다음과 같습니다.

```

on(click){
    trace(this);
}

```

on() 핸들러에 대한 이벤트가 발생하면(이 경우, 버튼 클릭) Flash에서 on() 핸들러 내부의 코드를 실행합니다.

4. 컨트롤 > 무비 테스트를 선택하고 버튼을 클릭하여 출력을 확인합니다.

구성 요소를 다른 응용 프로그램에서 사용할 경우에는 구성 요소 모양을 변경할 수 있습니다. 다음 세 가지 방법을 각각 또는 함께 사용하여 구성 요소의 모양을 사용자 정의할 수 있습니다.

**스타일** 사용자 인터페이스(UI) 구성 요소는 구성 요소의 일부 모양을 설정하는 스타일 속성을 가지고 있습니다. 각 구성 요소마다 수정 가능한 스타일 속성 집합을 가지고 있으며, 구성 요소의 일부 시각적 특성은 스타일 설정으로 변경할 수 없습니다. 자세한 내용은 [80페이지의 “스타일을 사용하여 구성 요소 색상 및 텍스트 사용자 정의”](#)를 참조하십시오.

**스킨** 스킨은 구성 요소의 그래픽 표시를 구성하는 심볼 모음으로 구성되어 있습니다. 스킨 넣은 구성 요소의 소스 그래픽을 수정하거나 바꾸어 구성 요소의 모양을 변경하는 과정입니다. 스킨은 테두리의 가장자리나 모서리 같은 작은 부분 또는 업 상태(버튼을 누르지 않은 상태)의 버튼 모양과 같은 복잡한 부분이 될 수 있습니다. 또한 스킨은 구성 요소의 특정 부분을 그리기 위한 코드를 포함하는 그래픽 없는 심볼일 수도 있습니다. 스타일 속성으로 설정할 수 없는 구성 요소의 일부 특성은 스킨을 수정함으로써 설정할 수 있습니다. 자세한 내용은 [93페이지의 “구성 요소 스킨링”](#)을 참조하십시오.

**테마** 테마는 FLA 파일로 저장하고 다른 문서에 적용할 수 있는 스타일 및 스킨 모음입니다. 자세한 내용은 [104페이지의 “테마”](#)를 참조하십시오.

이 장에서 설명하는 단원은 다음과 같습니다.

<a href="#">스타일을 사용하여 구성 요소 색상 및 텍스트 사용자 정의</a> .....	80
<a href="#">구성 요소 스킨링</a> .....	93
<a href="#">테마</a> .....	104
<a href="#">스킨링 및 스타일을 통합하여 구성 요소 사용자 정의</a> .....	115

# 스타일을 사용하여 구성 요소 색상 및 텍스트 사용자 정의

Flash에서는 모든 UI 구성 요소에 대해 편집 가능한 스타일 속성이 제공됩니다. 설명서의 각 특정 구성 요소에 대한 설명 부분에서 해당 구성 요소의 수정 가능한 스타일 목록 표를 볼 수 있습니다(예를 들어, *ActionScript 2.0 구성 요소 언어 참조 설명서*의 “Accordion 구성 요소에 스타일 사용”에서는 Accordion 구성 요소에 대한 스타일 표를 볼 수 있습니다). 또한 UI 구성 요소는 UIObject 클래스로부터 `setStyle()` 및 `getStyle()` 메서드를 상속합니다(*ActionScript 2.0 구성 요소 언어 참조 설명서*의 “UIObject.setStyle()” 및 “UIObject.getStyle()” 참조). 82페이지의 “구성 요소 인스턴스에 스타일 설정”에 설명된 것처럼 구성 요소 인스턴스에 대해 `setStyle()` 및 `getStyle()` 메서드를 사용하여 스타일 속성 값을 설정하고 가져올 수 있습니다.



미디어 구성 요소에 대해서는 스타일을 설정할 수 없습니다.

## 스타일 선언 및 테마 사용

넓게 보면 스타일은 여러 구성 요소 인스턴스에서 스타일 속성 값을 제어할 수 있는 스타일 선언 내에서 구성됩니다. 스타일 선언은 `CSSStyleDeclaration` 클래스에 의해 만들어지는 객체이며 그 속성은 구성 요소에 대해 지정할 수 있는 스타일 설정입니다. ActionScript의 스타일 선언은 CSS(“Cascading Style Sheets”)가 HTML 페이지가 적용된 후에 모델링됩니다. HTML 페이지의 경우, HTML 페이지 그룹의 내용에 대해 스타일 속성을 정의하는 스타일 시트 파일을 만들 수 있습니다. 구성 요소를 사용하여 스타일 선언 객체를 만들고 해당 스타일 선언 객체에 스타일 속성을 추가하여 Flash 문서의 구성 요소 모양을 제어할 수 있습니다. 또한 스타일 선언은 **테마** 내에서 구성됩니다. Flash에서는 구성 요소에 대해 두 가지 시각적 테마, 즉 Halo(HaloTheme.fla) 및 Sample(SampleTheme.fla)을 제공합니다. **테마**는 문서에서 구성 요소의 모양을 제어하는 일련의 스타일과 그래픽입니다. 각 테마는 구성 요소에 스타일을 제공합니다. 문서에서 어떤 테마를 사용하느냐에 따라 각 구성 요소에서 사용되는 스타일이 부분적으로 결정됩니다. `defaultIcon`과 같은 일부 스타일은 문서에 적용된 테마와 상관없이 연결된 구성 요소에 사용됩니다. `themeColor` 및 `symbolBackgroundColor`와 같은 기타 스타일은 해당 테마가 사용 중인 경우에만 구성 요소에 사용됩니다. 예를 들어, `themeColor`는 Halo 테마가 사용 중인 경우에만 사용되고 `symbolBackgroundColor`는 Sample 테마가 사용 중인 경우에만 사용됩니다. 구성 요소에 대해 어떤 스타일 속성을 설정할 수 있는지 확인하려면 해당 구성 요소에 할당되는 테마를 알아야 합니다. *ActionScript 2.0 구성 요소 언어 참조 설명서*에 있는 각 구성 요소에 대한 스타일 표는 각 스타일 속성이 제공된 테마 중 하나에만 적용되는지 둘 다에 적용되는지 나타냅니다. 자세한 내용은 104페이지의 “테마”를 참조하십시오.

## 스타일 설정 이해

스타일과 스타일 선언을 사용하다 보면 다양한 방법으로 전역, 테마, 클래스, 스타일 선언 또는 스타일 속성별로 스타일을 설정할 수 있게 됩니다. 일부 스타일 속성은 부모 구성 요소로부터 상속될 수 있습니다. 예를 들어, **Accordion** 자식 패널은 **Accordion** 구성 요소로부터 글꼴 처리를 상속받습니다. 다음은 스타일 비헤이비어에 대한 몇 가지 주요 사항입니다.

**테마 종속성** 특정 구성 요소에 대해 설정할 수 있는 스타일 속성은 현재 테마에 의해 결정됩니다. 기본적으로 Flash 구성 요소는 Halo 테마를 사용하도록 설계되었지만 Flash에서는 Sample 테마도 제공됩니다. 따라서 *ActionScript 2.0 구성 요소 언어 참조 설명서*의 “**Button** 구성 요소에 스타일 사용”에 있는 **Button** 구성 요소에 대한 것과 같은 스타일 속성 표를 읽을 때는 자신이 원하는 스타일을 지원하는 테마가 어떤 것인지 확인하십시오. 이 표에서는 Halo, Sample 또는 둘 다(두 테마 모두 스타일 속성 지원)로 표시됩니다. 현재 테마를 변경하는 방법은 [105페이지의 “테마 전환”](#)을 참조하십시오.

**상속** ActionScript 내에서는 상속을 설정할 수 없습니다. 자식 구성 요소는 부모 구성 요소로부터 스타일을 상속받거나 상속받지 않도록 설계되어 있습니다.

**전역 스타일 시트** Flash의 스타일 선언은 HTML 문서에서 CSS가 지원되듯 Flash 문서에 대해 “캐스케이딩”을 지원하지 않습니다. 모든 스타일 시트 선언 객체는 응용 프로그램(전역) 레벨에서 정의됩니다.

**우선 순위** 두 가지 이상의 방법으로 구성 요소 스타일이 설정된 경우(예를 들어, `textColor`가 전역 레벨과 구성 요소 인스턴스 레벨에서 설정된 경우) Flash에서는 [89페이지의 “같은 문서 안에 전역, 사용자 정의 및 클래스 스타일 사용”](#)에 나열된 순서에 따라 처음 나타나는 스타일이 사용됩니다.

## 스타일 설정

스타일 속성의 존재, 스타일 선언 내에서의 스타일 속성 구조, 테마로 더 넓게 구성된 스타일 선언 및 그래픽의 구조를 통해 다음과 같은 방법으로 구성 요소를 사용자 정의할 수 있습니다.

- 구성 요소 인스턴스에 스타일을 설정합니다.  
단일 구성 요소 인스턴스의 색상과 텍스트 속성을 변경할 수 있습니다. 특정 상황에서는 이 방법이 효율적일 수 있지만 문서 내에 있는 모든 구성 요소에 대해 속성을 개별적으로 설정하려면 너무 많은 시간이 소요될 수 있습니다.  
자세한 내용은 [82페이지의 “구성 요소 인스턴스에 스타일 설정”](#)을 참조하십시오.
- 문서 내의 모든 구성 요소에 대한 스타일을 설정하는 전역 스타일 선언을 조정합니다.  
전역 스타일 선언에서 스타일을 만들면 전체 문서에 일관된 모양을 적용할 수 있습니다.  
자세한 내용은 [84페이지의 “전역 스타일 설정”](#)을 참조하십시오.

- 사용자 정의 스타일 선언을 만들어 여러 구성 요소 인스턴스에 적용합니다.  
문서의 특정 구성 요소에서 같은 스타일을 공유하도록 하려면 사용자 정의 스타일 선언을 만들어 해당 구성 요소에 적용하면 됩니다.  
자세한 내용은 [85페이지의 “특정 구성 요소의 사용자 정의 스타일 설정”](#)을 참조하십시오.
  - 기본 클래스 스타일 선언을 만듭니다.  
클래스의 모든 인스턴스가 기본 모양을 공유하도록 기본 클래스 스타일 선언을 정의할 수도 있습니다.  
자세한 내용은 [87페이지의 “구성 요소 클래스의 스타일 설정”](#)을 참조하십시오.
  - 상속되는 스타일을 사용하여 문서의 일부에 있는 구성 요소에 대한 스타일을 설정합니다.  
포함된 구성 요소가 컨테이너에 설정된 스타일 속성의 값을 상속합니다.  
자세한 내용은 [87페이지의 “컨테이너에 상속되는 스타일 설정”](#)을 참조하십시오.
- Flash에서는 실시간 미리 보기 기능을 사용하여 스테이지에서 구성 요소를 볼 때 스타일 속성의 변경 내용이 표시되지 않습니다. 자세한 내용은 [60페이지의 “실시간 미리 보기의 구성 요소”](#)를 참조하십시오.

## 구성 요소 인스턴스에 스타일 설정

스타일 속성을 설정하고 가져오는 `ActionScript` 코드를 모든 구성 요소 인스턴스에 대해 작성할 수 있습니다. “`UIObject.setStyle()`” 및 “`UIObject.getStyle()`” 메서드는 모든 UI 구성 요소에서 직접 호출할 수 있습니다. 다음 구문은 구성 요소 인스턴스의 속성과 값을 지정합니다.

```
instanceName.setStyle("propertyName", value);
```

예를 들어, 다음 코드에서는 Halo 테마를 사용하는 `myButton`이라는 `Button` 인스턴스에 강조 색상을 설정합니다.

```
myButton.setStyle("themeColor", "haloBlue");
```

**애크**

값이 문자열인 경우에는 따옴표로 묶어야 합니다.

속성을 사용하여 스타일에 직접 액세스할 수도 있습니다(예: `myButton.color = 0xFF00FF`).

setStyle()을 통해 구성 요소 인스턴스에 설정된 스타일 속성은 우선 순위가 가장 높으며 스타일 선언이나 테마를 기반으로 하는 다른 모든 스타일 설정을 무시합니다. 그러나 단일 구성 요소 인스턴스에 대해 setStyle()을 사용하여 설정한 속성이 많을수록 런타임 시 구성 요소의 렌더링 속도가 더 느려집니다. “UIObject.createClassObject()”를 사용하여 구성 요소 인스턴스가 만들어지는 동안 스타일 속성을 정의하는 *ActionScript*를 사용하고 *initObject* 매개 변수에 스타일 설정을 배치함으로써 사용자 정의 구성 요소의 렌더링 속도를 높일 수 있습니다. 예를 들어, 현재 문서 라이브러리의 *ComboBox* 구성 요소에 대해 다음 코드를 사용하면 *my\_cb*라는 콤보 상자 인스턴스가 만들어지고 콤보 상자의 텍스트가 기울임체와 오른쪽 정렬로 설정됩니다.

```
createClassObject(mx.controls.ComboBox, "my_cb", 1, {fontStyle:"italic",
    textAlign:"right"});
my_cb.addItem({data:1, label:"One"});
```

**예제** 속성을 여러 개 변경하거나 여러 구성 요소 인스턴스의 속성을 변경하려면 사용자 정의 스타일을 만들면 됩니다. 구성 요소 인스턴스에서 여러 속성을 변경하기 위해 사용자 정의 스타일을 사용하면 setStyle() 호출을 여러 번 사용하는 경우보다 더 빨리 렌더링됩니다. 자세한 내용은 [85페이지의 “특정 구성 요소의 사용자 정의 스타일 설정”](#)을 참조하십시오.

### Halo 테마를 사용하는 단일 구성 요소 인스턴스의 속성을 설정하거나 변경하려면:

1. 스테이지에서 구성 요소 인스턴스를 선택합니다.
2. 속성 관리자에서 인스턴스 이름을 **myComponent**로 지정합니다.
3. 액션 패널을 열고 **장면 1**을 선택한 후 **레이어 1: 프레임 1**을 선택합니다.
4. 다음 코드를 입력하여 인스턴스를 주황색으로 변경합니다.  
`myComponent.setStyle("themeColor", "haloOrange");`
5. **컨트롤 > 무비 테스트**를 선택하여 변경 내용을 확인합니다.  
 특정 구성 요소에서 지원하는 스타일 목록은 *ActionScript 2.0 구성 요소 언어 참조 설명서*에서 해당 구성 요소 항목을 참조하십시오.

## ActionScript를 사용하여 구성 요소 인스턴스를 만들고 여러 속성을 동시에 설정하려면:

1. 구성 요소를 라이브러리로 드래그합니다.
2. 액션 패널을 열고 **장면 1**을 선택한 후 **레이어 1: 프레임 1**을 선택합니다.
3. 다음 구문을 입력하여 구성 요소의 인스턴스를 만들고 속성을 설정합니다.

```
createClassObject(className, "instance_name", depth, {style:"setting", style:"setting"});
```

예를 들어, 라이브러리의 **Button** 구성 요소에 대해 다음 ActionScript를 사용하면 텍스트 스타일이 자주색과 기울임체로 설정된 깊이 1의 my\_button 버튼 인스턴스가 만들어집니다.

```
createClassObject(mx.controls.Button, "my_button", 1, {label:"Hello", color:"0x9900CC", fontStyle:"italic"});
```

자세한 내용은 “UIObject.createClassObject()”를 참조하십시오.

4. **컨트롤 > 무비 테스트**를 선택하여 변경 내용을 확인합니다.

특정 구성 요소에서 지원하는 스타일 목록은 *ActionScript 2.0 구성 요소 언어 참조 설명서*에서 해당 구성 요소 항목을 참조하십시오.

## 전역 스타일 설정

기본적으로 모든 구성 요소는 다른 스타일 선언이 구성 요소에 추가되기 전까지 전역 스타일 선언을 따릅니다(85페이지의 “특정 구성 요소의 사용자 정의 스타일 설정” 참조). 전역 스타일 선언은 Adobe Component Architecture 버전 2로 만든 모든 Flash 구성 요소에 할당됩니다. `_global` 객체에는 `CSSStyleDeclaration`의 인스턴스이며 전역 스타일 선언의 역할을 하는 `style` 속성(`_global.style`)이 있습니다. 전역 스타일 선언에서 스타일 속성 값을 변경하면 Flash 문서의 모든 구성 요소에 해당 변경 내용이 적용됩니다.

**개요**

TextArea 및 TextInput 구성 요소의 `backgroundColor` 스타일과 같은 일부 스타일은 구성 요소 클래스의 `CSSStyleDeclaration` 인스턴스에 설정됩니다. 스타일 값을 결정할 때 클래스 스타일 선언이 전역 스타일 선언보다 우선하기 때문에 전역 스타일 선언에 `backgroundColor`를 설정해도 TextArea 및 TextInput 구성 요소에 아무런 영향을 주지 않습니다. 스타일 우선 순위에 대한 자세한 내용은 89페이지의 “같은 문서 안에 전역, 사용자 정의 및 클래스 스타일 사용”을 참조하십시오. 구성 요소 클래스의 `CSSStyleDeclaration`을 편집하는 방법에 대한 자세한 내용은 87페이지의 “구성 요소 클래스의 스타일 설정”을 참조하십시오.

### 전역 스타일 선언에서 하나 이상의 속성을 변경하려면:

1. 문서에 하나 이상의 구성 요소 인스턴스가 포함되어 있는지 확인합니다.  
자세한 내용은 [48페이지의 “Flash 문서에 구성 요소 추가”](#)를 참조하십시오.
2. 타임라인에서 구성 요소가 나타나는 프레임이나 그 앞의 프레임을 선택합니다.
3. 액션 패널에서 다음과 같은 코드를 사용하여 전역 스타일 선언의 속성을 변경합니다.  
다음과 같이 값을 변경할 속성만 나열해야 합니다.

```
_global.style.setStyle("color", 0xCC6699);
_global.style.setStyle("themeColor", "haloBlue");
_global.style.setStyle("fontSize", 16);
_global.style.setStyle("fontFamily", "_serif");
```
4. 컨트롤 > 무비 테스트를 선택하여 변경 내용을 확인합니다.

## 특정 구성 요소의 사용자 정의 스타일 설정

사용자 정의 스타일 선언을 만들어 Flash 문서의 특정 구성 요소에 대해 일련의 고유한 속성을 지정할 수 있습니다. [84페이지의 “전역 스타일 설정”](#)에서 설명된 것처럼 `_global` 객체의 `style` 속성은 전체 Flash 문서에 대한 기본 스타일 선언을 결정합니다. 또한 `_global` 객체에는 사용 가능한 사용자 정의 스타일 선언 목록인 `styles` 속성도 있습니다. 따라서 스타일 선언을 `CSSStyleDeclaration` 객체의 새 인스턴스로 만들고 사용자 정의 스타일 이름을 지정한 다음 이를 `_global.styles` 목록에 배치할 수 있습니다. 그런 다음 스타일의 속성과 값을 지정하고, 동일한 모습을 공유해야 하는 구성 요소 인스턴스에 스타일 이름을 지정할 수 있습니다.

구성 요소 인스턴스에 스타일 이름을 지정할 경우, 해당 구성 요소는 자신이 지원하는 스타일 속성에 대해서만 응답한다는 점에 유의하십시오. 각 구성 요소가 지원하는 스타일 속성 목록을 보려면 *ActionScript 2.0 구성 요소 언어 참조 설명서*에서 개별 구성 요소 항목을 참조하십시오.

사용자 정의 스타일 포맷을 변경하려면 다음 구문을 사용합니다.

```
_global.styles.CustomStyleName.setStyle(propertyName, propertyValue);
```

사용자 정의 스타일 설정은 클래스 스타일 설정, 상속된 스타일 설정 및 전역 스타일 설정보다 우선합니다. 스타일 우선 순위 목록을 보려면 [89페이지의 “같은 문서 안에 전역, 사용자 정의 및 클래스 스타일 사용”](#)을 참조하십시오.

### 특정 구성 요소의 사용자 정의 스타일 선언을 만들려면:

1. 스테이지에 하나 이상의 구성 요소를 추가합니다.  
자세한 내용은 [48페이지의 “Flash 문서에 구성 요소 추가”](#)를 참조하십시오.  
이 예제에서는 인스턴스 이름이 각각 a, b 및 c인 `Button` 구성 요소를 사용합니다. 다른 구성 요소를 사용하는 경우에는 속성 관리자에서 인스턴스 이름을 지정한 다음 8단계에 해당 인스턴스 이름을 사용하십시오.

2. 타임라인에서 구성 요소가 나타나는 프레임이나 그 앞의 프레임을 선택합니다.

3. 액션 패널을 엽니다.

4. `CSSStyleDeclaration` 클래스 내에서 새 스타일 선언을 만들기 위한 생성자 함수에 액세스할 수 있도록 다음의 `import` 문을 추가합니다.

```
import mx.styles.CSSStyleDeclaration;
```

5. 다음 구문을 사용하여 `CSSStyleDeclaration` 객체의 인스턴스를 만들어 새 사용자 정의 스타일 포맷을 정의합니다.

```
var new_style:Object = new CSSStyleDeclaration();
```

6. 사용자 정의 스타일 선언의 `_global.styles` 목록에서 스타일 선언에 “`myStyle`”과 같은 이름을 지정하고, 새 스타일 선언에 대한 모든 속성을 포함하는 객체를 식별합니다.

```
_global.styles.myStyle = new_style;
```

7. `UIObject` 클래스에서 상속된 `setStyle()` 메서드를 사용하여 `new_style` 객체에 새 속성을 추가합니다. 그러면 이 객체는 사용자 정의 스타일 선언 `myStyle`과 연관됩니다.

```
new_style.setStyle("fontFamily", "_serif");
new_style.setStyle("fontSize", 14);
new_style.setStyle("fontWeight", "bold");
new_style.setStyle("textDecoration", "underline");
new_style.setStyle("color", 0x666699);
```

8. 동일한 스크립트 창에서 다음 구문을 사용하여 특정 구성 요소 세 개의 `styleName` 속성을 사용자 정의 스타일 선언 이름으로 설정합니다.

```
a.setStyle("styleName", "myStyle");
b.setStyle("styleName", "myStyle");
c.setStyle("styleName", "myStyle");
```

또한 선언의 전역 `styleName` 속성을 통해 `setStyle()`과 `getStyle()` 메서드를 사용하여 사용자 정의 스타일 선언의 스타일에 액세스할 수 있습니다. 예를 들어, 다음 코드에서는 `myStyle` 스타일 선언에 `backgroundColor` 스타일을 설정합니다.

```
_global.styles.myStyle.setStyle("themeColor", "haloOrange");
```

그러나 5단계 및 6단계에서 `new_style` 인스턴스와 스타일 선언을 연관시켰으므로 `new_style.setStyle("themeColor", "haloOrange")`같은 더 짧은 구문을 사용할 수 있습니다.

`setStyle()` 및 `getStyle()` 메서드에 대한 자세한 내용은 *ActionScript 2.0 구성 요소 언어 참조 설명서*의 “`UIObject.setStyle()`” 및 “`UIObject.getStyle()`”을 참조하십시오.

## 구성 요소 클래스의 스타일 설정

Button, CheckBox 등 모든 구성 요소 클래스에 대해 해당 클래스의 각 인스턴스에 기본 스타일을 설정하는 클래스 스타일 선언을 정의할 수 있습니다. 이러한 스타일 선언은 인스턴스를 만들기 전에 만들어야 합니다. TextArea 및 TextInput와 같이 borderStyle과 backgroundColor 속성을 사용자 정의해야 할 일부 구성 요소에는 클래스 스타일 선언이 기본적으로 미리 정의되어 있습니다.

개  
요

클래스 스타일 시트를 바꿀 경우 이전 스타일 시트에서 원하는 스타일을 추가해야 합니다. 그렇지 않으면 스타일을 덮어씁니다.

다음 코드는 먼저 현재 테마에 이미 CheckBox에 대한 스타일 선언이 있는지 확인하고, 없는 경우 새 스타일 선언을 만듭니다. 그런 다음 setStyle() 메서드를 사용하여 CheckBox 스타일 선언에 대한 스타일 속성을 정의합니다(이 경우 “color”는 모든 체크 상자 레이블 텍스트 색을 파랑으로 설정합니다).

```
if (_global.styles.CheckBox == undefined) {
    _global.styles.CheckBox = new mx.styles.CSSStyleDeclaration();
}
_global.styles.CheckBox.setStyle("color", 0x0000FF);
```

CheckBox 구성 요소에 대해 설정할 수 있는 스타일 속성 표는 *ActionScript 2.0 구성 요소 언어 참조 설명서*의 “CheckBox 구성 요소에 스타일 사용”을 참조하십시오.

사용자 정의 스타일 설정은 상속된 스타일 설정과 전역 스타일 설정보다 우선합니다. 스타일 우선 순위 목록을 보려면 [89페이지의 “같은 문서 안에 전역, 사용자 정의 및 클래스 스타일 사용”](#)을 참조하십시오.

## 컨테이너에 상속되는 스타일 설정

상속되는 스타일은 문서의 MovieClip 계층에 있는 부모 구성 요소에서 값이 상속되는 스타일입니다. 인스턴스, 사용자 정의 또는 클래스 레벨에서 텍스트 또는 색상 스타일을 설정하지 않으면 스타일 값이 MovieClip 계층에서 검색됩니다. 따라서 컨테이너 구성 요소에 스타일을 설정하면 포함된 구성 요소가 이러한 스타일 설정을 상속합니다.

다음은 상속되는 스타일입니다.

- fontFamily
- fontSize
- fontStyle
- fontWeight
- textAlign
- textIndent

- 모든 단일 값 색상 스타일. 예를 들어, themeColor는 상속되는 스타일이지만 alternatingRowColors는 상속되는 스타일이 아닙니다.

스타일 관리자가 Flash에 스타일이 값을 상속하는지 알립니다. 런타임에 다른 스타일을 상속되는 스타일로 추가할 수도 있습니다. 자세한 내용은 *ActionScript 2.0 구성 요소 언어 참조 설명서*의 “StyleManager 클래스”를 참조하십시오.

**예제**

Flash 구성 요소에서 스타일이 구현되는 방식과 HTML 페이지에서 CSS가 구현되는 방식 사이의 한 가지 중요한 차이점은 Flash 구성 요소에서는 CSS inherit 값이 지원되지 않는다는 점입니다. 구성 요소 디자인에 의해 스타일의 상속 여부가 결정됩니다.

상속된 스타일은 전역 스타일보다 우선합니다. 스타일 우선 순위 목록을 보려면 [89페이지의 “같은 문서 안에 전역, 사용자 정의 및 클래스 스타일 사용”](#)을 참조하십시오.

다음 예제는 Flash에서 제공되는 Accordion 구성 요소에서 상속 스타일을 사용하는 방법을 보여줍니다.

### 개별 Accordion 창에서 구성 요소가 상속하는 스타일이 있는 Accordion 구성 요소를 만들려면:

1. 새 FLA 파일을 엽니다.
2. 구성 요소 패널에서 스테이지로 Accordion 구성 요소를 드래그합니다.
3. 속성 관리자를 사용하여 Accordion 구성 요소의 이름과 크기를 지정합니다. 이 예제에서는 구성 요소 인스턴스 이름을 **Accordion**으로 지정합니다.
4. 구성 요소 패널에서 라이브러리로 TextInput 구성 요소와 Button 구성 요소를 드래그합니다. 구성 요소를 라이브러리로 드래그하면 런타임에 스크립트에서 이를 사용할 수 있습니다.
5. 타임라인의 첫 번째 프레임에 다음 ActionScript를 추가합니다.

```
var section1 = accordion.createChild(mx.core.View, "section1", {label:
    "First Section"});
var section2 = accordion.createChild(mx.core.View, "section2", {label:
    "Second Section"});

var input1 = section1.createChild(mx.controls.TextInput, "input1");
var button1 = section1.createChild(mx.controls.Button, "button1");

input1.text = "Text Input";
button1.label = "Button";
button1.move(0, input1.height + 10);

var input2 = section2.createChild(mx.controls.TextInput, "input2");
var button2 = section2.createChild(mx.controls.Button, "button2");

input2.text = "Text Input";
button2.label = "Button";
button2.move(0, input2.height + 10);
```

위의 코드에서는 Accordion 구성 요소에 두 개의 자식을 추가하고 이 예제에서 스타일 상속을 보여 주는 데 사용한 TextInput 및 Button 컨트롤과 함께 각 자식을 로드합니다.

6. 컨트롤 > 무비 테스트를 선택하여 스타일 상속을 추가하기 전에 문서를 확인합니다.

7. 첫 번째 프레임의 스크립트 끝에 다음 ActionScript를 추가합니다.

```
accordion.setStyle("fontStyle", "italic");
```

8. 컨트롤 > 무비 테스트를 선택하여 변경 내용을 확인합니다.

Accordion 구성 요소의 fontStyle 설정은 Accordion 텍스트 자체에 영향을 미칠 뿐 아니라 Accordion 텍스트 내의 TextInput 및 Button 구성 요소와 연결된 텍스트에도 영향을 미칩니다.

## 같은 문서 안에 전역, 사용자 정의 및 클래스 스타일 사용

문서의 한 부분에만 스타일을 정의하면 Flash에서는 해당 정의를 사용하여 필요할 때 속성 값을 확인합니다. 하지만 Flash 문서 하나에 구성 요소 인스턴스에 직접 설정된 스타일 속성, 사용자 정의 스타일 선언, 기본 클래스 스타일 선언, 상속되는 스타일, 전역 스타일 선언 등의 다양한 스타일 설정을 사용할 수 있습니다. 이 경우, Flash에서는 속성 값을 확인하기 위해 특정 순서대로 전체를 검색합니다.

Flash에서는 값을 발견할 때까지 다음 순서로 스타일을 찾습니다.

1. 구성 요소 인스턴스에서 스타일 속성을 찾습니다.
2. 인스턴스의 styleName 속성을 찾아서 사용자 정의 스타일 선언이 할당되어 있는지 확인합니다.
3. 기본 클래스 스타일 선언에서 속성을 찾습니다.
4. 스타일이 상속되는 스타일에 속하면 부모 계층에서 상속된 값을 찾습니다.
5. 전역 스타일 선언에서 스타일을 찾습니다.
6. 아직 속성이 정의되지 않은 경우 속성의 값은 undefined입니다.

## 색상 스타일 속성

색상 스타일 속성은 색상 이외의 속성과는 다릅니다. backgroundColor, disabledColor, color 등과 같이 모든 색상 속성의 이름은 "Color"로 끝납니다. 색상 스타일 속성을 변경하면 인스턴스 및 적절한 모든 자식 인스턴스에서 색상이 즉시 변경됩니다. 반면 다른 모든 스타일 속성의 경우 속성을 변경하면 객체를 다시 그려야 한다는 사실만 표시되고 다음 프레임에서 변경 내용이 적용됩니다.

색상 스타일 속성 값은 숫자, 문자열 또는 객체가 될 수 있습니다. 숫자 값은 색상의 RGB 값을 16진수(0xRRGGBB)로 나타냅니다. 속성 값이 문자열인 경우에는 색상 이름만 사용할 수 있습니다.

색상 이름은 일반적으로 사용되는 색상에 해당하는 문자열입니다. 스타일 관리자(*ActionScript 2.0 구성 요소 언어 참조 설명서*의 `StyleManager` 클래스 참조)를 사용하여 새 색상 이름을 추가할 수 있습니다. 다음 표에서는 기본 색상 이름을 보여 줍니다.

색상 이름	값
검정	0x000000
흰색	0xFFFFFFFF
빨강	0xFF0000
녹색	0x00FF00
파랑	0x0000FF
마젠타	0xFF00FF
노랑	0xFFFF00
녹청	0x00FFFF
밝은 녹색	0x80FF4D
밝은 파랑	0x2BF5F5
밝은 주황	0xFFC200



색상 이름이 정의되어 있지 않으면 구성 요소가 제대로 그려지지 않을 수 있습니다.

유효한 `ActionScript` 식별자를 사용하여 자체 색상 이름(예: `"WindowText"` 또는 `"ButtonText"`)을 만들 수 있습니다. 다음과 같이 스타일 관리자를 사용하여 새 색상을 정의합니다.

```
mx.styles.StyleManager.registerColorName("special_blue", 0x0066ff);
```

대부분의 구성 요소는 객체를 색상 스타일 속성 값으로 처리하지 못하지만 일부 구성 요소는 그라디언트나 기타 색상 조합을 나타내는 색상 객체를 처리할 수 있습니다. 자세한 내용은 *ActionScript 2.0 구성 요소 언어 참조 설명서*에서 각 구성 요소 항목의 “스타일 사용” 단원을 참조하십시오.

클래스 스타일 선언과 색상 이름을 사용하면 스크린에 있는 텍스트 및 심볼의 색상을 쉽게 제어할 수 있습니다. 예를 들어, `Microsoft Windows`와 유사한 모양의 디스플레이 구성 스크린을 만들려면 `ButtonText`, `WindowText`와 같은 색상 이름과 `Button`, `CheckBox`, `Window`와 같은 클래스 스타일 선언을 정의하면 됩니다.



일부 구성 요소는 `alternatingRowColors`와 같이 색상이 배열된 스타일 속성을 제공합니다. 이러한 스타일은 색상 이름 대신에 숫자 RGB 값을 배열해서 설정해야 합니다.

## 구성 요소 애니메이션 사용자 정의

Accordion, ComboBox 및 Tree 구성 요소와 같은 여러 구성 요소는 Accordion 자식 간에 전환하거나 ComboBox 드롭다운 목록을 확장하거나 Tree 폴더를 확장/축소하는 경우와 같이 구성 요소 상태를 전환할 때 그 전환을 보여 주는 애니메이션을 제공합니다. 또한 구성 요소는 목록에 있는 행과 같은 항목의 선택 및 선택 취소와 관련된 애니메이션도 제공합니다.

다음 스타일을 통해 이러한 애니메이션의 요소를 제어할 수 있습니다.

애니메이션 스타일	설명
openDuration	Accordion, ComboBox 및 Tree 구성 요소에서 여유 열기 상태로 전환하는 데 필요한 지속 기간(밀리초)입니다. 기본값은 250입니다.
openEasing	Accordion, ComboBox 및 Tree 구성 요소에서 상태 애니메이션을 제어하는 트위닝 함수에 대한 참조입니다. 기본 수식은 사인 인/아웃 공식을 사용합니다.
popupDuration	Menu 구성 요소에서 메뉴 열기 상태로 전환하는 데 필요한 지속 기간(밀리초)입니다. 기본값은 150이지만 애니메이션은 항상 기본 사인 인/아웃 수식을 사용합니다.
selectionDuration	ComboBox, DataGrid, List 및 Tree 구성 요소에서 일반 상태에서 선택한 상태로 전환하거나 선택한 상태에서 일반 상태로 전환하는 데 필요한 지속 기간(밀리초)입니다. 기본값은 200입니다.
selectionEasing	ComboBox, DataGrid, List 및 Tree 구성 요소에서 선택 애니메이션을 제어하는 트위닝 함수에 대한 참조입니다. 이 스타일은 일반 상태에서 선택한 상태로의 전환에 대해서만 적용됩니다. 기본 수식은 사인 인/아웃 공식을 사용합니다.

mx.transitions.easing 패키지는 다음과 같이 여유 값을 제어하는 여섯 개의 클래스를 제공합니다.

여유 클래스	설명
Back	한 번에 한쪽 끝이나 양쪽 끝에서 전환 범위를 넘어가서 약간 넘치는 효과를 제공합니다.
Bounce	한쪽 끝이나 양쪽 끝에서 전환 범위 내에서 전체적으로 공이 튀는 효과를 제공합니다. 튀는 횟수는 지속 기간과 관련이 있습니다. 즉, 지속 기간이 길수록 여러 번 튕니다.
Elastic	한쪽 끝이나 양쪽 끝에서 전환 범위를 벗어나는 탄성 효과를 제공합니다. 탄성의 정도는 지속 기간의 영향을 받지 않습니다.
None	속도의 가감 효과 없이 처음부터 끝까지 같은 속도로 이동하는 움직임을 제공합니다. 일반적으로 이러한 전환을 선형 전환이라고도 합니다.

여유 클래스	설명
Regular	한쪽 끝이나 양쪽 끝에서 느린 움직임을 제공하여 가속 효과, 감속 효과 또는 두 효과를 모두 냅니다.
Strong	한쪽 끝이나 양쪽 끝에서 훨씬 더 느린 움직임을 제공합니다. 이 효과는 일반 효과와 비슷하지만 더 뚜렷합니다.

`mx.transitions.easing` 패키지의 각 클래스는 다음과 같은 세 가지 여유 메서드를 제공합니다.

여유 메서드	설명
<code>easeIn</code>	전환 시작 부분에서 여유 효과를 제공합니다.
<code>easeOut</code>	전환 끝 부분에서 여유 효과를 제공합니다.
<code>easeInOut</code>	전환 시작 부분과 끝 부분에서 여유 효과를 제공합니다.

여유 메서드가 여유 클래스의 정적 메서드이므로 여유 클래스를 인스턴스화하지 않아도 됩니다. 다음 예제와 같이 `setStyle()`을 호출할 때 메서드를 사용합니다.

```
import mx.transitions.easing.*;
trace("_global.styles.Accordion = " + _global.styles.Accordion);
_global.styles.Accordion.setStyle("openDuration", 1500);
_global.styles.Accordion.setStyle("openEasing", Bounce.easeOut);
```

**예제** 모든 전환에 사용된 기본 수식은 위에 나열된 여유 클래스에서 사용할 수 없습니다. 다른 여유 메서드가 지정된 후 구성 요소에서 기본 여유 메서드를 사용하도록 지정하려면 `setStyle("openEasing", null)`을 호출합니다.

자세한 내용은 *ActionScript 2.0 구성 요소 언어 참조 설명서*의 “구성 요소에 여유 메서드 적용”을 참조하십시오.

## 스타일 속성 값 가져오기

스타일 속성 값을 가져오려면 *ActionScript 2.0 구성 요소 언어 참조 설명서*의 “`UIObject.getStyle()`”을 사용합니다. `Media` 구성 요소를 제외한 모든 버전 2 구성 요소가 포함된 `UIObject`의 하위 클래스인 모든 구성 요소는 `getStyle()` 메서드를 상속합니다. 즉, 구성 요소 인스턴스에서 `setStyle()`을 호출할 수 있는 것처럼 구성 요소 인스턴스에서 `getStyle()`을 호출할 수 있습니다.

다음 코드에서는 `themeColor` 스타일의 값을 가져와서 `oldStyle` 변수에 할당합니다.

```
var myCheckBox:mx.controls.CheckBox;
var oldFontSize:Number

oldFontSize = myCheckBox.getStyle("fontSize");
trace(oldFontSize);
```

# 구성 요소 스키닝

스킨은 구성 요소의 모양을 표시하는 데 사용되는 무비 클립 심볼로, 대부분의 스킨에는 구성 요소를 나타내는 여러 가지 모양이 들어 있지만 일부 스킨에는 문서에 구성 요소를 그리는 **ActionScript** 코드만 들어 있습니다.

버전 2 구성 요소는 컴파일된 클립이며 라이브러리에서 에셋을 볼 수 없습니다. 하지만 **Flash** 를 설치하면 모든 구성 요소 스킨이 있는 **FLA** 파일이 함께 설치됩니다. 이러한 **FLA** 파일을 **테마**라고 합니다. 각 테마는 모양과 비헤이비어가 서로 다르지만 사용하는 스킨의 심볼 이름과 링크 식별자는 같습니다. 따라서 문서에서 스테이지로 테마를 드래그하여 그 모양을 변경할 수 있습니다. 테마 **FLA** 파일을 사용하여 구성 요소 스킨을 변경할 수도 있습니다. 스킨은 각 테마 **FLA** 파일에 대한 **라이브러리** 패널의 **Themes** 폴더에 있습니다. 테마에 대한 자세한 내용은 [104페이지의 “테마”](#)를 참조하십시오.

각 구성 요소는 여러 개의 스킨으로 구성됩니다. 예를 들어, **ScrollBar** 하위 구성 요소의 아래쪽 화살표는 **ScrollDownArrowDisabled**, **ScrollDownArrowDown**, **ScrollDownArrowOver** 및 **ScrollDownArrowUp**의 네 스킨으로 구성됩니다. 전체 **ScrollBar**는 13개의 스킨 심볼을 사용합니다.

일부 구성 요소는 스킨을 공유합니다. 예를 들어, **ComboBox**, **List** 및 **ScrollPane**과 같이 스크롤 막대를 사용하는 구성 요소는 **ScrollBar Skins** 폴더의 스킨을 공유합니다. 기존의 스킨을 편집하거나 새 스킨을 만들어 구성 요소 모양을 변경할 수 있습니다.

각 구성 요소 클래스를 정의하는 **AS** 파일에는 해당 구성 요소의 특정 스킨을 로드하는 코드가 들어 있습니다. 각 구성 요소 스킨은 스킨 심볼의 링크 식별자에 할당되는 스킨 속성에 해당합니다. 예를 들어, **ScrollBar** 구성 요소의 아래쪽 화살표를 누른 상태에 대한 스킨 속성 이름은 **downArrowDownName**입니다. **downArrowDownName** 속성의 기본값은 테마 **FLA** 파일에 있는 스킨 심볼의 링크 식별자인 **"ScrollDownArrowDown"**입니다. 기존 스킨을 편집한 다음 기존 링크 식별자는 그대로 둔 채 스킨 심볼을 편집하여 해당 스킨을 사용하는 모든 구성 요소에 편집한 스킨을 적용할 수 있습니다. 또는 새 스킨을 만든 다음 구성 요소 인스턴스의 스킨 속성을 설정하여 특정 구성 요소 인스턴스에 새로 만든 스킨을 적용할 수 있습니다. 구성 요소의 **AS** 파일을 편집하여 해당 스킨 속성을 변경하지 않아도 됩니다. 즉, 문서에 구성 요소를 만들 때 해당 구성 요소의 생성자 함수에 스킨 속성 값을 전달하면 됩니다.

각 구성 요소의 스킨 속성은 구성 요소 사전의 각 구성 요소 항목에 나열됩니다. 예를 들어, **Button** 구성 요소의 스킨 속성은 **ActionScript 2.0 구성 요소 언어 참조 설명서 > Button 구성 요소 > Button 구성 요소 사용자 정의 > Button 구성 요소에 스킨 사용**에 있습니다.

수행할 작업에 따라 다음 구성 요소 스키닝 방법 중 하나를 선택합니다. 이러한 방법은 난이도에 따라 가장 쉬운 것부터 차례로 나열되어 있습니다.

- 단일 문서에서 특정 구성 요소의 모든 인스턴스와 연결된 스킨을 변경하려면 개별 스킨 요소를 복사하여 수정합니다. 자세한 내용은 [94페이지의 “문서의 구성 요소 스킨 편집”](#)을 참조하십시오.

이 스키닝 방법은 스크립팅을 필요로 하지 않으므로 초급 사용자에게 권장됩니다.

- 각 종류의 구성 요소가 같은 모양을 공유하는 새 스킨 세트는 문서의 모든 스킨을 바꾸려면 테마를 적용합니다. 자세한 내용은 [104페이지의 “테마”](#)를 참조하십시오.  
이 스킨링 방법은 모든 구성 요소와 여러 문서에 일관된 모양과 느낌을 적용하려는 경우에 권장됩니다.
- 스킨 요소의 색상을 스타일 속성과 연결하려면 스킨에 `ActionScript` 코드를 추가하여 해당 스킨 요소를 색칠된 스킨 요소로 등록합니다. 자세한 내용은 [97페이지의 “스타일에 스킨 색상 연결”](#)을 참조하십시오.
- 같은 구성 요소의 여러 인스턴스에 서로 다른 스킨을 사용하려면 새 스킨을 만들고 스킨 속성을 설정합니다. 자세한 내용은 [96페이지의 “새 구성 요소 스킨 만들기”](#) 및 [99페이지의 “구성 요소에 새 스킨 적용”](#)을 참조하십시오.
- 하위 구성 요소(예: `List` 구성 요소의 스크롤 막대)의 스킨을 변경하려면 구성 요소의 하위 클래스를 만듭니다. 자세한 내용은 [100페이지의 “하위 구성 요소에 새 스킨 적용”](#)을 참조하십시오.
- 기본 구성 요소에서 직접 액세스할 수 없는 하위 구성 요소(예: `ComboBox` 구성 요소의 `List` 구성 요소)의 스킨을 변경하려면 프로토타입의 스킨 속성을 바꿉니다. 자세한 내용은 [103페이지의 “하위 구성 요소의 스킨 속성 변경”](#)을 참조하십시오.

## 문서의 구성 요소 스킨 편집

단일 문서에서 특정 구성 요소의 모든 인스턴스와 연결된 스킨을 편집하려면 테마에서 문서로 스킨 심볼을 복사하고 원하는 대로 그래픽을 편집합니다.

아래에 설명된 절차는 새 테마를 만들어 적용하는 것([104페이지의 “테마”](#) 참조)과 매우 비슷합니다. 기본적인 차이는 이 절차의 경우 이미 사용 중인 테마에서 단일 문서로 직접 심볼을 복사하고 사용 가능한 모든 스킨 중 몇 개만 편집하는 것에 대해 설명한다는 점입니다. 이 절차는 단일 문서에서만 수정하고 몇 개의 구성 요소에 대해서만 스킨을 수정할 때 적절합니다. 편집된 스킨이 여러 문서에서 공유되거나 여러 구성 요소의 변경을 포함할 경우에는 새 테마를 만드는 것이 스킨을 편집하는 측면에서 더 간편할 수 있습니다.

고급 스킨링에 대한 자료는 Adobe 개발자 센터([www.adobe.com/devnet/flash/articles/skinning\\_2004.html](http://www.adobe.com/devnet/flash/articles/skinning_2004.html))에서 온라인으로 찾을 수 있습니다.

## 문서의 구성 요소 스킨을 편집하려면:

1. 이미 문서에 Sample 테마를 적용한 경우 5단계로 건너뛴니다.
2. 파일 > 가져오기 > 외부 라이브러리 열기를 선택하고 SampleTheme fla 파일을 선택합니다.  
이 파일은 응용 프로그램 수준의 Configuration 폴더에 있습니다. 운영 체제에서의 정확한 위치는 104페이지의 “테마”를 참조하십시오.
3. 테마의 라이브러리 패널에서 Flash UI Components 2/Themes/MMDefault를 선택한 다음 문서에 있는 구성 요소의 Assets 폴더를 해당 문서 라이브러리로 드래그합니다.  
예를 들어, RadioButton Assets 폴더를 ThemeApply fla 라이브러리로 드래그합니다.
4. 개별 구성 요소의 Assets 폴더를 라이브러리로 드래그한 경우 각 구성 요소의 Assets 심볼이 첫 프레임으로 내보내기로 설정되어 있는지 확인합니다.  
예를 들어, RadioButton 구성 요소의 Assets 폴더는 RadioButton Assets라고 하며 모든 개별 에셋 심볼을 포함하는 RadioButtonAssets라고 하는 심볼을 가집니다. RadioButtonAssets 심볼에 첫 프레임으로 내보내기를 설정한 경우에는 모든 개별 에셋 심볼도 첫 프레임으로 내보내집니다.
5. 수정할 스킨 심볼을 두 번 클릭하여 심볼 편집 모드로 엽니다.  
예를 들어, States/RadioFalseDisabled 심볼을 엽니다.
6. 심볼을 수정하거나 그래픽을 삭제한 다음 새 그래픽을 만듭니다.  
확대하려면 보기 > 확대를 선택합니다. 스킨을 편집할 때 등록 포인트를 유지하여 스킨이 올바르게 표시되도록 해야 합니다. 편집한 모든 심볼의 왼쪽 모서리는 (0,0)에 위치해야 합니다.  
예를 들어, 내부 원을 연한 회색으로 변경합니다.
7. 스킨 심볼 편집을 마쳤으면 스테이지 위쪽의 편집 막대 왼쪽에 있는 뒤로 버튼을 클릭하여 문서 편집 모드로 돌아갑니다.
8. 5-7단계를 반복하여 변경할 모든 스킨을 편집합니다.



스테이지에 있는 구성 요소의 실시간 미리 보기에는 편집된 스킨이 반영되지 않습니다.

9. 컨트롤 > 무비 테스트를 선택합니다.  
이 예제에서는 비활성화 상태의 새 RadioButton 모양을 확인하기 위해 스테이지에 RadioButton 인스턴스가 있고 액션 패널에서 해당 enabled 속성을 false로 설정해야 합니다.

## 새 구성 요소 스킨 만들기

같은 구성 요소의 여러 인스턴스에서 서로 다른 스킨을 사용하려면 테마 FLA 파일을 열고 새 스킨 심볼을 만들어야 합니다. 구성 요소는 각 인스턴스에서 서로 다른 스킨을 쉽게 적용할 수 있도록 디자인되었습니다.

### 새 스킨을 만들려면:

1. **파일 > 열기**를 선택하고 템플릿으로 사용할 테마 FLA 파일을 엽니다.
2. **파일 > 다른 이름으로 저장**을 선택하고 **MyTheme.fla**와 같이 고유한 이름을 선택합니다.
3. 편집할 스킨(이 예제의 경우 RadioTrueUp)을 선택합니다.  
스킨은 Themes/MMDefault/*Component Assets* 폴더(이 예제의 경우 Themes/MMDefault/RadioButton Assets/States)에 있습니다.
4. 라이브러리 옵션 메뉴에서 또는 심볼을 마우스 오른쪽 버튼으로 클릭하여 **복제**를 선택하고 심볼에 **MyRadioTrueUp**과 같이 고유한 이름을 지정합니다.
5. 심볼 속성 대화 상자에서 **고급**을 클릭하고 **ActionScript에 내보내기**를 선택합니다.  
심볼 이름과 일치하는 링크 식별자가 자동으로 입력됩니다.
6. 라이브러리에서 새 스킨을 두 번 클릭하여 심볼 편집 모드에서 엽니다.
7. 무비 클립을 수정하거나 삭제한 다음 새로 만듭니다.  
확대하려면 **보기 > 확대**를 선택합니다. 스킨을 편집할 때 등록 포인트를 유지하여 스킨이 올바르게 표시되도록 해야 합니다. 편집한 모든 심볼의 왼쪽 모서리는 (0,0)에 위치해야 합니다.
8. 스킨 심볼 편집을 마쳤으면 스테이지 위쪽의 편집 막대 왼쪽에 있는 **뒤로** 버튼을 클릭하여 문서 편집 모드로 돌아갑니다.
9. **파일 > 저장**을 선택하고 MyTheme.fla는 열어 놓은 상태로 둡니다. 이제 편집한 스킨을 구성 요소에 적용하는 데 사용할 새 문서를 만들어야 합니다.  
자세한 내용은 99페이지의 “구성 요소에 새 스킨 적용”, 100페이지의 “하위 구성 요소에 새 스킨 적용” 또는 103페이지의 “하위 구성 요소의 스킨 속성 변경”을 참조하십시오.

아이  
FO

Flash에서는 실시간 미리 보기를 사용하여 스테이지에서 구성 요소를 볼 때 구성 요소 스킨의 변경 내용이 표시되지 않습니다.

## 스타일에 스킨 색상 연결

버전 2 구성 요소 프레임워크를 사용하면 스킨을 사용하는 구성 요소에 설정된 스타일에 스킨 요소의 시각적 에셋을 쉽게 연결할 수 있습니다. 무비 클립 인스턴스를 스타일에 등록하거나 전체 스킨 요소를 스타일에 등록하려면 스킨의 타임라인에서 **ActionScript** 코드를 추가하여 `mx.skins.ColoredSkinElement.setColorStyle(targetMovieClip, styleName)` 을 호출합니다.

### 스타일 속성에 스킨을 연결하려면:

1. 이미 문서에 **Sample** 테마를 적용한 경우 5단계로 건너뛴니다.
2. **파일 > 가져오기 > 외부 라이브러리 열기**를 선택하고 **SampleTheme.fla** 파일을 선택합니다. 이 파일은 응용 프로그램 수준의 **Configuration** 폴더에 있습니다. 운영 체제에서의 정확한 위치는 [104페이지의 “테마”](#)를 참조하십시오.
3. 테마의 **라이브러리** 패널에서 **Flash UI Components 2/Themes/MMDefault**를 선택한 다음 문서에 있는 구성 요소의 **Assets** 폴더를 해당 문서 라이브러리로 드래그합니다. 예를 들어, **RadioButton Assets** 폴더를 대상 라이브러리로 드래그합니다.
4. 개별 구성 요소의 **Assets** 폴더를 라이브러리로 드래그한 경우 각 구성 요소의 **Assets** 심볼이 **첫 프레임으로 내보내기**로 설정되어 있는지 확인합니다. 예를 들어, **RadioButton** 구성 요소의 **Assets** 폴더는 **RadioButton Assets**라고 하며 모든 개별 에셋 심볼을 포함하는 **RadioButtonAssets**라고 하는 심볼을 가집니다. **RadioButtonAssets** 심볼에 **첫 프레임으로 내보내기**를 설정한 경우에는 모든 개별 에셋 심볼도 첫 프레임으로 내보내집니다.
5. 수정할 스킨 심볼을 두 번 클릭하여 심볼 편집 모드로 엽니다. 예를 들어, **States/RadioFalseDisabled** 심볼을 엽니다.
6. 색칠될 요소가 무비 클립 인스턴스가 아닌 그래픽 심볼인 경우 **수정 > 심볼로 변환**을 사용하여 무비 클립 인스턴스로 변환합니다. 이 예제에서는 그래픽 심볼 **RadioShape1**의 인스턴스인 가운데 그래픽을 무비 클립 심볼로 변경한 다음 이름을 **Inner Circle**로 지정합니다. **ActionScript**에 **내보내기**는 선택하지 않아도 됩니다. 반드시 필요한 것은 아니지만 새로 만든 무비 클립 심볼을 편집 중인 구성 요소 에셋의 **Elements** 폴더로 이동하는 것이 좋습니다.
7. 이전 단계에서 그래픽 심볼을 무비 클립 인스턴스로 변환한 경우 해당 인스턴스에 이름을 지정하여 **ActionScript**에서 대상으로 삼을 수 있게 합니다. 이 예제에서는 인스턴스 이름을 **innerCircle**로 지정합니다.

8. **ActionScript** 코드를 추가하여 스킨 요소나 해당 스킨 요소가 색칠된 스킨 요소로 포함되어 있는 무비 클립 인스턴스를 등록합니다.

예를 들어, 스킨 요소의 타임라인에 다음 코드를 추가합니다.

```
mx.skins.ColoredSkinElement.setColorStyle(innerCircle,  
"symbolBackgroundDisabledColor");
```

이 예제에서는 이미 **Sample** 스타일의 기존 스타일 이름에 해당하는 색상을 사용하고 있습니다. 가능하면 **Halo** 및 **Sample** 테마에서 제공한 공식 **CSS(Cascading Style Sheet)** 표준 또는 스타일에 해당하는 스타일 이름을 사용하는 것이 가장 좋습니다.

9. 5-8 단계를 반복하여 변경할 모든 스킨을 편집합니다.  
이 예제에서는 **RadioTrueDisabled** 스킨에 대해 이러한 단계를 반복하지만 기존 그래픽을 무비 클립으로 변환하는 대신에 그래픽을 삭제하고 기존 **Inner Circle** 심볼을 **RadioTrueDisabled** 스킨 요소로 드래그합니다.
10. 스킨 심볼 편집을 마쳤으면 스테이지 위쪽의 편집 막대 왼쪽에 있는 **뒤로** 버튼을 클릭하여 문서 편집 모드로 돌아갑니다.
11. 구성 요소 인스턴스를 스테이지로 드래그합니다.  
이 예제에서는 두 개의 **RadioButton** 구성 요소를 스테이지로 드래그하고 하나를 선택 상태로 설정한 다음 **ActionScript**를 통해 둘 다 비활성 상태로 설정하여 변경 내용을 확인합니다.
12. 문서에 **ActionScript** 코드를 추가하여 구성 요소 인스턴스에 또는 전역 수준으로 새 스타일 속성을 설정합니다.  
이 예제에서는 다음과 같이 전역 수준으로 속성을 설정합니다.  

```
_global.style.setStyle("symbolBackgroundDisabledColor", 0xD9D9D9);
```
13. **컨트롤 > 무비 테스트**를 선택합니다.

## 구성 요소에 새 스킨 적용

새 스킨을 만든 후에는 문서에 있는 구성 요소에 적용해야 합니다. `createClassObject()` 메서드를 사용하여 구성 요소 인스턴스를 동적으로 만들거나 스테이지에 구성 요소 인스턴스를 수동으로 배치할 수 있습니다. 문서에 구성 요소를 추가한 방법에 따라 두 가지 방식으로 구성 요소 인스턴스에 스킨을 적용할 수 있습니다.

### 구성 요소를 동적으로 만들고 새 스킨을 적용하려면:

1. **파일 > 새로 만들기**를 선택하여 새 Flash 문서를 만듭니다.
2. **파일 > 저장**을 선택하고 파일에 **DynamicSkinning.fla**와 같이 고유한 이름을 지정합니다.
3. 스킨을 편집한 구성 요소(이 예제의 경우 **RadioButton**)를 포함하여, 구성 요소 패널에서 라이브러리로 구성 요소를 드래그합니다.

이렇게 하면 심볼이 문서의 라이브러리에만 추가되고 실제 문서에는 표시되지 않습니다.

4. **MyRadioTrueUp** 및 사용자 정의한 기타 심볼을 **MyTheme.fla**에서 **DynamicSkinning.fla**의 라이브러리로 드래그합니다.

이렇게 하면 심볼이 문서의 라이브러리에만 추가되고 실제 문서에는 표시되지 않습니다.

5. **액션 패널**을 열고 **프레임 1**에 다음을 입력합니다.

```
import mx.controls.RadioButton;
createClassObject(RadioButton, "myRadio", 0,
    {trueUpIcon:"MyRadioTrueUp", label: "My Radio Button"});
```

6. **컨트롤 > 무비 테스트**를 선택합니다.

### 스테이지에 구성 요소를 수동으로 추가하고 새 스킨을 적용하려면:

1. **파일 > 새로 만들기**를 선택하여 새 Flash 문서를 만듭니다.
2. **파일 > 저장**을 선택하고 파일에 **ManualSkinning.fla**와 같이 고유한 이름을 지정합니다.
3. 스킨을 편집한 구성 요소(이 예제의 경우 **RadioButton**)를 포함하여, 구성 요소 패널에서 스테이지로 구성 요소를 드래그합니다.

4. **MyRadioTrueUp** 및 사용자 정의한 기타 심볼을 **MyTheme.fla**에서 **ManualSkinning.fla**의 라이브러리로 드래그합니다.

이렇게 하면 심볼이 문서의 라이브러리에만 추가되고 실제 문서에는 표시되지 않습니다.

5. 스테이지에서 **RadioButton** 구성 요소를 선택하고 **액션 패널**을 엽니다.

6. **RadioButton** 인스턴스에 다음 코드를 첨부합니다.

```
onClipEvent(initialize){
    trueUpIcon = "MyRadioTrueUp";
}
```

7. **컨트롤 > 무비 테스트**를 선택합니다.

## 하위 구성 요소에 새 스킨 적용

구성 요소의 하위 구성 요소에 대한 스킨을 수정해야 하는데 해당 스킨 속성에 직접 액세스하지 못하는 경우가 생길 수 있습니다. 예를 들어, List 구성 요소의 스크롤 막대 스킨을 직접 변경할 수 있는 방법은 없습니다. 그러나 다음 코드를 사용하면 스크롤 막대 스킨에 액세스할 수 있습니다. 이 코드를 실행하면 이후에 만드는 모든 스크롤 막대에 새 스킨이 사용됩니다.

구성 요소가 하위 구성 요소로 구성되어 있는 경우에는 *ActionScript 2.0 구성 요소 언어 참조 설명서*의 해당 구성 요소 항목에 하위 구성 요소가 표시됩니다.

### 하위 구성 요소에 새 스킨을 적용하려면:

1. 96페이지의 “**새 구성 요소 스킨 만들기**”에 설명된 단계에 따라 스크롤 막대 스킨을 편집합니다. 이 예제에서는 ScrollDownArrowDown 스킨을 편집한 다음 **MyScrollDownArrowDown**이라는 새 이름을 지정합니다.
2. **파일 > 새로 만들기**를 선택하여 새 Flash 문서를 만듭니다.
3. **파일 > 저장**을 선택하고 파일에 **SubcomponentProject.fla**와 같이 고유한 이름을 지정합니다.
4. 구성 요소 패널의 List 구성 요소를 라이브러리로 드래그합니다.  
이렇게 하면 구성 요소가 **라이브러리** 패널에만 추가되고 실제 문서에는 나타나지 않습니다.
5. MyScrollDownArrowDown 및 편집한 기타 심볼을 MyTheme.fla에서 SubcomponentProject.fla의 라이브러리로 드래그합니다.  
이렇게 하면 심볼이 **라이브러리** 패널에만 추가되고 실제 문서에는 나타나지 않습니다.
6. 다음 중 하나를 수행합니다.
  - 문서의 모든 스크롤 막대를 변경하려면 **액션** 패널에서 타임라인의 프레임 1에 다음 코드를 입력합니다.

```
import mx.controls.List;
import mx.controls.scrollClasses.ScrollBar;
ScrollBar.prototype.downArrowDownName = "MyScrollDownArrowDown";
```

그런 다음 프레임 1에 다음 코드를 입력하여 동적으로 목록을 만듭니다.

```
createClassObject(List, "myListBox", 0, {dataProvider:
    ["AL", "AR", "AZ", "CA", "HI", "ID", "KA", "LA", "MA"]});
```

또는 라이브러리에서 스테이지로 List 구성 요소를 드래그할 수 있습니다.
  - 문서의 특정 스크롤 막대를 변경하려면 **액션** 패널에서 타임라인의 프레임 1에 다음 코드를 입력합니다.

```
import mx.controls.List;
import mx.controls.scrollClasses.ScrollBar;
var oldName = ScrollBar.prototype.downArrowDownName;
ScrollBar.prototype.downArrowDownName = "MyScrollDownArrowDown";
```

```
createClassObject(List, "myList1", 0, {dataProvider: ["AL", "AR", "AZ",
    "CA", "HI", "ID", "KA", "LA", "MA"]});
myList1.redraw(true);
ScrollBar.prototype.downArrowDownName = oldName;
```

예제 10

스크롤 막대가 표시되도록 데이터를 충분히 설정하거나 `vScrollPolicy` 속성을 `true`로 설정합니다.

## 7. 컨트롤 > 무비 테스트를 선택합니다.

스킨 심볼의 `#initclip` 섹션에서 하위 구성 요소의 `prototype` 객체에 스킨 속성을 설정하여 문서의 모든 구성 요소에 대해 하위 구성 요소 스킨을 설정할 수도 있습니다.

### #initclip을 사용하여 문서의 모든 구성 요소에 편집된 스킨을 적용하려면:

1. 96페이지의 “새 구성 요소 스킨 만들기”에 설명된 단계에 따라 스크롤 막대 스킨을 편집합니다. 이 예제에서는 `ScrollDownArrowDown` 스킨을 편집한 다음 `MyScrollDownArrowDown`이라는 새 이름을 지정합니다.
2. 파일 > 새로 만들기를 선택하여 새 Flash 문서를 만듭니다. `SkinsInitExample.fla`와 같이 고유한 이름으로 저장합니다.
3. 편집한 테마 라이브러리 예제의 라이브러리에서 `MyScrollDownArrowDown` 심볼을 선택하여 `SkinsInitExample.fla`의 라이브러리로 드래그합니다.  
이렇게 하면 심볼이 라이브러리에 추가되고 스테이지에는 표시되지 않습니다.
4. `SkinsInitExample.fla` 라이브러리에서 `MyScrollDownArrowDown`을 선택한 다음 라이브러리 옵션 메뉴에서 링크를 선택합니다.
5. **ActionScript에 내보내기** 체크 상자를 선택합니다. **확인**을 클릭합니다.  
첫 프레임으로 내보내기가 자동으로 선택됩니다. 그렇지 않은 경우 직접 선택합니다.
6. 라이브러리에서 `MyScrollDownArrowDown`을 두 번 클릭하여 심볼 편집 모드에서 엽니다.
7. `MyScrollDownArrowDown` 심볼의 프레임 1에 다음 코드를 입력합니다.  

```
#initclip 10
import mx.controls.scrollClasses.ScrollBar;
ScrollBar.prototype.downArrowDownName = "MyScrollDownArrowDown";
#endinitclip
```
8. 다음 중 하나를 수행하여 문서에 List 구성 요소를 추가합니다.
  - **구성 요소 패널**에서 스테이지로 List 구성 요소를 드래그합니다. 세로 스크롤 막대가 표시되도록 `label` 매개 변수를 충분히 입력합니다.

- 구성 요소 패널에서 라이브러리로 List 구성 요소를 드래그합니다.  
SkinsInitExample.fla에서 주 타임라인의 프레임 1에 다음 코드를 입력합니다.

```
createClassObject(mx.controls.List, "myListBox1", 0, {dataProvider:
    ["AL", "AR", "AZ", "CA", "HI", "ID", "KA", "LA", "MA"]});
```

예제

세로 스크롤 막대가 표시되도록 데이터를 충분히 추가하거나 vScrollPolicy를 true로 설정합니다.

다음 예제에서는 스테이지에 이미 있는 항목을 스키닝하는 방법을 보여 줍니다. 이 예제에서는 List 스크롤 막대만 스키닝하고 TextArea 또는 ScrollPane 스크롤 막대는 스키닝하지 않습니다.

### #initclip을 사용하여 문서의 특정 구성 요소에 편집된 스킨을 적용하려면:

1. 94페이지의 “문서의 구성 요소 스킨 편집”에 설명된 단계에 따라 스크롤 막대 스킨을 편집합니다. 이 예제에서는 ScrollDownArrowDown 스킨을 편집한 다음 MyScrollDownArrowDown이라는 새 이름을 지정합니다.

2. 파일 > 새로 만들기를 선택하여 새 Flash 문서를 만듭니다.
3. 파일 > 저장을 선택하고 MyVScrollTest.fla와 같이 고유한 이름을 지정합니다.
4. 테마 라이브러리에서 MyVScrollTest.fla 라이브러리로 MyScrollDownArrowDown을 드래그합니다.
5. 삽입 > 새 심볼을 선택하고 심볼에 MyVScrollBar와 같이 고유한 이름을 지정합니다.
6. ActionScript에 내보내기 체크 상자를 선택합니다. 확인을 클릭합니다.  
첫 프레임으로 내보내기가 자동으로 선택됩니다. 그렇지 않은 경우 직접 선택합니다.
7. MyVScrollBar 심볼의 프레임 1에 다음 코드를 입력합니다.

```
#initclip 10
import MyVScrollBar
Object.registerClass("VScrollBar", MyVScrollBar);
#endinitclip
```

8. 구성 요소 패널에서 스테이지로 List 구성 요소를 드래그합니다.
9. 세로 스크롤 막대가 표시되도록 속성 관리자에서 Label 매개 변수를 충분히 입력합니다.
10. 파일 > 저장을 선택합니다.
11. 파일 > 새로 만들기를 선택하여 새 ActionScript 파일을 만듭니다.
12. 다음 코드를 입력합니다.

```
import mx.controls.VScrollBar
import mx.controls.List
class MyVScrollBar extends VScrollBar{
function init():Void{
if (_parent instanceof List){
downArrowDownName = "MyScrollDownArrowDown";
}
}
```

```

        super.init();
    }
}

```

13. **파일 > 저장**을 선택하고 이 파일을 **MyVScrollBar.as**로 저장합니다.
14. 스테이지의 비어 있는 영역을 클릭한 다음 속성 관리자에서 **제작 설정** 버튼을 클릭합니다.
15. **ActionScript** 버전 옆의 **설정** 버튼을 클릭합니다.
16. **새 경로 추가(+)** 버튼을 클릭하여 새 클래스 경로를 추가한 다음 **대상** 버튼을 선택하여 하드 디스크에 있는 **MyVScrollBar.as** 파일을 찾습니다.
17. **컨트롤 > 무비 테스트**를 선택합니다.

## 하위 구성 요소의 스킨 속성 변경

구성 요소에서 스킨 변수를 직접 지원하지 않는 경우에는 구성 요소의 하위 클래스를 만들어 스킨을 바꿀 수 있습니다. 예를 들어, **ComboBox** 구성 요소는 **List** 구성 요소를 드롭 다운 목록으로 사용하기 때문에 드롭 다운 목록 스킨을 직접 지원하지 않습니다.

구성 요소가 하위 구성 요소로 구성되어 있는 경우에는 *ActionScript 2.0* 구성 요소 언어 참조 설명서의 해당 구성 요소 항목에 하위 구성 요소가 표시됩니다.

### 하위 구성 요소를 스킨하려면:

1. 94페이지의 “문서의 구성 요소 스킨 편집”에 설명된 단계에 따라 스크롤 막대 스킨을 편집합니다. 이 예제에서는 **ScrollDownArrowDown** 스킨을 편집한 다음 **MyScrollDownArrowDown**이라는 새 이름을 지정합니다.
2. **파일 > 새로 만들기**를 선택하여 새 Flash 문서를 만듭니다.
3. **파일 > 저장**을 선택하고 **MyComboTest.fla**와 같이 고유한 이름을 지정합니다.
4. 위의 테마 라이브러리에서 **MyComboTest.fla**의 라이브러리로 **MyScrollDownArrowDown**을 드래그합니다.  
이렇게 하면 심볼이 라이브러리에 추가되고 스테이지에는 표시되지 않습니다.
5. **삽입 > 새 심볼**을 선택하고 심볼에 **MyComboBox**와 같이 고유한 이름을 지정합니다.
6. **ActionScript**에 **내보내기** 체크 상자를 선택한 다음 **확인**을 클릭합니다.  
첫 프레임으로 **내보내기**가 자동으로 선택됩니다. 그렇지 않은 경우 직접 선택합니다.
7. 액션 패널에서 **MyComboBox** 심볼의 프레임 1에 다음 코드를 입력합니다.

```

#initclip 10
    import MyComboBox
    Object.registerClass("ComboBox", MyComboBox);
#endinitclip

```
8. 심볼 편집을 마쳤으면 스테이지 위쪽의 편집 막대 왼쪽에 있는 **뒤로** 버튼을 클릭하여 문서 편집 모드로 돌아옵니다.

9. ComboBox 구성 요소를 스테이지로 드래그합니다.
10. 세로 스크롤 막대가 표시되도록 속성 관리자에 Label 매개 변수를 충분히 입력합니다.
11. **파일 > 저장**을 선택합니다.
12. **파일 > 새로 만들기**를 선택하여 새 **ActionScript** 파일을 만듭니다.
13. 다음 코드를 입력합니다.

```
import mx.controls.ComboBox
import mx.controls.scrollClasses.ScrollBar
class MyComboBox extends ComboBox{
    function getDropdown():Object{
        var oldName = ScrollBar.prototype.downArrowDownName;
        ScrollBar.prototype.downArrowDownName = "MyScrollDownArrowDown";
        var r = super.getDropdown();
        ScrollBar.prototype.downArrowDownName = oldName;
        return r;
    }
}
```

14. **파일 > 저장**을 선택하고 이 파일을 **MyComboBox.as**로 저장합니다.
15. MyComboTest.fla 파일로 돌아갑니다.
16. 스테이지의 비어 있는 영역을 클릭한 다음 속성 관리자에서 **제작 설정** 버튼을 클릭합니다.
17. **ActionScript** 버전 옆의 **설정** 버튼을 클릭합니다.
18. **새 경로 추가(+)** 버튼을 클릭하여 새 클래스 경로를 추가한 다음 **대상** 버튼을 선택하여 하드 디스크에 있는 MyComboBox.as 파일을 찾습니다.
19. **컨트롤 > 무비 테스트**를 선택합니다.

## 테마

테마는 스타일 및 스킨의 모음입니다. Flash에서 제공되는 기본 테마는 Halo(HaloTheme.fla)입니다. Halo 테마는 사용자에게 빠르게 응답하고 다양하게 표현할 수 있는 환경을 제공합니다. Flash에는 Sample(SampleTheme.fla)과 같은 추가 테마가 포함되어 있습니다. Sample 테마는 더 많은 스타일을 사용하여 사용자 정의하는 방법에 대한 예제를 제공합니다. Halo 테마가 Sample 테마에 포함된 모든 스타일을 사용하는 것은 아닙니다. 기본 설치할 경우 테마 파일은 다음 폴더에 있습니다.

- Windows의 경우: C:\Program Files\Adobe\Adobe Flash CS3\language\Configuration\ComponentFLA\
  - Macintosh의 경우: HD/Applications/Adobe Flash CS3/Configuration/ComponentFLA/
- 새 테마를 만들고 응용 프로그램에 적용하여 모든 구성 요소의 모양과 느낌을 변경할 수도 있습니다. 예를 들어, 원래 운영 체제 모양을 모방한 테마를 만들 수 있습니다.

구성 요소의 모양은 스킨(그래픽 또는 무비 클립 심볼)을 사용하여 표시됩니다. 각 구성 요소를 정의하는 AS 파일에는 해당 구성 요소의 특정 스킨을 로드하는 코드가 들어 있습니다. Halo 또는 Sample 테마를 복사한 다음 스킨의 그래픽을 변경하는 방법으로 새 테마를 손쉽게 만들 수 있습니다.

새 스타일 기본값 세트도 테마에 포함되었을 수 있는데 그러려면 `ActionScript` 코드를 작성하여 전역 스타일 선언이나 기타 필요한 스타일 선언을 만들어야 합니다. 자세한 내용은 [109페이지](#)의 “[테마에서 기본 스타일 속성 값 수정](#)”을 참조하십시오.

## 테마 전환

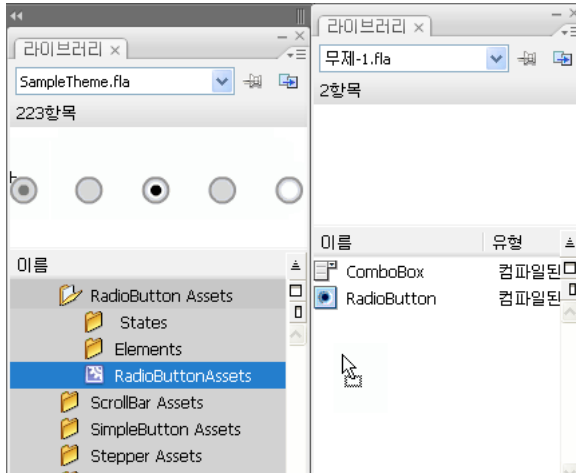
Flash를 설치하면 Halo와 Sample이라는 두 개의 테마가 설치됩니다. 각 구성 요소에 대한 참조 정보에는 두 테마 중 하나 또는 모두에 대해 설정할 수 있는 스타일 속성 표가 포함되어 있습니다. 따라서 *ActionScript 2.0 구성 요소 언어 참조 설명서*의 “`Button` 구성 요소에 스타일 사용”에 있는 `Button` 구성 요소에 대한 것과 같은 스타일 속성 표를 읽을 때는 자신이 원하는 스타일을 지원하는 테마가 어떤 것인지 확인하십시오. 이 표에서는 Halo, Sample 또는 둘 다(두 테마 모두 스타일 속성 지원)로 표시됩니다.

Halo 테마가 구성 요소의 기본 테마입니다. 따라서 Sample 테마를 사용하려면 현재 테마를 Halo에서 Sample로 전환해야 합니다.

### Sample 테마로 전환하려면:

1. **파일 > 열기**를 선택하여 Flash에서 버전 2 구성 요소를 사용하는 문서를 열거나 **파일 > 새로 만들기**를 선택하여 버전 2 구성 요소를 사용하는 새 문서를 만듭니다.
2. **파일 > 가져오기 > 외부 라이브러리 열기**를 선택하고 자신의 문서에 적용할 `SampleTheme.fla` 파일을 선택합니다.  
이 파일은 응용 프로그램 수준의 `Configuration` 폴더에 있습니다. 운영 체제에서의 정확한 위치는 [104페이지](#)의 “[테마](#)”를 참조하십시오.
3. `SampleTheme.fla` 테마의 **라이브러리** 패널에서 `Flash UI Components 2/Themes/MMDefault`를 선택한 다음 문서에 있는 구성 요소의 `Assets` 폴더를 자신의 Flash 문서의 라이브러리 패널로 드래그합니다.

예를 들어, RadioButton Assets 폴더를 자신의 라이브러리로 드래그합니다.



문서에 있는 구성 요소가 무엇인지 모르면 전체 Sample 테마 무비 클립을 스테이지로 드래그합니다. 문서에 있는 구성 요소에 스킨이 자동으로 할당됩니다.

**예외** 스테이지에 있는 구성 요소의 실시간 미리 보기에는 새 테마가 반영되지 않습니다.

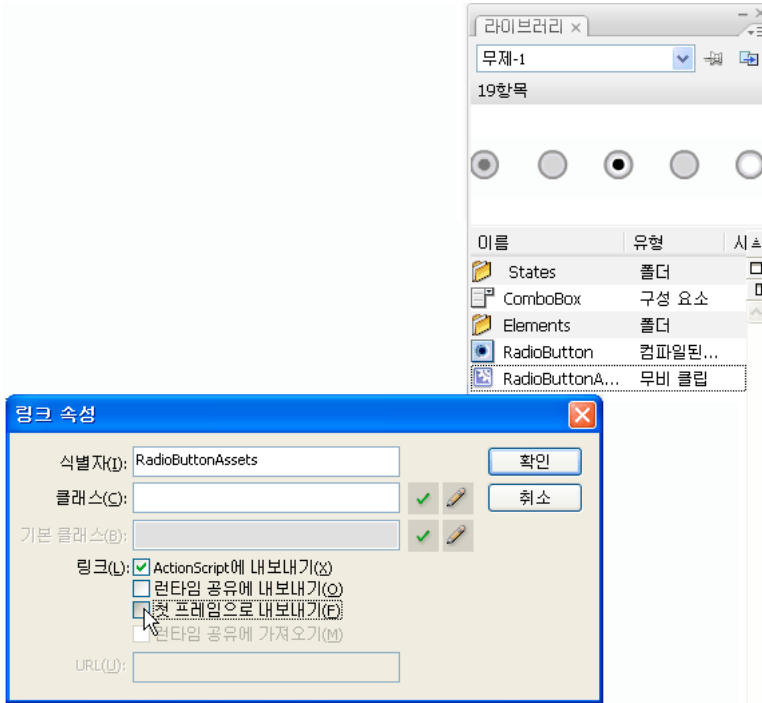
4. 개별 구성 요소 Assets 폴더를 문서의 라이브러리 패널로 드래그하는 경우 각 구성 요소의 Assets 심볼이 첫 프레임으로 내보내기로 설정되어 있는지 확인합니다.

예를 들어, RadioButton 구성 요소의 Assets 폴더는 RadioButton Assets입니다.

RadioButtonAssets 폴더를 열면 RadioButtonAssets라는 무비 클립 심볼을 볼 수 있습니다.

RadioButtonAssets 심볼에는 그 자체에 개별 에셋 심볼이 모두 포함되어 있습니다.

문서의 라이브러리에서 RadioButtonAssets 심볼을 마우스 오른쪽 버튼으로 클릭하거나 (Windows) Control 키를 누른 상태에서 클릭한 다음(Macintosh) 링크 메뉴 옵션을 선택합니다. 개별 예셋 심볼도 모두 첫 프레임으로 내보낼 수 있도록 첫 프레임으로 내보내기를 선택합니다. 그런 다음 확인을 클릭하여 설정을 저장합니다.



5. 컨트롤 > 무비 테스트를 선택하여 새 테마가 적용된 문서를 봅니다.

## 새 테마 만들기

Halo 테마나 Sample 테마를 사용하지 않을 경우 이들 테마 중 하나를 수정하여 새 테마를 만들 수 있습니다.

테마의 일부 스킨 크기는 고정되어 있습니다. 이러한 스킨의 크기를 늘리거나 줄이면 해당 스킨 크기에 맞게 구성 요소의 크기가 자동으로 조절됩니다. 기타 스킨은 여러 부분으로 구성되며 그 중에는 정적인 부분도 있고 확장 가능한 부분도 있습니다.

RectBorder 및 ButtonSkin 과 같은 일부 스킨의 그래픽은 크기와 성능 면에서 더 효율적인 ActionScript 드로잉 API를 사용하여 그려집니다. 이러한 스킨에 대해서는 ActionScript 코드를 템플릿으로 사용하여 필요에 맞게 스킨을 조정할 수 있습니다.

각 구성 요소에서 지원하는 스킨과 해당 속성 목록은 *ActionScript 2.0 구성 요소 언어 참조 설명서*를 참조하십시오.

### 새 테마를 만들려면:

1. 템플릿으로 사용할 테마 FLA 파일을 선택한 다음 복사합니다.  
복사본에 **MyTheme.fla**와 같이 고유한 이름을 지정합니다.
2. Flash에서 **파일 > MyTheme.fla 열기**를 선택합니다.
3. 아직 열려 있지 않은 경우 **윈도우 > 라이브러리**를 선택하여 라이브러리를 엽니다.
4. 수정할 스킨 심볼을 두 번 클릭하여 심볼 편집 모드로 엽니다.  
스킨은 Flash UI Components 2/Themes/MMDefault/*Component Assets* 폴더(이 예에서는 *RadioButton Assets* 폴더)에 있습니다.
5. 심볼을 수정하거나 그래픽을 삭제한 다음 새 그래픽을 만듭니다.  
확대하려면 **보기 > 확대**를 선택합니다. 스킨을 편집할 때 등록 포인트를 유지하여 스킨이 올바르게 표시되도록 해야 합니다. 편집한 모든 심볼의 왼쪽 모서리는 (0,0)에 위치해야 합니다.  
예를 들어, States/RadioFalseDisabled 에셋을 열고 내부 원을 연한 회색으로 변경합니다.
6. 스킨 심볼 편집을 마쳤으면 스테이지 위쪽의 편집 막대 왼쪽에 있는 **뒤로** 버튼을 클릭하여 문서 편집 모드로 돌아갑니다.
7. 4-6단계를 반복하여 변경할 모든 스킨을 편집합니다.
8. 이 장 뒷부분에 설명된 단계에 따라 문서에 MyTheme.fla를 적용합니다. 자세한 내용은 [110페이지의 “문서에 새 테마 적용”](#)을 참조하십시오.

## 테마에서 기본 스타일 속성 값 수정

기본 스타일 속성 값은 Default라는 클래스의 각 테마에서 제공합니다. 테마를 사용자 정의하기 위해 기본값을 변경하려면 테마와 관련된 패키지에서 Default라고 하는 새 ActionScript 클래스를 만들고 원하는 대로 기본 설정을 변경합니다.

### 테마에서 기본 스타일 값을 수정하려면:

1. First Run/Classes/mx/skins에 새 테마 폴더를 만듭니다.  
예를 들어, **myTheme**라고 하는 폴더를 만듭니다.
2. 기존 Defaults 클래스를 새 테마 폴더에 복사합니다.  
예를 들어, mx/skins/halo/Defaults.as를 mx/skins/myTheme/Defaults.as에 복사합니다.
3. ActionScript 편집기에서 새 Defaults 클래스를 엽니다.  
Flash 사용자는 Flash 내에서 파일을 열 수 있습니다. 또는 Windows의 메모장이나 Macintosh의 SimpleText에서 파일을 열 수도 있습니다.
4. 클래스 선언을 수정하여 새 패키지를 반영합니다.  
예를 들어, 새 클래스 선언은 `class mx.skins.myTheme.Defaults`입니다.
5. 원하는 대로 스타일 설정을 수정합니다.  
예를 들어, 비활성화 상태의 기본 색상을 진한 빨강으로 변경합니다.  
`o.disabledColor = 0x663333;`
6. 변경된 Defaults 클래스 파일을 저장합니다.
7. 소스 테마에서 사용자 정의 테마로 기존 FocusRect 클래스를 복사합니다.  
예를 들어, mx/skins/halo/FocusRect.as를 mx/skins/myTheme/FocusRect.as에 복사합니다.
8. ActionScript 편집기에서 새 FocusRect 클래스를 엽니다.
9. 새 테마 패키지를 가리키도록 소스 테마 패키지에 대한 모든 참조를 수정합니다.  
예를 들어, 모든 “halo”를 “myTheme”로 변경합니다.
10. 변경된 FocusRect 클래스 파일을 저장합니다.
11. 사용자 정의 테마의 FLA 파일을 엽니다.  
이 예제에서는 MyTheme.fla를 사용합니다.
12. 라이브러리(원도우 > 라이브러리)를 열고 Defaults 심볼을 찾습니다.  
이 예제에서는 Flash UI Components 2/Themes/MMDefault/Defaults에 있습니다.
13. Default 심볼의 심볼 속성을 편집합니다.
14. AS 2.0 클래스 설정을 변경하여 새 패키지를 반영합니다.  
예제 클래스는 `mx.skins.myTheme.Defaults`입니다.
15. 확인을 클릭합니다.

**16.** FocusRect 심볼을 찾습니다.

이 예제에서는 Flash UI Components 2/Themes/MMDefault/FocusRect에 있습니다.

**17.** FocusRect 심볼의 심볼 속성을 편집합니다.

**18.** AS 2.0 클래스 설정을 변경하여 새 패키지를 반영합니다.

예제 클래스는 mx.skins.myTheme.FocusRect입니다.

**19.** 확인을 클릭합니다.

**20.** 다음 단원에 설명된 단계에 따라 문서에 사용자 정의 테마를 적용합니다.

사용자 정의 테마에서 대상 문서로 예셋을 드래그할 때 Defaults 및 FocusRect 심볼을 포함해야 합니다.

이 예제에서는 새 테마를 사용하여 비활성화된 구성 요소의 텍스트 색상을 사용자 정의했습니다. 단일 기본 스타일 속성 값을 변경하는 이 사용자 정의 과정은 [80페이지의 “스타일을 사용하여 구성 요소 색상 및 텍스트 사용자 정의”](#)에 설명된 대로 스타일 지정을 통해 더 쉽게 완료할 수 있습니다. 새 테마를 사용하여 기본값을 사용자 정의하는 것은 많은 스타일 속성을 사용자 정의하거나 이미 새 테마를 만들어 구성 요소 그래픽을 사용자 정의한 경우에 적절합니다.

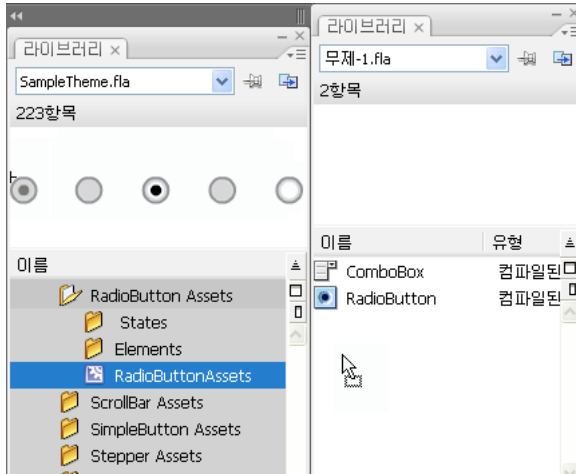
## 문서에 새 테마 적용

문서에 새 테마를 적용하려면 테마 FLA 파일을 외부 라이브러리로 연 다음 외부 라이브러리에서 문서 라이브러리로 테마 폴더를 드래그합니다. 다음 단계에서는 이미 새 테마가 있다고 가정하고 이 과정에 대해 자세히 설명합니다. 자세한 내용은 [108페이지의 “새 테마 만들기”](#)를 참조하십시오.

### 문서에 테마를 적용하려면:

- 1.** 파일 > 열기를 선택하여 Flash에서 버전 2 구성 요소를 사용하는 문서를 열거나 파일 > 새로 만들기를 선택하여 버전 2 구성 요소를 사용하는 새 문서를 만듭니다.
- 2.** 파일 > 가져오기 > 외부 라이브러리 열기를 선택한 다음 문서에 적용할 테마의 FLA 파일을 선택합니다.
- 3.** 테마의 라이브러리 패널에서 Flash UI Components 2/Themes/MMDefault를 선택한 다음 사용하려는 구성 요소의 Assets 폴더를 자신의 문서의 라이브러리로 드래그합니다.

예를 들어, RadioButton Assets 폴더를 자신의 라이브러리로 드래그합니다.

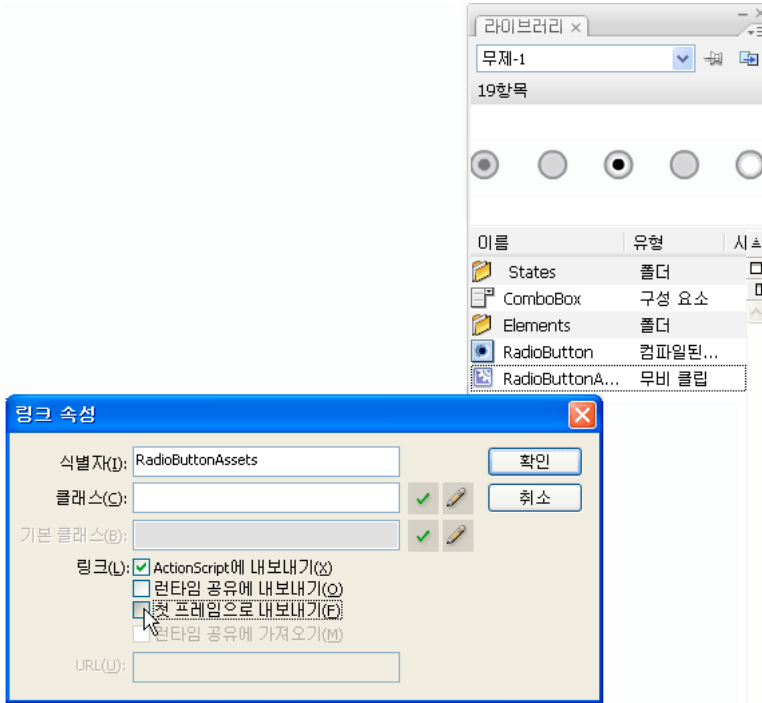


문서에 있는 구성 요소가 무엇인지 모르면 스테이지로 전체 테마 무비 클립을 드래그합니다(예를 들어, SampleTheme.fla의 경우 기본 테마 무비 클립은 **Flash UI Components 2 > SampleTheme**입니다). 문서에 있는 구성 요소에 스킨이 자동으로 할당됩니다.

**예외** 스테이지에 있는 구성 요소의 실시간 미리 보기에는 새 테마가 반영되지 않습니다.

4. 개별 구성 요소의 Assets 폴더를 ThemeApply.fla 라이브러리로 드래그한 경우 각 구성 요소의 Assets 심볼이 첫 프레임으로 내보내기로 설정되어 있는지 확인합니다.

예를 들어, RadioButton 구성 요소의 Assets 폴더는 RadioButton Assets라고 하며 모든 개별 에셋 심볼을 포함하는 RadioButtonAssets라고 하는 심볼을 가집니다. RadioButtonAssets 심볼에 첫 프레임으로 내보내기를 설정한 경우에는 모든 개별 에셋 심볼도 첫 프레임으로 내보내집니다.



5. 컨트롤 > 무비 테스트를 선택하여 새 테마가 적용되었는지 확인합니다.

## 내보내기 설정 변경

문서에 Sample 또는 Halo 테마를 적용하는 경우 많은 스킨 에셋이 첫 프레임에서 내보내지도록 설정되어 재생 시 구성 요소에서 바로 사용할 수 있습니다. 그러나 FLA 파일의 제작 내보내기 설정(파일 > 제작 설정 > Flash 탭 > ActionScript 버전 설정 버튼 > 클래스용 내보내기 프레임)을 첫 프레임 이후의 프레임으로 변경하는 경우 Sample 및 Halo 테마의 에셋에 대한 내보내기 설정도 변경해야 합니다. 이렇게 하려면 문서의 라이브러리에서 다음 구성 요소 에셋을 열고 첫 프레임으로 내보내기 체크 상자의 선택을 취소합니다(마우스 오른쪽 버튼 클릭 > 링크 > 첫 프레임으로 내보내기).

### Sample 테마

- Flash UI Components 2/Base Classes/UIObject
- Flash UI Components 2/Themes/MMDefault/Defaults
- Flash UI Components 2/Base Classes/UIObjectExtensions
- Flash UI Components 2/Border Classes/BoundingBox
- Flash UI Components 2/SampleTheme
- Flash UI Components 2/Themes/MMDefault/Button Assets/Elements/ButtonIcon
- Flash UI Components 2/Themes/MMDefault/DateChooser Assets/ Elements/Arrows/cal\_disabledArrow
- Flash UI Components 2/Themes/MMDefault/FocusRect
- Flash UI Components 2/Themes/MMDefault/Window Assets/ States/CloseButtonOver
- Flash UI Components 2/Themes/MMDefault/Accordion Assets/AccordionHeaderSkin
- Flash UI Components 2/Themes/MMDefault/Alert Assets/AlertAssets
- Flash UI Components 2/Themes/MMDefault/Border Classes/Border
- Flash UI Components 2/Themes/MMDefault/Border Classes/CustomBorder
- Flash UI Components 2/Themes/MMDefault/Border Classes/RectBorder
- Flash UI Components 2/Themes/MMDefault/Button Assets/ActivatorSkin
- Flash UI Components 2/Themes/MMDefault/Button Assets/ButtonSkin

### Halo 테마

- Flash UI Components 2/Base Classes/UIObject
- Flash UI Components 2/Themes/MMDefault/Defaults
- Flash UI Components 2/Base Classes/UIObjectExtensions
- Flash UI Components 2/Component Assets/BoundingBox
- Flash UI Components 2/HaloTheme
- Flash UI Components 2/Themes/MMDefault/Accordion Assets/AccordionHeaderSkin
- Flash UI Components 2/Themes/MMDefault/Alert Assets/AlertAssets

- Flash UI Components 2/Themes/MMDefault/Border Classes/Border
- Flash UI Components 2/Themes/MMDefault/Border Classes/CustomBorder
- Flash UI Components 2/Themes/MMDefault/Border Classes/RectBorder
- Flash UI Components 2/Themes/MMDefault/Button Assets/ActivatorSkin
- Flash UI Components 2/Themes/MMDefault/Button Assets/ButtonSkin
- Flash UI Components 2/Themes/MMDefault/Button Assets/Elements/ButtonIcon
- Flash UI Components 2/Themes/MMDefault/CheckBox Assets/Elements/  
CheckThemeColor1
- Flash UI Components 2/Themes/MMDefault/CheckBox Assets/CheckBoxAssets
- Flash UI Components 2/Themes/MMDefault/ComboBox Assets/ComboBoxAssets
- Flash UI Components 2/Themes/MMDefault/DataGrid Assets/DataGridAssets
- Flash UI Components 2/Themes/MMDefault/DateChooser Assets/DateChooserAssets
- Flash UI Components 2/Themes/MMDefault/FocusRect
- Flash UI Components 2/Themes/MMDefault/Menu Assets/MenuAssets
- Flash UI Components 2/Themes/MMDefault/MenuBar Assets/MenuBarAssets
- Flash UI Components 2/Themes/MMDefault/ProgressBar Assets/ProgressBarAssets
- Flash UI Components 2/Themes/MMDefault/RadioButton Assets/Elements/  
RadioThemeColor1
- Flash UI Components 2/Themes/MMDefault/RadioButton Assets/Elements/  
RadioThemeColor2
- Flash UI Components 2/Themes/MMDefault/RadioButton Assets/RadioButtonAssets
- Flash UI Components 2/Themes/MMDefault/ScrollBar Assets/HScrollBarAssets
- Flash UI Components 2/Themes/MMDefault/ScrollBar Assets/ScrollBarAssets
- Flash UI Components 2/Themes/MMDefault/ScrollBar Assets/VScrollBarAssets
- Flash UI Components 2/Themes/MMDefault/Stepper Assets/Elements/StepThemeColor1
- Flash UI Components 2/Themes/MMDefault/Stepper Assets/NumericStepperAssets
- Flash UI Components 2/Themes/MMDefault/Tree Assets/TreeAssets
- Flash UI Components 2/Themes/MMDefault/Window Assets/Window Assets

# 스키닝 및 스타일을 통합하여 구성 요소 사용자 정의

이 단원에서는 스타일, 테마 및 스키닝 설정을 사용하여 콤보 상자 구성 요소 인스턴스를 사용자 정의합니다. 이 절차에서는 스키닝과 스타일 설정을 결합하여 구성 요소의 고유한 프리젠테이션을 만드는 방법을 보여 줍니다.

## 스테이지에 구성 요소 인스턴스 만들기

이 연습의 첫 번째 부분에서는 사용자 정의할 ComboBox 인스턴스를 만들어야 합니다.

### ComboBox 인스턴스를 만들려면:

1. ComboBox 구성 요소를 스테이지로 드래그합니다.
2. 속성 패널에서 인스턴스 이름을 **my\_cb**로 지정합니다.
3. 기본 타임라인의 첫 번째 프레임에서 다음 `ActionScript`를 추가합니다(구성 요소가 아닌 프레임에 추가해야 하며, 액션 패널의 제목 막대에는 “액션 - 프레임”으로 나타나야 함).  

```
my_cb.addItem({data:1, label:"One"});  
my_cb.addItem({data:2, label:"Two"});
```
4. 컨트롤 > 무비 테스트를 선택하여 Halo 테마의 기본 스타일과 스키닝이 적용된 콤보 상자를 확인합니다.

## 새 스타일 선언 만들기

이제 새 스타일 선언을 만들어 해당 스타일 선언에 스타일을 지정해야 합니다. 스타일 선언에 원하는 스타일이 모두 들어 있으면 콤보 상자 인스턴스에 새 스타일 이름을 지정할 수 있습니다.

### 새 스타일 선언을 만들어 이름을 지정하려면:

1. 기본 타임라인의 첫 번째 프레임에서 `ActionScript`의 시작 부분에 다음 행을 추가합니다(코딩 규칙에 따라 모든 `import` 문은 `ActionScript`의 시작 부분에 배치해야 합니다).

```
import mx.styles.CSSStyleDeclaration;
```

2. 다음 행에서 새 스타일 선언의 이름을 지정하여 전역 스타일 정의에 추가합니다.

```
var new_style:Object = new CSSStyleDeclaration();  
_global.styles.myStyle = new_style;
```

새 스타일 선언을 `_global` 스타일 시트에 지정하고 나면 개별 스타일 설정을 `new_style` 스타일 선언에 연결할 수 있습니다. 단일 인스턴스의 스타일 정의 대신 구성 요소의 그룹에 대해 스타일 시트를 만드는 방법에 대한 자세한 내용은 [85페이지의 “특정 구성 요소의 사용자 정의 스타일 설정”](#)을 참조하십시오.

3. 일부 스타일 설정은 `new_style` 스타일 선언에 연결합니다. 다음 스타일 설정에는 `ComboBox` 구성 요소에 사용되는 스타일 정의(전체 목록은 *ActionScript 2.0 구성 요소 언어 참조 설명서*의 “`ComboBox` 구성 요소에 스타일 사용” 참조) 및 `RectBorder` 클래스의 스타일이 포함되어 있습니다. 여기서 `RectBorder` 클래스의 스타일이 포함된 이유는 `ComboBox` 구성 요소에서 이 클래스를 사용하기 때문입니다.

```
new_style.setStyle("textAlign", "right");
new_style.setStyle("selectionColor", "white");
new_style.setStyle("useRollOver", false);
// RectBorder 클래스의 borderStyle
new_style.setStyle("borderStyle", "none");
```

## 콤보 상자에 스타일 정의 지정

이제 다양한 스타일이 포함된 스타일 선언을 얻었지만 스타일 이름을 구성 요소 인스턴스에 명시적으로 지정해야 합니다. 이 새로운 스타일 선언은 문서 내의 *any* 구성 요소 인스턴스에 다음과 같은 방식으로 지정할 수 있습니다. `my_cb`에 대해 `addItem()` 문 이후에 다음 행을 추가합니다(코딩 규칙에 따라 모든 콤보 상자 구성 문을 함께 유지해야 합니다).

```
my_cb.setStyle("styleName", "myStyle");
```

기본 타임라인의 첫 번째 프레임에 첨부된 `ActionScript` 코드는 다음과 같습니다.

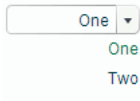
```
import mx.styles.CSSStyleDeclaration;

var new_style:Object = new CSSStyleDeclaration();
_global.styles.myStyle = new_style;

new_style.setStyle("textAlign", "right");
new_style.setStyle("selectionColor", "white");
new_style.setStyle("useRollOver", false);
// RectBorder 클래스의 borderStyle
new_style.setStyle("borderStyle", "none");

my_cb.addItem({data:1, label:"One"});
my_cb.addItem({data:2, label:"Two"});
my_cb.setStyle("styleName", "myStyle");
```

컨트롤 > 테스트 무비를 선택하여 스타일이 설정된 콤보 상자를 확인합니다.





## 콤보 상자 스킨 에셋 편집

구성 요소의 모양을 편집하려면 구성 요소를 그래픽적으로 구성하고 있는 스킨을 편집합니다. 스킨을 편집하려면 현재 테마에서 구성 요소의 그래픽 에셋을 열고 해당 구성 요소의 심볼을 편집합니다. 이렇게 하면 다른 구성 요소와 달리 심볼이 제거 또는 추가되지 않고 기존 구성 요소 스킨 심볼의 모양을 편집하므로, 이 접근 방식을 사용하는 것이 좋습니다.

정보

구성 요소에서 이름이 다른 심볼을 스킨으로 사용하도록 해당 구성 요소의 소스 클래스 파일을 편집할 수는 있지만, 권장되지는 않습니다. 또한 스킨 심볼 내에서 ActionScript를 프로그래밍 방식으로 변경할 수 있습니다(사용자 정의된 ActionScript 및 스킨 심볼의 예는 *ActionScript 2.0 구성 요소 언어 참조 설명서*의 “Accordion 구성 요소 사용자 정의” 참조). 그러나 ComboBox 구성 요소를 비롯한 여러 구성 요소는 다른 구성 요소와 에셋을 공유하므로 소스 파일을 편집하거나 스킨 심볼 이름을 변경하면 예기치 못한 결과가 발생할 수 있습니다.

구성 요소 스킨 심볼을 편집하는 경우:

- 구성 요소의 모든 인스턴스에서는 인스턴스에 스타일을 명시적으로 연결하지 않는 한 사용자 정의 스타일이 아닌 새 스킨을 사용하며, 이 구성 요소에 종속된 일부 구성 요소에서도 새 스킨을 사용하게 됩니다.
- 구성 요소 스킨을 편집한 후 새 테마 이름을 지정하는 경우 기존에 “편집한” 스킨을 덮어쓰지 않습니다(스킨을 덮어쓸 것인지 묻는 대화 상자가 나타나 Flash에서 스킨을 덮어쓰지 않도록 선택할 수 있습니다).

이 단원에서는 계속해서 이전 단원의 콤보 상자를 사용합니다(자세한 내용은 [117페이지의 “콤보 상자 테마 변경”](#) 참조). 다음 단계에서는 콤보 상자 메뉴를 여는 아래쪽 화살표의 모양을 화살표에서 원으로 변경합니다.

### 콤보 상자의 아래쪽 화살표 심볼을 편집하려면:

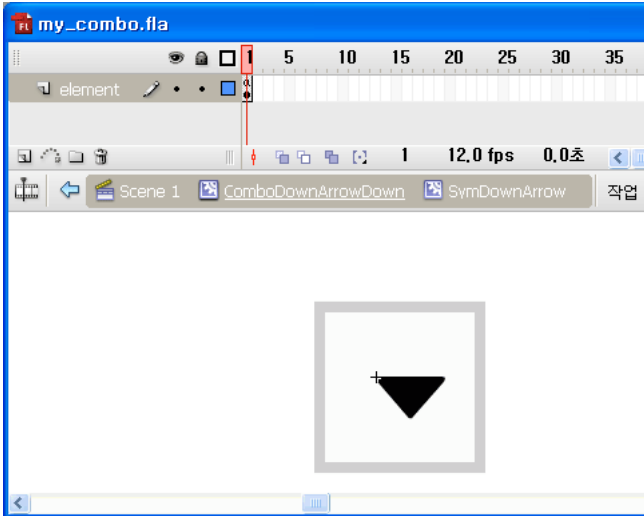
1. 문서 라이브러리에서 ComboBox 에셋을 열어 런타임에 콤보 상자 인스턴스를 열고 닫는 버튼의 스킨인 무비 클립을 확인합니다. 구체적으로 문서 라이브러리에서 Themes > MMDefault > ComboBox Assets > States 폴더를 엽니다.

States 폴더에는 ComboDownArrowDisabled, ComboDownArrowDown, ComboDownArrowOver 및 ComboDownArrowUp의 네 개의 무비 클립이 들어 있습니다. 이러한 네 개의 심볼은 모두 다른 심볼로 구성되어 있습니다. 또한 이들 심볼은 모두 아래쪽 화살표(삼각형)에 대해 SymDownArrow라는 동일한 심볼을 사용합니다.

2. ComboDownArrowDown 심볼을 두 번 클릭하여 편집합니다.  
버튼의 세부 정보를 확인하려면 800%까지 확대해야 합니다.

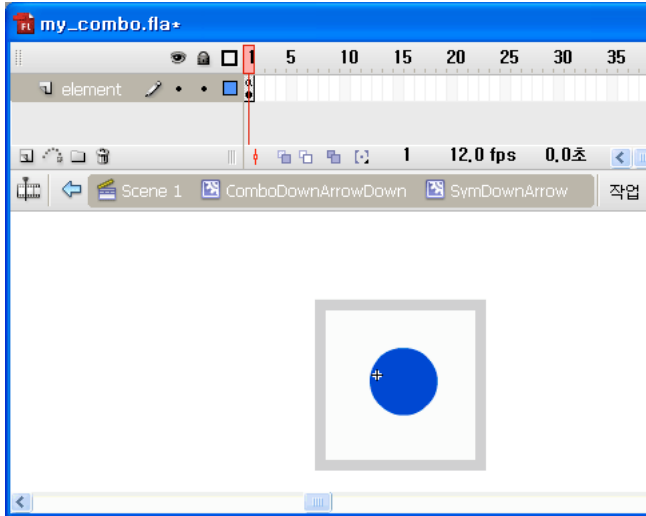
3. 아래쪽 화살표(검정 삼각형)을 두 번 클릭하여 편집합니다.

**예외** 무비 클립 심볼 자체가 아닌 무비 클립 내의 모양만 삭제하도록 SymDownArrow 심볼을 선택했는지 확인합니다.

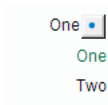


4. 스테이지에서 선택한 아래쪽 화살표(전체 무비 클립이 아닌 검정 삼각형 모양)를 삭제합니다.
5. SymDownArrow를 편집하는 상태에서 아래쪽 화살표가 있던 위치에 원을 그립니다.

변경 내용이 보다 잘 나타나도록 하려면 파랑과 같은 밝은 색상(약 4 x 4픽셀)의 원을  $x$  및  $y$  좌표로 0, -1을 사용하여 중앙에 오도록 그립니다.



6. 컨트롤 > 무비 테스트를 선택하여 스키닝한 콤보 상자를 확인합니다.



문서 라이브러리에서 ComboDownArrowOver 및 ComboDownArrowUp을 선택하면 이들 역시 아래쪽 화살표 심볼에 대해 SymDownArrow를 사용하므로 검정 삼각형 대신 파랑 원이 나타나게 됩니다.

# 구성 요소 만들기

이 장에서는 구성 요소를 직접 만들어 배포할 수 있게 패키지화하는 방법에 대해 설명합니다. 이 장에서 설명하는 단원은 다음과 같습니다.

구성 요소 소스 파일.....	121
구성 요소 구조 개요.....	122
첫 번째 구성 요소 만들기.....	123
부모 클래스 선택.....	131
구성 요소 무비 클립 만들기.....	134
ActionScript 클래스 파일 만들기.....	138
기존 구성 요소를 자신의 구성 요소에 통합.....	166
구성 요소 내보내기 및 배포.....	175
구성 요소 개발의 마지막 단계.....	178

## 구성 요소 소스 파일

**구성 요소** 패널에서 사용할 수 있는 구성 요소는 미리 컴파일된 SWC 클립입니다. 사용자 정의 구성 요소를 만들 수 있도록, 그래픽이 포함된 소스 Flash 문서(FLA)와 이러한 구성 요소에 대한 코드가 포함된 소스 ActionScript 클래스 파일(AS)도 함께 제공됩니다. 버전 2 구성 요소의 소스 파일은 Adobe Flash와 함께 설치됩니다. 구성 요소를 직접 만들기 전에 이러한 파일을 열어 살펴 보고 구조를 이해하면 큰 도움이 됩니다. RadioButton 구성 요소는 가장 먼저 살펴볼 수 있는 간단한 구성 요소입니다. 모든 구성 요소는 StandardComponents.fla의 라이브러리에 있는 심볼입니다. 각 심볼은 ActionScript 클래스에 연결되어 있습니다. 위치는 다음과 같습니다.

- FLA 파일 소스 코드
  - Windows의 경우: C:\Program Files\Adobe\Adobe Flash CS3\language\Configuration\ComponentFLA\StandardComponents.fla.
  - Macintosh의 경우: HD/Applications/Adobe Flash CS3/Configuration/ComponentFLA/StandardComponents.fla

- ActionScript 클래스 파일
  - Windows의 경우: C:\Program Files\Adobe\Adobe Flash CS3\language\First Run\Classes\mx
  - Macintosh의 경우: HD/Applications/Adobe Flash CS3/First Run/Classes/mx

## 구성 요소 구조 개요

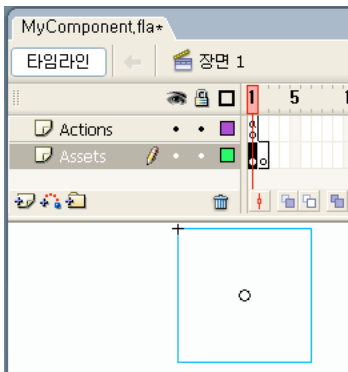
구성 요소는 Flash(FLA) 파일과 ActionScript(AS) 파일로 구성됩니다. 선택적으로 만들어 구성 요소와 함께 패키지화할 수 있는 다른 파일(예: 아이콘 및 .swd 디버깅 파일)도 있지만 FLA 파일과 ActionScript 파일은 모든 구성 요소에서 필수입니다. 구성 요소 개발이 완료되면 해당 구성 요소를 SWC 파일로 내보냅니다.



### Flash(FLA) 파일, ActionScript(AS) 파일 및 SWC 파일

FLA 파일에는 링크 속성 대화 상자와 구성 요소 정의 대화 상자 모두에서 AS 파일에 연결해야 할 무비 클립 심볼이 포함되어 있습니다.

무비 클립 심볼에는 프레임과 레이어가 두 개씩 있습니다. 첫 번째 레이어는 Actions 레이어로, 프레임 1에 stop() 전역 함수가 있습니다. 두 번째 레이어는 두 개의 키 프레임이 있는 Assets 레이어입니다. 프레임 1은 경계 상자를 포함하며 프레임 2는 구성 요소에서 사용하는 그래픽 및 기본 클래스와 같은 나머지 에셋을 포함합니다.



구성 요소의 속성과 메서드를 지정하는 ActionScript 코드는 별도의 ActionScript 클래스 파일에 있습니다. 이 클래스 파일은 구성 요소가 어떤 클래스를 확장하는지도 선언합니다(해당 클래스가 있는 경우). AS 클래스 파일 이름은 ".as" 확장명을 가진 구성 요소 이름입니다. 예를 들어, MyComponent.as는 MyComponent 구성 요소에 대한 소스 코드를 포함합니다.

구성 요소의 FLA 파일과 AS 파일은 동일한 폴더에 저장하고 동일한 이름을 지정하는 것이 좋습니다. AS 파일을 동일한 폴더에 저장하지 않았으면 클래스 경로에 폴더가 있어야 FLA 파일에서 해당 폴더를 찾을 수 있습니다. 클래스 경로에 대한 자세한 내용은 *Adobe Flash에서 ActionScript 2.0 학습*의 “클래스”를 참조하십시오.

## 첫 번째 구성 요소 만들기

이 단원에서는 Dial 구성 요소를 만듭니다. Dial 구성 요소의 샘플은 Flash 샘플 페이지 ([www.adobe.com/go/learn\\_fl\\_tutorials\\_kr](http://www.adobe.com/go/learn_fl_tutorials_kr))를 참조하십시오. 다음과 같은 샘플을 사용할 수 있습니다.

- Dial fla
- Dial.as
- DialAssets fla

Dial 구성 요소는 전위차계로, 전압의 잠재적인 차이를 측정하는 데 사용하는 것과 동일합니다. 사용자가 바늘을 클릭하고 드래그하여 위치를 변경할 수 있습니다. Dial의 API에는 바늘 위치를 알아내거나 설정하는 데 사용할 수 있는 속성인 value가 있습니다.

이 단원에서는 구성 요소를 만드는 절차를 배울 수 있습니다. 이후 단원에서 이 절차에 대해 보다 자세히 다룹니다(131페이지의 “부모 클래스 선택”, 134페이지의 “구성 요소 무비 클립 만들기”, 138페이지의 “ActionScript 클래스 파일 만들기” 및 175페이지의 “구성 요소 내보내기 및 배포” 포함). 이 단원에서 설명하는 항목은 다음과 같습니다.

- 123페이지의 “Dial Flash(FLA) 파일 만들기”
- 126페이지의 “Dial 클래스 파일 만들기”
- 129페이지의 “Dial 구성 요소 테스트 및 내보내기”

## Dial Flash(FLA) 파일 만들기

구성 요소를 만드는 첫 단계는 FLA 문서 파일 내에 구성 요소 무비 클립을 만드는 작업이 포함됩니다.

### Dial FLA 파일을 만들려면:

1. Flash에서 **파일 > 새로 만들기**를 선택하고 새 문서를 만듭니다.
2. **파일 > 다른 이름으로 저장**을 선택하고 **Dial fla**라는 이름으로 파일을 저장합니다.  
아무 파일 이름이나 지정할 수 있지만 구성 요소와 동일한 이름을 지정하는 것이 좋습니다.
3. **삽입 > 새 심볼**을 선택합니다. 구성 요소 자체는 새 **MovieClip** 심볼로 만들어지므로 라이브러리를 통해 사용할 수 있습니다.  
구성 요소 **Dial**의 이름을 지정하고 무비 클립 비헤이비어를 지정합니다.

4. 새 심볼 생성 대화 상자의 링크 섹션이 열려 있지 않으면 고급 버튼을 클릭하여 엽니다.
5. 링크 영역에서 **ActionScript**에 내보내기를 선택하고 첫 프레임으로 내보내기는 선택 취소합니다.
6. 식별자 텍스트 상자에 링크 식별자로 **Dial\_ID**를 입력합니다.
7. **AS 2.0 클래스** 텍스트 상자에 **Dial**을 입력합니다. 이 값은 구성 요소 클래스 이름입니다. 클래스가 패키지(예: mx.controls.Button)에 포함된 경우에는 전체 패키지 이름을 입력합니다.
8. **확인**을 클릭합니다.  
이렇게 하면 심볼 편집 모드로 전환됩니다.
9. 새 레이어를 삽입합니다. 맨 위 레이어의 이름으로 **Actions**를, 맨 아래 레이어의 이름으로 **Assets**를 지정합니다.
10. Assets 레이어에서 프레임 2를 선택하고 키프레임(F6)을 삽입합니다.  
이것은 구성 요소 무비 클립의 구조이며 Actions 레이어와 Assets 레이어로 구성됩니다. Actions 레이어에는 한 개의 키프레임이 Assets 레이어에는 두 개의 키프레임이 있습니다.
11. Actions 레이어에서 프레임 1을 선택하고 **액션** 패널(F9)을 엽니다. stop(); 전역 함수를 추가합니다.  
이렇게 하면 무비 클립이 프레임 2로 진행하지 않게 됩니다.
12. **파일 > 가져오기 > 외부 라이브러리 열기**를 선택하고 Configuration/ComponentFLA 폴더에서 StandardComponents.fla 파일을 선택합니다. 예를 들면 다음과 같습니다.
  - Windows의 경우: C:\Program Files\Adobe\Adobe Flash CS3\language\Configuration\ComponentFLA\StandardComponents.fla.
  - Macintosh의 경우: HD/Applications/Adobe Flash CS3/Configuration/ComponentFLA/StandardComponents.fla

<b>참고</b>	폴더 위치에 대한 자세한 내용은 <i>Flash 사용 설명서</i> 의 “Flash와 함께 설치된 Configuration 폴더”를 참조하십시오.
-----------	---

13. Dial이 UIComponent 기본 클래스를 확장하므로 UIComponent의 인스턴스를 Dial 문서로 드래그해야 합니다. StandardComponents.fla 라이브러리에서 Flash UI Components 2 > Base Classes > FUIObject Subclasses 폴더에 있는 UIComponent 무비 클립으로 이동한 다음 해당 무비 클립을 Dial.fla 라이브러리로 드래그합니다.

UIComponent와 함께 예셋 종속 항목들이 자동으로 Dial 라이브러리로 복사됩니다.

<b>참고</b>	UIComponent를 Dial 라이브러리로 드래그하면 Dial 라이브러리의 폴더 계층이 변경됩니다. 라이브러리를 다시 사용하거나 버전 2 구성 요소 같은 다른 구성 요소 그룹과 함께 사용할 계획인 경우에는 StandardComponents.fla 라이브러리와 일치하도록 폴더 계층을 다시 구성해야 해당 폴더 계층이 잘 정리되고 심볼이 중복되지 않습니다.
-----------	--

14. Assets 레이어에서 프레임 2를 선택하고 UIComponent의 인스턴스를 스테이지로 드래그합니다.

15. StandardComponents.fla 라이브러리를 닫습니다.

16. 파일 > 가져오기 > 외부 라이브러리 열기를 선택하고 DialAssets.fla 파일을 선택합니다.

DialAssets.fla 파일의 샘플은 Flash 샘플 페이지([www.adobe.com/go/learn\\_fl\\_tutorials\\_kr](http://www.adobe.com/go/learn_fl_tutorials_kr))를 참조하십시오.

17. Assets 레이어에서 프레임 2를 선택하고 DialAssets 라이브러리에서 스테이지로 DialFinal 무비 클립의 인스턴스를 드래그합니다.

모든 구성 요소 예셋은 Assets 레이어의 프레임 2에 추가됩니다. 액션 레이어의 프레임 1에 stop() 전역 함수가 있기 때문에 프레임 2에 있는 예셋은 스테이지에 배열될 때 보이지 않습니다.

다음 두 가지 이유 때문에 프레임 2에 예셋을 추가합니다.

- 모든 예셋과 하위 예셋을 자동으로 라이브러리로 복사하여, DialFinal일 경우에는 동적으로 인스턴스화하고 UIComponent의 경우에는 해당 메서드, 속성 및 이벤트에 액세스할 수 있습니다.
- 예셋을 프레임에 배치하면 무비가 스트리밍될 때 더욱 매끄럽게 로드되므로 라이브러리 예셋을 첫 프레임에 내보내도록 설정할 필요가 없습니다. 이러한 방법으로 다운로드 시 데이터 전송의 초기 부담을 피할 수 있습니다.

18. DialAssets.fla 라이브러리를 닫습니다.

19. Assets 레이어에서 프레임 1을 선택합니다. 라이브러리(Flash UI Components 2 >

Component Assets 폴더)에서 스테이지로 BoundingBox 무비 클립을 드래그합니다.

BoundingBox 인스턴스의 이름을 **boundingBox\_mc**로 지정합니다. 정보 패널을 사용하여 DialFinal 무비 클립의 높이와 폭을 250픽셀로 지정하고 x, y 좌표를 0, 0으로 지정합니다.

BoundingBox 인스턴스는 제작하는 동안 구성 요소의 실시간 미리 보기를 만들고 크기를 조절하는 데 사용됩니다. 모든 그래픽 요소를 구성 요소에 넣을 수 있도록 경계 상자의 크기를 조절해야 합니다.

예제 10

구성 요소(버전 2 구성 요소 포함)를 확장하는 경우 해당 구성 요소에서 이미 사용 중인 인스턴스 이름이 있으면 해당 코드가 이러한 인스턴스 이름을 참조하기 때문에 유지해야 합니다. 예를 들어, boundingBox\_mc라는 인스턴스 이름을 사용하는 버전 2 구성 요소를 포함할 경우 이 이름을 변경하지 마십시오. 인스턴스 이름을 직접 지정할 경우 동일한 범위 내의 기존 이름과 충돌하지 않는 고유한 이름을 사용할 수 있습니다.

20. 라이브러리에서 Dial 무비 클립을 선택하고 라이브러리 컨텍스트 메뉴에서 구성 요소 정의의 선택합니다(Windows: 마우스 오른쪽 버튼 클릭, Mac: Control 키를 누른 상태에서 클릭).

21. AS 2.0 클래스 텍스트 상자에 **Dial**을 입력합니다.

이 값은 ActionScript 클래스 이름입니다. 클래스가 패키지(mx.controls.CheckBox)에 포함된 경우에는 전체 패키지 이름을 입력합니다.

22. 확인을 클릭합니다.

23. 파일을 저장합니다.

## Dial 클래스 파일 만들기

이제 새 ActionScript 파일인 Dial 클래스 파일을 작성해야 합니다.

### Dial 클래스 파일을 작성하려면

1. Flash에서 **파일 > 새로 만들기**를 선택한 다음 **ActionScript 파일**을 선택합니다.
2. **파일 > 다른 이름으로 저장**을 선택하고 이 파일을 Dial.fla 파일과 같은 폴더에 **Dial.as**로 저장합니다.

**예제**

모든 텍스트 편집기를 사용하여 Dial.as 파일을 저장할 수 있습니다.

3. 다음의 Dial 구성 요소 ActionScript 클래스 코드를 새 Dial.as 파일로 복사하거나 입력할 수 있습니다. 코드를 복사하지 않고 입력하면 구성 요소 코드의 각 요소를 빨리 익힐 수 있습니다.

각 섹션에 대한 설명은 코드 주석을 참조하십시오. 구성 요소 클래스의 요소에 대한 자세한 내용은 [140페이지의 "구성 요소 클래스 파일 개요"](#)를 참조하십시오.

```
// 클래스를 직접 참조할 수 있도록 패키지를
// 가져옵니다.
import mx.core.UIComponent;

// Event 메타데이터 태그
[Event("change")]
class Dial extends UIComponent
{
    // 구성 요소가 이들이 구성 요소 프레임워크에서
    // 적절한 구성 요소임을 선언해야 합니다.
    static var symbolName:String = "Dial";
    static var symbolOwner:Object = Dial;
    var className:String = "Dial";

    // 바늘 및 Dial 무비 클립은 구성 요소를
    // 그래픽으로 표현한 것입니다.
    private var needle:MovieClip;
    private var dial:MovieClip;
    private var boundingBox_mc:MovieClip;

    // 전용 멤버 변수 "__value"는 암시적 getter/setter
    // 메서드를 통해 공개적으로 액세스할 수 있습니다.
```

```

// 값을 설정할 때 이 속성을 업데이트하면 바늘 위치가
// 업데이트됩니다.
private var __value:Number = 0;

// 이 플래그는 사용자가 마우스로 바늘을
// 드래그할 때 설정되고 이후에는 지워집니다.
private var dragging:Boolean = false;

// 생성자 ;
// 모든 클래스에 생성자가 필요하지만 v2 구성 요소의 경우에는
// 생성자가 인수 없이 비어 있어야 합니다.
// 모든 초기화는 클래스 인스턴스가 생성된 후
// 필수 init 메서드에서 수행됩니다.
function Dial() {
}

// 초기화 코드 :
// v2 구성 요소에 init 메서드가 필요합니다.
// 또한 init 메서드는 super.init()를 사용하여 해당 부모 클래스 init() 메서드를
// 호출합니다.
// UIComponent를 확장하는 구성 요소에 init 메서드가 필요합니다.
function init():Void {
    super.init();
    useHandCursor = false;
    boundingBox_mc._visible = false;
    boundingBox_mc._width = 0;
    boundingBox_mc._height = 0;
}

// 시작할 때 필요한 자식 객체를 만듭니다.
// UIComponent를 확장하는 구성 요소에 createChildren
// 메서드가 필요합니다.
public function createChildren():Void {
    dial = createObject("DialFinal", "dial", 10);
    size();
}

// v2 구성 요소에 draw 메서드가 필요합니다.
// draw 메서드는 invalidate()를 호출하는 사용자에게
// 의해 구성 요소가 무효화된 후 호출됩니다.
// 이 방법은 다른 속성이 있는 경우 여러 변경 내용을 개별적으로
// 처리하는 것보다 한 번의 다시 그리기로 일괄 처리하는 것이
// 낫기 때문에 값에 대한 set() 함수에서 다시 그리는 것보다
// 편리합니다. 이를 통해 효율성을 높이고 코드를 중앙 집중화할
// 수 있습니다.
function draw():Void {
    super.draw();
    dial.needle._rotation = value;
}

// size 메서드는 구성 요소의 크기가 변경될 때 호출됩니다.
// 이를 통해 자식과 Dial 및 바늘 그래픽의 크기를 조절할 수 있습니다.
// UIComponent를 확장하는 구성 요소에 size 메서드가 필요합니다.
function size():Void {

```

```

    super.size();
    dial._width = width;
    dial._height = height;
    // 필요한 경우 바늘이 다시 그려지게 합니다 .
    invalidate();
}

// 이는 값 속성에 대한 getter/setter입니다 .
// [Inspectable] 메타데이터는 속성 관리자에 속성이
// 나타나게 합니다 . 이 메타데이터는 값이 변경될 때 invalidate 를
// 호출하고 구성 요소가 다시 그리도록 하는 getter/setter입니다 .
[Bindable]
[ChangeEvent("change")]
[Inspectable(defaultValue=0)]
function set value (val:Number)
{
    __value = val;
    invalidate();
}

function get value ():Number
{
    return twoDigits(__value);
}

function twoDigits(x:Number):Number
{
    return (Math.round(x * 100) / 100);
}

// 마우스 누름을 예상하도록 구성 요소에 알립니다 .
function onPress()
{
    beginDrag();
}

// Dial 을 누르면 드래그 플래그가 설정됩니다 .
// 마우스 이벤트에 콜백 함수가 지정됩니다 .
function beginDrag()
{
    dragging = true;
    onMouseMove = mouseMoveHandler;
    onMouseUp = mouseUpHandler;
}

// 드래그가 완료되면 마우스 이벤트를 제거하고
// 플래그를 지웁니다 .
function mouseUpHandler()
{
    dragging = false;
    delete onMouseMove;
    delete onMouseUp;
}

```

```

    }

function mouseMoveHandler()
{
    // 각도를 계산합니다.
    if (dragging) {
        var x:Number = _xmouse - width/2;
        var y:Number = _ymouse - height/2;

        var oldValue:Number = value;
        var newValue:Number = 90+180/Math.PI*Math.atan2(y, x);
        if (newValue<0) {
            newValue += 360;
        }
        if (oldValue != newValue) {
            value = newValue;
            dispatchEvent( {type:"change"} );
        }
    }
}
}
}

```

## Dial 구성 요소 테스트 및 내보내기

그래픽 요소와 기본 클래스 그리고 Dial 구성 요소의 모든 기능이 포함된 클래스 파일로 이루어진 Flash 파일을 만들었습니다. 이제 구성 요소를 테스트해야 합니다.

작업하면서, 특히 클래스 파일을 작성하는 동안 구성 요소를 테스트하는 것이 가장 좋습니다. 작업하면서 테스트하는 가장 빠른 방법은 구성 요소를 컴파일된 클립으로 변환한 다음 구성 요소의 FLA 파일에서 해당 클립을 사용하는 것입니다.

구성 요소가 완성되면 SWC 파일로 내보냅니다. 자세한 내용은 [175페이지의 “구성 요소 내보내기 및 배포”](#)를 참조하십시오.

### Dial 구성 요소를 테스트하려면:

1. Dial.fla 파일의 라이브러리에서 Dial 구성 요소를 선택하고 **라이브러리** 컨텍스트 메뉴를 연 다음(Windows: 마우스 오른쪽 버튼으로 클릭, Mac: **Control** 키를 누른 상태에서 클릭) **컴파일된 클립으로 변환**을 선택합니다.  
컴파일된 클립이 Dial SWF라는 이름으로 라이브러리에 추가됩니다.

**이제**

두세 번 테스트한 적이 있거나 컴파일된 클립을 이미 만든 경우에는 라이브러리 충돌 해결 대화 상자가 표시됩니다. 기존 항목 교체를 선택하여 문서에 새 버전을 추가합니다.

2. Dial SWF를 기본 타임라인의 스테이지로 드래그합니다.
3. 속성 관리자나 구성 요소 관리자에서 크기를 조절하고 value 속성을 설정할 수 있습니다. value 속성을 설정하면 그에 따라 바늘 위치가 변경됩니다.

4. 런타임에 value 속성을 테스트하려면 Dial의 인스턴스 이름을 **dial**로 지정하고 기본 타임 라인의 프레임 1에 다음 코드를 추가합니다.

```
// 텍스트 필드의 위치
var textXPos:Number = dial.width/2 + dial.x
var textYPos:Number = dial.height/2 + dial.y;

// dial.value 를 볼 텍스트 필드를 만듭니다 .
createTextField("dialValue", 10, textXPos, textYPos, 100, 20);

// 리스너를 만들어 change 이벤트를 처리합니다 .
function change(evt){
// 박늘이 이동할 때마다 텍스트 필드에 값
// 속성을 배치합니다 .
    dialValue.text = dial.value;
}
dial.addEventListener("change", this);
```

5. 컨트롤 > 무비 테스트를 선택하여 Flash Player에서 구성 요소를 테스트합니다.

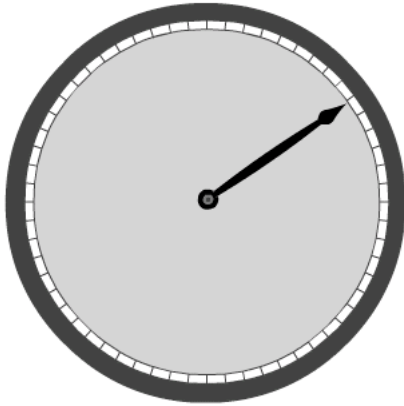
#### Dial 구성 요소를 내보내려면:

1. Dial.fla 파일의 라이브러리에서 Dial 구성 요소를 선택하고 **라이브러리** 컨텍스트 메뉴를 연 다음(Windows: 마우스 오른쪽 버튼으로 클릭, Mac: **Control** 키를 누른 상태에서 클릭) **SWC 파일로 내보내기**를 선택합니다.
2. SWC 파일을 저장할 위치를 선택합니다.

사용자 수준의 configuration 폴더에 있는 Components 폴더에 저장하면 Flash를 다시 시작하지 않고도 구성 요소 패널을 다시 로드할 수 있으며 이 경우 패널에 구성 요소가 표시됩니다.

예제

폴더 위치에 대한 자세한 내용은 *Flash 사용 설명서*의 “Flash와 함께 설치된 Configuration 폴더”를 참조하십시오.



완성된 Dial 구성 요소

## 부모 클래스 선택

구성 요소를 만들 때 맨 처음 결정해야 할 사항은 버전 2 클래스 중 하나를 확장할지 여부입니다. 버전 2 클래스를 확장하도록 선택하면 구성 요소 클래스(예: Button, CheckBox, ComboBox, List 등)를 확장하거나 기본 클래스(UIObject 또는 UICComponent) 중 하나를 확장할 수 있습니다. Media 구성 요소를 제외한 모든 구성 요소 클래스는 기본 클래스를 확장합니다. 구성 요소 클래스를 확장하면 자동으로 기본 클래스에서도 상속 받습니다.

두 기본 클래스는 구성 요소에 일반 기능을 제공합니다. 이러한 클래스를 확장하여 구성 요소는 기본 메서드, 속성 및 이벤트 집합으로 시작됩니다.

하위 클래스 UIObject나 UICComponent 또는 버전 2 프레임워크에 있는 다른 클래스를 작성할 필요는 없습니다. 구성 요소 클래스가 MovieClip 클래스에서 직접 상속 받기는 하지만 SWC 파일이나 컴파일된 클립으로 내보내고 내장된 실시간 미리 보기를 사용하며 관리 가능 속성을 보는 등의 많은 강력한 구성 요소 기능을 사용할 수 있습니다. 하지만 구성 요소와 Adobe 버전 2 구성 요소를 함께 사용하여 작업하고 관리자 클래스를 사용하려면 UIObject나 UICComponent를 확장해야 합니다.

다음 표에서는 버전 2 기본 클래스에 대해 간략히 설명합니다.

기본 클래스	확장 클래스	설명
<code>mx.core.UIObject</code>	<code>MovieClip</code>	<p>UIObject는 모든 그래픽 객체의 기본 클래스입니다. 이 클래스는 모양을 가지며 자체적으로 그려지며 보이지 않게 할 수도 있습니다.</p> <p>UIObject는 다음 기능을 제공합니다.</p> <ul style="list-style-type: none"><li>• 스타일 편집</li><li>• 이벤트 처리</li><li>• 크기 비율에 따라 크기 조절</li></ul>
<code>mx.core.UIComponent</code>	<code>UIObject</code>	<p>UIComponent는 모든 구성 요소의 기본 클래스입니다. UIComponent는 다음 기능을 제공합니다.</p> <ul style="list-style-type: none"><li>• 포커스 탐색 기능 만들기</li><li>• Tab 키를 눌렀을 때의 이동 체계</li><li>• 구성 요소 활성화 및 비활성화</li><li>• 구성 요소 크기 조절</li><li>• 하위 수준의 마우스 및 키보드 이벤트 처리</li></ul>

## UIObject 클래스의 이해

Adobe Component Architecture 버전 2 기반 구성 요소는 `MovieClip` 클래스의 하위 클래스인 `UIObject` 클래스에서 파생됩니다. `MovieClip` 클래스는 Flash에서 스크린에 시각적 객체를 표현하는 모든 클래스의 기본 클래스입니다.

`UIObject`는 스타일과 이벤트를 처리하는 데 사용할 수 있는 메서드를 추가하며 그리기 직전에(draw 이벤트는 `MovieClip.onEnterFrame` 이벤트와 같음), 로드하거나 언로드할 때(load, unload), 해당 레이아웃이 변경될 때(move, resize) 그리고 숨겨지거나 표시될 때(hide, reveal) 이벤트를 리스너에 게시합니다.

`UIObject`는 구성 요소의 위치 및 크기를 결정하기 위한 대체 읽기 전용 변수(width, height, x, y)와 객체의 위치 및 크기를 변경하기 위한 `move()` 및 `setSize()` 메서드를 제공합니다.

`UIObject` 클래스는 다음을 구현합니다.

- 스타일
- 이벤트
- 크기 비율에 따라 크기 조절

자세한 내용은 *ActionScript 2.0 구성 요소 언어 참조 설명서*를 참조하십시오.



## MovieClip 클래스 확장

버전 2 클래스를 확장하지 않도록 선택하고 구성 요소가 ActionScript MovieClip 클래스에서 직접 상속 받도록 할 수 있습니다. 하지만 UIObject 및 UIComponent 기능을 원할 경우에는 직접 만들어야 합니다. UIObject 및 UIComponent 클래스(First Run/Classes/mx/core)를 열어 서 작성 방식을 조사할 수 있습니다.

## 구성 요소 무비 클립 만들기

구성 요소를 만들려면 무비 클립 심볼을 만들어 구성 요소의 클래스 파일에 연결해야 합니다. 무비 클립에는 프레임과 레이어가 두 개씩 있습니다. 첫 번째 레이어는 Actions 레이어로, 프레임 1에 stop() 전역 함수가 있습니다. 두 번째 레이어는 두 개의 키프레임이 있는 Assets 레이어입니다. 프레임 1에는 경계 상자나 최종 아트의 자리 표시자 역할을 하는 그래픽이, 프레임 2에는 그래픽과 기본 클래스를 비롯하여 구성 요소에서 사용하는 다른 모든 에셋이 포함됩니다.

### 새 무비 클립 심볼 삽입

모든 구성 요소는 MovieClip 객체입니다. 구성 요소를 새로 만들려면 먼저 새 심볼을 새 FLA 파일에 삽입해야 합니다.

#### 새 구성 요소 심볼을 추가하려면:

1. Flash에서 빈 Flash 문서를 만듭니다.
2. **삽입 > 새 심볼**을 선택합니다.  
새 심볼 만들기 대화 상자가 나타납니다.
3. 심볼 이름을 입력합니다. 구성 요소의 각 단어 첫 문자를 대문자로 바꾸어 구성 요소 이름을 지정합니다(예: MyComponent).
4. 비헤이비어로 **무비 클립**을 선택합니다.
5. 고급 설정을 표시하려면 **고급** 버튼을 클릭합니다.
6. **ActionScript에 내보내기**를 선택하고 첫 프레임으로 내보내기는 선택 취소합니다.
7. 링크 식별자를 입력합니다.

8. **AS 2.0 클래스** 텍스트 상자에 ActionScript 2.0 클래스의 전체 경로를 입력합니다. 클래스 이름은 **구성 요소** 패널에 나타나는 구성 요소 이름과 동일해야 합니다. 예를 들어, Button 구성 요소의 클래스는 mx.controls.Button입니다.

**어**

파일 확장명은 포함시키지 마십시오. AS 2.0 클래스 텍스트 상자는 파일에 대한 파일 시스템 이름이 아니라 클래스의 패키지화된 위치를 가리키기 때문입니다.

ActionScript 파일이 패키지에 포함된 경우에는 패키지 이름을 포함시켜야 합니다. 이 값은 클래스 경로에 대한 상대 경로이거나 절대 패키지 경로 (예: mypackage.MyComponent)일 수 있습니다.

9. 대부분의 경우 기본적으로는 선택되어 있는 **첫 프레임으로 내보내기**를 선택 취소해야 합니다. 자세한 내용은 179페이지의 “구성 요소 개발 검사 목록”을 참조하십시오.
10. **확인**을 클릭합니다.

이렇게 하면 라이브러리에 심볼이 추가되고 심볼 편집 모드로 전환됩니다. 이 모드에서는 심볼 이름이 스테이지의 왼쪽 위 모서리에 표시되고 십자형의 심볼 등록 포인트가 나타납니다.

## 무비 클립 편집

새 심볼을 만들고 이 심볼에 대한 링크를 정의했으면 심볼의 타임라인에서 구성 요소의 에셋을 정의할 수 있습니다.

구성 요소의 심볼에는 두 개의 레이어가 있어야 합니다. 이 단원에서는 삽입할 레이어와 이러한 레이어에 추가할 대상에 대해 설명합니다.

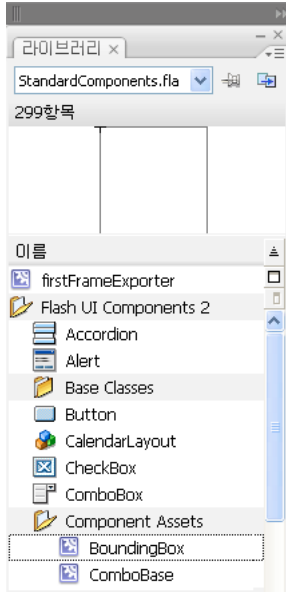
### 무비 클립을 편집하려면:

1. 레이어 1의 이름을 **Actions**로 바꾸고 프레임 1을 선택합니다.
2. **액션** 패널을 열고 다음과 같이 stop() 함수를 추가합니다.
 

```
stop();
```

 이 프레임에 그래픽 에셋은 추가하지 마십시오.
3. **Assets**라는 레이어를 추가합니다.
4. Assets 레이어에서 프레임 2를 선택하고 빈 키프레임을 삽입합니다. 이제 이 레이어에는 두 개의 빈 키프레임이 있습니다.
5. 다음 중 하나를 수행합니다.
  - 구성 요소에 경계 영역을 정의하는 시각적 에셋이 있는 경우 심볼을 프레임 1로 드래그하여 적절히 배열합니다.

- 구성 요소가 런타임에 모든 관련 시각적 에셋을 만드는 경우에는 BoundingBox 심볼을 프레임 1의 스테이지로 드래그하고 올바른 크기로 조정한 다음, 인스턴스 이름을 **boundingBox\_mc**로 지정합니다. 심볼은 Configuration/ComponentFLA 폴더에 있는 StandardComponents.fla의 라이브러리에 있습니다.



6. 기존 구성 요소를 확장하는 중이면 Assets 레이어의 프레임 2에 해당 구성 요소와 다른 기본 클래스의 인스턴스를 배치합니다.

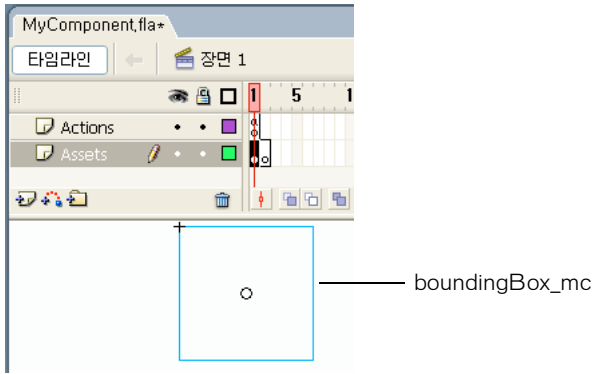
이렇게 하려면 **구성 요소** 패널에서 심볼을 선택하여 스테이지로 드래그합니다. 기본 클래스를 확장하는 중이면 Configuration/ComponentFLA 폴더에서 StandardComponents.fla를 열고 라이브러리에서 스테이지로 클래스를 드래그합니다.

**참고** UIComponent를 구성 요소 라이브러리로 드래그하면 라이브러리의 폴더 계층이 변경됩니다. 라이브러리를 다시 사용하거나 버전 2 구성 요소 같은 다른 구성 요소 그룹과 함께 사용할 계획인 경우에는 StandardComponents.fla 라이브러리와 일치하도록 폴더 계층을 다시 구성해야 해당 폴더 계층이 잘 정리되고 심볼이 중복되지 않습니다.

7. 이 구성 요소에서 사용하는 그래픽 에셋을 구성 요소 Assets 레이어의 프레임 2에 추가합니다.

Assets 레이어의 프레임 2에 구성 요소에서 사용하는 에셋(비트맵 같은 다른 구성 요소나 미디어어도 상관없음)의 인스턴스가 배치되어 있어야 합니다.

8. 완성된 심볼은 다음과 같습니다.



## 무비 클립을 구성 요소로 정의

**구성 요소 정의** 대화 상자에서 무비 클립 심볼을 `ActionScript` 클래스 파일에 연결해야 Flash에서 구성 요소의 메타태그를 찾을 위치를 알게 됩니다. 메타태그에 대한 자세한 내용은 [144 페이지의 “구성 요소 메타데이터 추가”](#)를 참조하십시오. 구성 요소 정의 대화 상자에는 선택할 수 있는 다른 옵션도 있습니다.

### 무비 클립을 구성 요소로 정의하려면:

1. 라이브러리에서 무비 클립을 선택하고 **라이브러리** 컨텍스트 메뉴에서 **구성 요소 정의**를 선택합니다(Windows: 마우스 오른쪽 버튼으로 클릭, Mac: **Control** 키를 누른 상태로 클릭).
2. AS 2.0 클래스를 입력해야 합니다.  
클래스가 패키지의 일부이면 전체 패키지 이름을 입력합니다.
3. 원하면 **구성 요소 정의** 대화 상자에서 다른 옵션을 지정합니다.
  - **+** 버튼을 클릭하여 매개 변수를 정의합니다.  
이것은 선택 사항입니다. 구성 요소의 클래스 파일에서 메타데이터 `Inspectable` 태그를 사용하여 매개 변수를 지정하는 것이 가장 좋습니다. `ActionScript 2.0` 클래스를 지정하지 않았으면 여기에서 구성 요소의 매개 변수를 정의합니다.
  - 사용자 정의 **UI**를 지정합니다.  
이것은 구성 요소 관리자에서 실행되는 `SWF` 파일입니다. 구성 요소의 `FLA` 파일에 포함시키거나 외부 `SWF`로 이동할 수 있습니다.

- 실시간 미리 보기를 지정합니다.  
이것은 외부 또는 포함된 SWF 파일입니다. 여기서는 실시간 미리 보기를 지정하지 않아도 됩니다. 구성 요소 무비 클립에 경계 상자를 추가하면 실시간 미리 보기가 자동으로 만들어집니다. 자세한 내용은 [134페이지의 “구성 요소 무비 클립 만들기”](#)를 참조하십시오.
- 설명을 입력합니다.  
참조 패널이 제거되었기 때문에 설명 필드는 Flash MX 2004에서 더 이상 사용할 수 없습니다. 이 필드는 FLA 파일을 Flash MX 포맷으로 저장할 때 이전 버전과의 호환을 위해 제공됩니다.
- 아이콘을 선택합니다.  
이 옵션에서는 구성 요소의 아이콘으로 사용할 PNG 파일을 지정합니다. ActionScript 2.0 클래스 파일에서 IconFile 메타데이터 태그를 지정하는 것이 좋습니다. 그러면 이 필드가 무시됩니다.
- 인스턴스에서 매개 변수 잠금을 선택하거나 선택 취소합니다.  
이 옵션을 선택 취소하면 구성 요소의 매개 변수와는 다른 매개 변수를 각 구성 요소 인스턴스에 추가할 수 있습니다. 일반적으로 이 설정을 선택해야 합니다. 이 옵션은 Flash MX와의 호환을 가능케 합니다.
- 구성 요소 패널에 나타나는 도구 설명을 지정합니다.

## ActionScript 클래스 파일 만들기

모든 구성 요소 심볼은 ActionScript 2.0 클래스 파일에 연결됩니다. 연결에 대한 자세한 내용은 [134페이지의 “구성 요소 무비 클립 만들기”](#)를 참조하십시오.

ActionScript 클래스 파일을 편집하려는 경우 Flash, 텍스트 편집기 또는 IDE(통합 개발 환경)를 사용할 수 있습니다.

외부 ActionScript 클래스는 다른 클래스(버전 2 구성 요소든 버전 2 기본 클래스든 ActionScript MovieClip 클래스든 상관없음)를 확장합니다. 만들려는 구성 요소와 가장 비슷한 기능을 만드는 클래스를 확장해야 합니다. 한 클래스에서만 상속 받을 수 있습니다. 즉, 한 클래스만 확장할 수 있습니다. ActionScript 2.0에서는 복수의 상속을 허용하지 않습니다.

## 간단한 구성 요소 클래스 파일 예제

다음은 MyComponent.as라는 클래스 파일의 간단한 예제입니다. 이 구성 요소를 만들었으면 Flash에서 이 파일을 구성 요소 무비 클립에 연결합니다.

이 예제에는 UIComponent 클래스를 상속 받는 구성 요소(MyComponent)에 대한 최소한의 가져오기, 메서드 및 선언들이 포함되어 있습니다. MyComponents.as 파일은 myPackage 폴더에 저장되어 있습니다.

```
[Event("eventName")]

// 패키지를 가져옵니다 .
import mx.core.UIObject;

// 클래스를 선언하고 부모 클래스로부터 확장합니다 .
class mypackage.MyComponent extends UIObject {

    // 이 클래스가 바인딩되는 심볼 이름을 식별합니다 .
    static var symbolName:String = "mypackage.MyComponent";

    // 심볼 소유자의 정규화된 패키지 이름을 식별합니다 .
    static var symbolOwner:Object = Object(mypackage.MyComponent);

    // className 변수를 제공합니다 .
    var className:String = "MyComponent";

    // 빈 생성자를 정의합니다 .
    function MyComponent() {
    }

    // 부모의 init() 메서드를 호출합니다 .
    // 경계 상자를 숨깁니다 .
    // 이 상자는 제작 중에만 사용됩니다 .
    function init():Void {
        super.init();

        boundingBox_mc.width = 0;
        boundingBox_mc.height = 0;
        boundingBox_mc.visible = false;
    }

    function createChildren():Void{
        // createClassObject 를 호출하여 하위 객체를 만듭니다 .
        size();
        invalidate();
    }

    function size(){
        // 코드를 작성하여 크기 조절을 처리합니다 .
    }
}
```

```

        super.size();
        invalidate();
    }

    function draw(){
        // 코드를 작성하여 시각적 표현을 처리합니다.
        super.draw();
    }
}

```

## 구성 요소 클래스 파일 개요

다음은 구성 요소 클래스에 대해 `ActionScript` 파일을 만드는 방법을 보여 주는 일반적인 절차입니다. 만드는 구성 요소의 유형에 따라 일부 단계는 선택적일 수 있습니다.

### 구성 요소 클래스 파일을 작성하려면:

1. (선택 사항) 클래스를 가져옵니다. 자세한 내용은 [141페이지의 “클래스 가져오기”](#)를 참조하십시오.  
이렇게 하면 패키지를 기록하지 않고도 클래스를 참조할 수 있습니다. 예를 들어, `mx.controls.Button` 대신 `Button`을 사용할 수 있습니다.
2. `class` 키워드를 사용하여 클래스를 정의합니다. 부모 클래스를 확장하려면 `extend` 키워드를 사용합니다. 자세한 내용은 [142페이지의 “클래스 및 슈퍼 클래스 정의”](#)를 참조하십시오.
3. `symbolName`, `symbolOwner` 및 `className` 변수를 정의합니다. 자세한 내용은 [142페이지의 “클래스, 심볼 및 소유자 이름 식별”](#)을 참조하십시오.  
이러한 변수는 버전 2 구성 요소에서만 필요합니다.
4. 멤버 변수를 정의합니다. 자세한 내용은 [143페이지의 “변수 정의”](#)를 참조하십시오.  
이러한 변수는 `getter/setter` 메서드에서 사용할 수 있습니다.
5. 생성자 함수를 정의합니다. 자세한 내용은 [158페이지의 “생성자 함수 정의”](#)를 참조하십시오.
6. `init()` 메서드를 정의합니다. 자세한 내용은 [156페이지의 “init\(\) 메서드 정의”](#)를 참조하십시오.  
이 메서드는 클래스가 `UIComponent`를 확장하는 경우 클래스를 만들 때 호출됩니다. 클래스가 `MovieClip`을 확장하는 경우에는 생성자 함수에서 이 메서드를 호출합니다.
7. `createChildren()` 메서드를 정의합니다. 자세한 내용은 [156페이지의 “createChildren\(\) 메서드 정의”](#)를 참조하십시오.  
이 메서드는 클래스가 `UIComponent`를 확장하는 경우 클래스를 만들 때 호출됩니다. 클래스가 `MovieClip`을 확장하는 경우에는 생성자 함수에서 이 메서드를 호출합니다.

8. `size()` 메서드를 정의합니다. 자세한 내용은 159페이지의 “`size()` 메서드 정의”를 참조하십시오.  
이 메서드는 클래스가 `UIComponent`를 확장하는 경우 구성 요소의 크기를 조절할 때 호출됩니다. 또한 제작하는 동안 구성 요소의 실시간 미리 보기 크기를 조절할 때 이 메서드가 호출됩니다.
9. `draw()` 메서드를 정의합니다. 자세한 내용은 160페이지의 “무효화”를 참조하십시오.  
이 메서드는 클래스가 `UIComponent`를 확장하는 경우 구성 요소를 무효화할 때 호출됩니다.
10. 메타데이터 태그와 선언을 추가합니다. 자세한 내용은 144페이지의 “구성 요소 메타데이터 추가”를 참조하십시오.  
태그와 선언을 추가하면 `getter/setter` 속성이 `Flash`의 속성 관리자와 구성 요소 관리자에 나타나게 됩니다.
11. `getter/setter` 메서드를 정의합니다. 자세한 내용은 143페이지의 “`getter/setter` 메서드를 사용하여 매개 변수 정의”를 참조하십시오.
12. (선택 사항) 구성 요소에서 사용하는 모든 스킨 요소/링크의 변수를 만듭니다. 자세한 내용은 162페이지의 “스킨 지정”을 참조하십시오.  
이렇게 하면 사용자가 구성 요소에서 매개 변수를 변경하여 다른 스킨 요소를 설정할 수 있습니다.

## 클래스 가져오기

클래스 파일을 가져오면 코드에 전체 클래스 이름을 기록하지 않아도 되므로 코드가 간결하고 읽기 쉬워집니다. 클래스를 가져오려면 다음과 같이 클래스 파일 맨 위에서 `import` 문을 사용합니다.

```
import mx.core.UIObject;
import mx.core.ScrollView;
import mx.core.ext.UIObjectExtensions;
```

```
class MyComponent extends UIComponent{
```

와일드카드(\*)를 사용하여 지정된 패키지 내의 클래스를 모두 가져올 수도 있습니다.

예를 들어, 다음 명령문은 `mx.core` 패키지의 모든 클래스를 가져옵니다.

```
import mx.core.*;
```

가져온 클래스가 스크립트에서 사용되지 않으면 해당 클래스는 최종 SWF 파일의 바이트코드에 포함되지 않습니다. 따라서 와일드카드를 사용하여 전체 패키지를 가져와도 SWF 파일이 불필요하게 커지지 않습니다.

## 클래스 및 수퍼 클래스 정의

구성 요소 클래스 파일은 일반 클래스 파일처럼 정의됩니다. `class` 키워드를 사용하여 클래스 이름을 나타냅니다. 클래스 이름이 클래스 파일의 이름도 되어야 합니다. `extends` 키워드를 사용하여 수퍼 클래스를 나타냅니다. 자세한 내용은 *Adobe Flash에서 ActionScript 2.0 학습*의 제6장, “사용자 정의 클래스 파일 작성”을 참조하십시오.

```
class MyComponentName extends UIComponent{  
  
}
```

## 클래스, 심볼 및 소유자 이름 식별

Flash에서 적절한 ActionScript 클래스 및 패키지를 찾고 구성 요소의 이름을 보존하도록 하려면 구성 요소의 ActionScript 클래스 파일에 `symbolName`, `symbolOwner` 및 `className` 변수를 설정해야 합니다.

`symbolOwner` 변수는 심볼을 참조하는 Object 참조입니다. 구성 요소가 자체 `symbolOwner` 이거나 `symbolOwner`를 가져온 경우에는 전체 이름이 아니어도 됩니다.

다음 표에서는 이러한 변수에 대해 설명합니다.

변수	유형	설명
<code>symbolName</code>	String	ActionScript 클래스 이름(예: ComboBox)입니다. 이 이름은 심볼의 링크 식별자와 일치해야 합니다. 이 변수는 정적이어야 합니다.
<code>symbolOwner</code>	Object	전체 클래스 이름(예: mypackage.MyComponent)입니다. <code>symbolOwner</code> 값은 Object 데이터 유형이므로 따옴표로 묶지 마십시오. 이 이름은 링크 속성 대화 상자의 AS 2.0 클래스와 일치해야 합니다. 이 변수는 <code>createClassObject()</code> 메서드에 대한 내부 호출에 사용됩니다. 이 변수는 정적이어야 합니다.
<code>className</code>	String	구성 요소 클래스 이름입니다. 이 변수는 패키지 이름을 포함하지 않으며 Flash 개발 환경에 해당 설정이 없습니다. 스타일 속성을 설정할 때 이 변수의 값을 사용할 수 있습니다.

다음 예제에서는 `MyButton` 클래스에 `symbolName`, `symbolOwner` 및 `className` 변수를 추가합니다.

```
class MyButton extends mx.controls.Button {  
    static var symbolName:String = "MyButton";  
    static var symbolOwner = myPackage.MyButton;  
    var className:String = "MyButton";  
}
```

## 변수 정의

다음 코드 샘플은 `Button.as` 파일(`mx.controls.Button`)에서 가져온 것으로, 클래스 파일에서 사용할 `btnOffset` 변수를 정의하며 `__label` 및 `__labelPlacement` 변수도 정의합니다. 뒤의 두 변수에는 `getter/setter` 메서드에서 사용하고 구성 요소에서 속성과 매개 변수로 사용할 때 이름이 충돌하지 않도록 두 개의 밑줄이 앞에 붙습니다. 자세한 내용은 *Adobe Flash에서 ActionScript 2.0 학습*의 “getter/setter 메서드를 사용하여 매개 변수 정의”를 참조하십시오.

```
/**
 * 버튼을 누를 때 레이블 및 / 또는 아이콘을 오프셋하는 데 사용할 숫자
 */
    var btnOffset:Number = 0;

/**
 * @private
 * 값이 지정되지 않은 경우 레이블에 표시할 텍스트
 */
    var __label:String = "default value";

/**
 * @private
 * 기본 레이블 위치
 */
    var __labelPlacement:String = "right";
```

## getter/setter 메서드를 사용하여 매개 변수 정의

가장 간단하게 구성 요소 매개 변수를 정의하는 방법은 매개 변수를 “`Inspectable`”로 만드는 공용 멤버 변수를 추가하는 것입니다. 이 작업은 구성 요소 관리자의 `Inspectable` 태그를 사용하거나 다음과 같이 `Inspectable` 변수를 추가하여 수행할 수 있습니다.

```
[Inspectable(defaultValue="strawberry")]
public var flavorStr:String;
```

하지만 구성 요소를 사용하는 코드에서 `flavorStr` 속성을 수정하는 경우 해당 구성 요소는 대개 속성 변경에 대한 응답으로 자기 자신을 업데이트하는 조치를 취해야 합니다. 예를 들어, `flavorStr`를 “`cherry`”로 설정하면 기본 딸기 이미지 대신 체리 이미지를 사용하여 구성 요소가 다시 그려집니다.

일반 멤버 변수의 경우에는 멤버 변수의 값이 변경되었음을 알리는 메시지가 구성 요소에 자동으로 전달되지 않습니다.

`getter/setter` 메서드를 사용하면 구성 요소 속성의 변경 내용을 쉽게 감지할 수 있습니다.

`var`로 일반 변수를 선언하는 대신 다음과 같이 `getter/setter` 메서드를 선언합니다.

```
private var __flavorStr:String = "strawberry";
```

```
[Inspectable(defaultValue="strawberry")]

public function get flavorStr():String{
    return __flavorStr;
}

public function set flavorStr(newFlavor:String) {
    __flavorStr = newFlavor;
    invalidate();
}
```

invalidate()를 호출하면 새 flavor를 사용하여 구성 요소가 다시 그려집니다. 이것은 flavorStr 속성에 일반 멤버 변수 대신 getter/setter 메서드를 사용할 때 얻을 수 있는 이점입니다. 자세한 내용은 159페이지의 “draw() 메서드 정의”를 참조하십시오.

getter/setter 메서드를 정의하려면 다음 사항을 주의해야 합니다.

- 메서드 이름 앞에 get 또는 set를 붙인 다음 공백과 속성 이름을 차례로 입력합니다.
- 속성 값을 저장하는 변수는 getter 또는 setter와 이름이 달라야 합니다. 규칙에 따라 getter 및 setter 변수 이름 앞에 두 개의 밑줄을 붙여야 합니다.

getter 및 setter는 일반적으로 속성 관리자와 구성 요소 관리자에 표시되는 속성을 정의하는 태그와 함께 사용됩니다.

getter/setter 메서드에 대한 자세한 내용은 *Adobe Flash에서 ActionScript 2.0 학습*의 “getter 및 setter 메서드”를 참조하십시오.

## 구성 요소 메타데이터 추가

외부 ActionScript 클래스 파일에서 구성 요소 메타데이터 태그를 추가하여 컴파일러에 구성 요소 매개 변수, 데이터 바인딩 속성 및 이벤트에 대한 내용을 알릴 수 있습니다. 메타데이터 태그는 Flash 제작 환경에서 다양한 용도로 사용됩니다.

메타데이터 태그는 외부 ActionScript 클래스 파일에서만 사용할 수 있습니다. FLA 파일에서는 메타데이터 태그를 사용할 수 없습니다.

메타데이터는 클래스 선언 또는 개별 데이터 필드와 관련되어 있습니다. 속성 값의 유형이 string이면 속성을 따옴표로 묶어야 합니다.

메타데이터 명령문은 ActionScript 파일의 다음 행에 바인딩됩니다. 구성 요소 속성을 정의할 때는 속성 선언 이전 행에 메타데이터 태그를 추가합니다. 유일한 예외는 Event 메타데이터 태그입니다. 구성 요소 이벤트를 정의할 때는 이벤트가 전체 클래스에 바인딩되도록 클래스 정의 외부에 메타데이터 태그를 추가합니다.

다음 예제에서 Inspectable 태그는 flavorStr, colorStr 및 shapeStr 매개 변수를 정의합니다.

```
[Inspectable(defaultValue="strawberry")]
public var flavorStr:String;
[Inspectable(defaultValue="blue")]
```

```
public var colorStr:String;
[Inspectable(defaultValue="circular")]
public var shapeStr:String;
```

속성 관리자와 구성 요소 관리자의 **매개 변수** 탭에는 이러한 매개 변수가 String 유형으로 표시됩니다.

## 메타데이터 태그

다음 표에서는 ActionScript 클래스 파일에서 사용할 수 있는 메타데이터 태그를 보여 줍니다.

태그	설명
Inspectable	구성 요소 관리자와 속성 관리자에 속성을 표시합니다. 자세한 내용은 <a href="#">146페이지</a> 의 “Inspectable 태그”를 참조하십시오.
InspectableList	속성 관리자와 구성 요소 관리자에 표시되어야 하는 관리 가능 속성의 일부를 식별합니다. 구성 요소의 클래스에 InspectableList 속성을 추가하지 않으면 모든 관리 가능 매개 변수가 속성 관리자에 표시됩니다. 자세한 내용은 <a href="#">148페이지</a> 의 “InspectableList 태그”를 참조하십시오.
Event	구성 요소 이벤트를 정의합니다. 자세한 내용은 <a href="#">148페이지</a> 의 “Event 태그”를 참조하십시오.
Bindable	구성 요소 관리자의 바인딩 탭에 속성을 표시합니다. 자세한 내용은 <a href="#">149페이지</a> 의 “Bindable 태그”를 참조하십시오.
ChangeEvent	데이터 바인딩을 발생시키는 이벤트를 식별합니다. 자세한 내용은 <a href="#">150페이지</a> 의 “ChangeEvent 태그”를 참조하십시오.
Collection	구성 요소 관리자에 표시된 collection 속성을 식별합니다. 자세한 내용은 <a href="#">151페이지</a> 의 “Collection 태그”를 참조하십시오.
IconFile	구성 요소 패널에서 이 구성 요소를 나타내는 아이콘의 파일 이름을 지정합니다. 자세한 내용은 <a href="#">152페이지</a> 의 “IconFile 태그”를 참조하십시오.
ComponentTask	하나 이상의 연관된 JSFL 파일의 이름을 지정하여 제작 환경에서 업무를 수행합니다. 자세한 내용은 <a href="#">153페이지</a> 의 “ComponentTask 태그”를 참조하십시오.

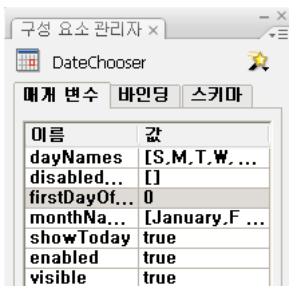
다음 단원에서는 구성 요소 메타데이터 태그에 대해 좀 더 자세히 설명합니다.

## Inspectable 태그

Inspectable 태그를 사용하여 구성 요소 관리자와 속성 관리자에 표시되는 사용자 편집이 가능한(관리 가능) 매개 변수를 지정합니다. 이 매개 변수를 사용하여 관리 가능 속성과 기본 `ActionScript` 코드를 같은 위치에 유지할 수 있습니다. 구성 요소 속성을 보려면 구성 요소의 인스턴스를 스테이지로 드래그하고 구성 요소 관리자에서 **매개 변수** 탭을 선택합니다.

컬렉션 매개 변수도 관리 가능 매개 변수입니다. 자세한 내용은 [151 페이지의 “Collection 태그”](#)를 참조하십시오.

다음 그림에서는 `DateChooser` 구성 요소에 대한 구성 요소 관리자의 **매개 변수** 탭을 보여줍니다.



또는 속성 관리자의 **매개 변수** 탭에서 구성 요소 속성의 일부를 볼 수 있습니다.

Flash에서는 제작 환경에 표시할 매개 변수를 결정할 때 `Inspectable` 태그를 사용합니다. 이 태그의 구문은 다음과 같습니다.

```
[Inspectable(value_type=value[,attribute=value,...])]  
property_declaration name:type;
```

다음 예제에서는 `enabled` 매개 변수를 관리 가능 매개 변수로 정의합니다.

```
[Inspectable(defaultValue=true, verbose=1, category="Other")]  
var enabled:Boolean;
```

`Inspectable` 태그는 다음과 같이 간략하게 입력한 속성도 지원합니다.

```
[Inspectable("danger", 1, true, maybe)]
```

메타데이터 명령문은 속성의 변수 선언 바로 앞에 와야 속성에 바인딩됩니다.

다음 표에서는 Inspectable 태그의 속성에 대해 설명합니다.

특성	유형	설명
defaultValue	String 또는 Number	(선택 사항) 관리 가능 속성의 기본값입니다.
enumeration	String	(선택 사항) 유효한 속성 값 목록을 쉼표로 구분하여 지정합니다.
listOffset	Number	(선택 사항) 이전 버전의 Flash MX 구성 요소와의 호환성을 위해 추가되었습니다. List 값에 대한 기본 인덱스로 사용됩니다.
name	String	(선택 사항) 속성의 표시 이름입니다. 예로 Font Width를 들 수 있습니다. 지정되지 않았으면 <code>_fontWidth</code> 같은 속성 이름을 사용합니다.
type	String	(선택 사항) 유형 식별자입니다. 이 속성을 생략하면 속성의 유형이 사용됩니다. 가능한 값은 다음과 같습니다. <ul style="list-style-type: none"> <li>• Array</li> <li>• Boolean</li> <li>• Color</li> <li>• Font Name</li> <li>• List</li> <li>• Number</li> <li>• Object</li> <li>• String</li> </ul>
variable	String	(선택 사항) 이전 버전의 Flash MX 구성 요소와의 호환성을 위해 추가되었습니다. 이 매개 변수가 바인딩되는 변수를 지정합니다.
verbose	Number	(선택 사항) verbose 속성이 1로 설정된 관리 가능 속성은 속성 관리자에는 표시되지 않고 구성 요소 관리자에 표시됩니다. 이것은 대개 자주 수정되지 않는 속성에 사용됩니다.

이러한 속성은 필요하지 않으며 Inspectable은 메타데이터 태그로 사용할 수 있습니다.

Inspectable로 표시된 슈퍼 클래스의 모든 속성은 자동으로 현재 클래스에서 관리 가능합니다. 현재 클래스에 대해 이러한 속성 중 일부를 숨기려면 InspectableList 태그를 사용합니다.

## InspectableList 태그

`InspectableList` 태그를 사용하여 속성 관리자에 표시되어야 할 관리 가능 속성의 일부를 지정합니다. `InspectableList`를 `Inspectable`과 함께 사용하면 하위 클래스인 구성 요소의 상속된 속성을 숨길 수 있습니다. 구성 요소의 클래스에 `InspectableList` 태그를 추가하지 않으면 속성 관리자에 구성 요소 부모 클래스의 매개 변수를 비롯한 모든 관리 가능 매개 변수가 표시됩니다.

`InspectableList` 구문은 다음과 같습니다.

```
[InspectableList("attribute1"[...])]  
// 클래스 정의
```

`InspectableList` 태그는 전체 클래스에 적용되므로 클래스 정의 바로 앞에 와야 합니다.

다음 예제에서는 속성 관리자에 `flavorStr` 및 `colorStr` 속성이 표시되도록 하지만 `Parent` 클래스에서 다른 관리 가능 속성은 제외시킵니다.

```
[InspectableList("flavorStr", "colorStr")]  
class BlackDot extends DotParent {  
    [Inspectable(defaultValue="strawberry")]  
    public var flavorStr:String;  
    [Inspectable(defaultValue="blue")]  
    public var colorStr:String;  
    ...  
}
```

## Event 태그

`Event` 태그를 사용하여 구성 요소가 내보내는 이벤트를 정의합니다.

이 태그의 구문은 다음과 같습니다.

```
[Event("event_name")]
```

예를 들어, 다음 코드는 `click` 이벤트를 정의합니다.

```
[Event("click")]
```

클래스의 특정 멤버가 아닌 클래스에 이벤트가 바인딩되도록 `Event` 문을 `ActionScript` 파일의 클래스 정의 외부에 추가합니다.

다음 예제에서는 `UIObject` 클래스에 대한 `Event` 메타데이터를 보여 줍니다. 이 메타데이터는 `resize`, `move` 및 `draw` 이벤트를 처리합니다.

```
...  
import mx.events.UIEvent;  
[Event("resize")]  
[Event("move")]  
[Event("draw")]
```

```
class mx.core.UIObject extends MovieClip {
    ...
}
```

특정 인스턴스를 브로드캐스팅하려면 `dispatchEvent()` 메서드를 호출합니다. 자세한 내용은 [160페이지의 “dispatchEvent\(\) 메서드 사용”](#)을 참조하십시오.

## Bindable 태그

데이터 바인딩을 통해 구성 요소가 서로 연결됩니다. 구성 요소 관리자의 **바인딩** 탭을 통해 시각적 데이터 바인딩을 수행할 수 있습니다. 여기에서 구성 요소에 대한 바인딩을 추가하고 보고 제거할 수 있습니다.

모든 구성 요소에 대해 데이터 바인딩을 사용할 수 있지만 데이터 바인딩의 주요 목적은 사용자 인터페이스 구성 요소를 웹 서비스 및 XML 문서 같은 외부 데이터 소스에 연결하는 것입니다. 이러한 데이터 소스는 속성과 함께 구성 요소로 사용할 수 있으며 다른 구성 요소 속성에 바인딩할 수도 있습니다.

`ActionScript` 클래스에서 속성 앞에 `Bindable` 태그를 사용하여 해당 속성이 구성 요소 관리자의 **바인딩** 탭에 표시되도록 합니다. `var` 또는 `getter/setter` 메서드를 사용하여 속성을 선언할 수 있습니다. 속성에 `getter` 메서드와 `setter` 메서드가 모두 있으면 둘 중 하나에만 `Bindable` 태그를 적용합니다.

`Bindable` 태그의 구문은 다음과 같습니다.

```
[Bindable "readonly"|"writeonly",type="datatype"]
```

두 속성 모두 선택 사항입니다.

다음 예제에서는 `flavorStr` 변수를 구성 요소 관리자의 **바인딩** 탭에서 액세스할 수 있는 속성으로 정의합니다.

```
[Bindable]
public var flavorStr:String = "strawberry";
```

Bindable 태그에는 속성의 데이터 유형과 속성에 대한 액세스 유형을 지정하는 세 가지 옵션을 사용할 수 있습니다. 다음 표에서는 이러한 옵션에 대해 설명합니다.

옵션	설명
readonly	구성 요소 관리자에서 바인딩을 만들 때 이 속성을 소스로 사용하는 바인딩만 만들 수 있음을 나타냅니다. 하지만 ActionScript를 사용하여 바인딩을 만들면 그와 같은 제한이 적용되지 않습니다. [Bindable("readonly")]
writable	구성 요소 관리자에서 바인딩을 만들 때 이 속성을 바인딩의 대상으로만 사용할 수 있음을 나타냅니다. 하지만 ActionScript를 사용하여 바인딩을 만들면 그와 같은 제한이 적용되지 않습니다. [Bindable("writable")]
type="datatype"	바인딩되는 속성의 데이터 유형을 나타냅니다. Flash의 나머지 부분에는 선언된 유형이 사용됩니다. 이 옵션을 지정하지 않으면 바인딩되는 속성의 데이터 유형으로 ActionScript 코드에 선언된 속성의 데이터 유형이 사용됩니다. 다음 예제에서는 데이터 바인딩에서 x가 실제로는 Object 유형이지만 이를 DataProvider 유형으로 처리합니다. [Bindable(type="DataProvider")] var x: Object;

모든 구성 요소의 모든 속성을 데이터 바인딩에 사용할 수 있습니다. Bindable 태그는 속성 중에서 구성 요소 관리자에서 바인딩에 사용할 수 있는 속성을 제어할 뿐입니다. 앞에 Bindable 태그가 없는 속성도 데이터 바인딩에 사용할 수 있지만 이 경우 ActionScript를 사용하여 바인딩을 만들어야 합니다.

Bindable 태그는 ChangeEvent 태그를 사용할 때 필요합니다.

Flash 제작 환경에서 데이터 바인딩을 만드는 작업에 대한 자세한 내용은 *Flash 사용 설명서*의 “데이터 바인딩”을 참조하십시오.

## ChangeEvent 태그

ChangeEvent 태그는 데이터 바인딩에서 특정 속성의 값이 변경되면 구성 요소가 이벤트를 생성하도록 지시합니다. 이벤트에 대한 응답으로 해당 속성을 소스로 갖는 모든 바인딩이 실행됩니다. 구성 요소는 구성 요소에서 올바른 ActionScript 코드를 작성한 경우에만 이벤트를 생성합니다. 클래스에서 선언한 Event 메타데이터 목록에 이벤트가 포함되어 있어야 합니다.

var 또는 getter/setter 메서드를 사용하여 속성을 선언할 수 있습니다. 속성에 getter 메서드와 setter 메서드가 모두 있으면 둘 중 하나에만 ChangeEvent 태그를 적용합니다.

ChangeEvent 태그의 구문은 다음과 같습니다.

```
[Bindable]
[ChangeEvent("event")]
property_declaration or getter/setter function
```

다음 예제에서 구성 요소는 바인딩 가능 속성 flavorStr의 값이 변경될 때 change 이벤트를 생성합니다.

```
[Bindable]
[ChangeEvent("change")]
public var flavorStr:String;
```

메타데이터에 지정된 이벤트가 발생하면 속성이 변경되었음을 바인딩에 알리는 작업이 수행됩니다.

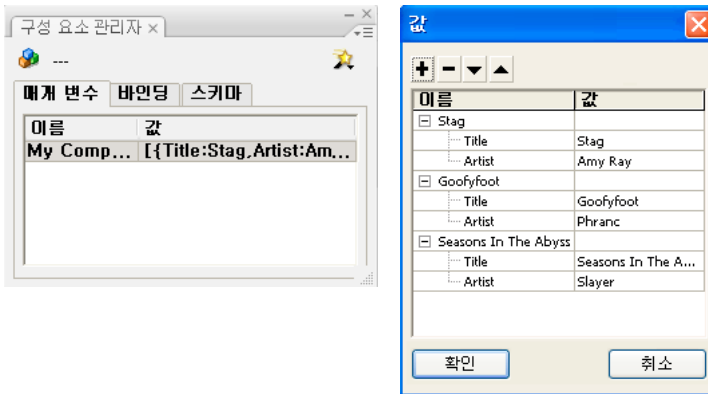
다음 예제와 같이 태그에 여러 이벤트를 등록할 수 있습니다.

```
[ChangeEvent("change1", "change2", "change3")]
```

이러한 이벤트 중 하나는 속성의 변경을 나타냅니다. 속성 변경을 나타내기 위해 모든 이벤트가 발생할 필요는 없습니다.

## Collection 태그

Collection 태그를 사용하여 제작하는 동안 값 대화 상자에서 항목들의 모음으로 수정될 수 있는 객체 배열을 설명합니다. 객체의 유형은 collectionItem 속성에 의해 식별됩니다. 컬렉션 속성에는 별개의 클래스에서 정의하는 컬렉션 항목이 포함됩니다. 이 클래스는 mx.utils.CollectionImpl 또는 하위 클래스입니다. 각 객체는 collectionClass 속성에 의해 식별된 클래스의 메서드를 통해 액세스됩니다.



구성 요소 관리자와 돋보기 아이콘을 클릭하면 나타나는 값 대화 상자에 있는 컬렉션 속성입니다.

Collection 태그의 구문은 다음과 같습니다.

```
[Collection (name="name", variable="varname",
  collectionClass="mx.utils.CollectionImpl",
  collectionItem="coll-item-classname", identifier="string" )]
public var varname:mx.utils.Collection;
```

다음 표에서는 Collection 태그의 속성에 대해 설명합니다.

특성	유형	설명
name	String	(필수 사항) 컬렉션에 대해 구성 요소 관리자에 표시되는 이름입니다.
variable	String	(필수 사항) 기본 Collection 객체를 가리키는 ActionScript 변수입니다. 예를 들어, Collection 매개 변수의 이름으로 Columns를 지정할 수 있지만 기본 variable 속성은 __columns입니다.
collectionClass	String	(필수 사항) 컬렉션 속성에 대해 인스턴스화할 클래스 유형을 지정합니다. 일반적으로 mx.utils.CollectionImpl이지만 mx.utils.CollectionImpl을 확장하는 클래스가 될 수도 있습니다.
collectionItem	String	(필수 사항) 컬렉션 내에 저장할 컬렉션 항목의 클래스를 지정합니다. 이 클래스에는 메타데이터를 통해 표시되는 자체 관리 가능 속성이 있습니다.
identifier	String	(필수 사항) 사용자가 값 대화 상자를 통해 새 컬렉션 항목을 추가할 때 기본 식별자로 사용되는 컬렉션 항목 클래스 내에 있는 관리 가능 속성의 이름을 지정합니다. 사용자가 새 컬렉션 항목을 만들 때마다 항목 이름은 identifier와 고유 인덱스를 결합한 이름으로 설정됩니다. 예를 들어, identifier=name인 경우 값 대화 상자에는 name0, name1, name2 등이 표시됩니다.

자세한 내용은 [181페이지](#)의 “컬렉션 속성”을 참조하십시오.

## IconFile 태그

Flash 제작 환경의 구성 요소 패널에 사용자 구성 요소를 나타내는 아이콘을 추가할 수 있습니다. 자세한 내용은 [178페이지](#)의 “아이콘 추가”를 참조하십시오.

## ComponentTask 태그

Flash 제작 환경 내에서 구성 요소의 업무 수행을 위해 하나 이상의 Flash JavaScript(JSFL) 파일을 지정할 수 있습니다. 그러면 `taskName` 속성에 따라 구성 요소 관리자 마법사 메뉴 (“자동 선택” 클릭)에 작업이 자동으로 표시됩니다. `ComponentTask` 태그를 사용하여 컴퓨터와 JSFL 파일 사이의 연관성을 정의하고 JSFL 파일에 필요한 추가 파일을 연결합니다. JSFL 파일은 Flash 제작 환경에서 JavaScript API와 상호 작용합니다.

예제

`ComponentTask` 태그로 선언된 모든 JSFL 작업 파일 및 필수 종속 파일들은 구성 요소를 SWC 파일로 내보낼 때 구성 요소 FLA 파일과 동일한 폴더에 위치해야 합니다.

`ComponentTask` 태그의 구문은 다음과 같습니다.

```
[ComponentTask [taskName,taskFile [,otherFile[,...]]]
```

`taskName` 및 `taskFile` 속성은 필수입니다. `otherFile` 속성은 선택 사항입니다.

다음 예제에서는 `SetUp.jsfl` 및 `AddNewSymbol.jsfl`를 `myComponent`라는 구성 요소 클래스와 연결합니다. `AddNewSymbol.jsfl`는 `testXML.xml` 파일을 필요로 하며 `otherFile` 속성 내에 지정됩니다.

```
[ComponentTask("Do Some Setup","SetUp.jsfl")]
[ComponentTask("Add a new Symbol","AddNewSymbol.jsfl","testXML.xml")]
class myComponent{
    //...
}
```

다음 표에서는 `ComponentTask` 태그의 속성에 대해 설명합니다.

특성	유형	설명
<code>taskName</code>	String	(필수 사항) 문자열 작업 이름입니다. 이 이름은 구성 요소 관리자의 스크 마 태에 표시되는 작업 팝업 메뉴에 표시됩니다.
<code>taskFile</code>	String	(필수 사항) 제작 환경 내에서 작업을 구현하는 JSFL 파일입니다. 이 파일은 SWC 파일로 구성 요소를 내보낼 때 구성 요소 FLA와 동일한 폴더에 위치해야 합니다.
<code>otherFile</code>	String	(선택 사항) XML 파일과 같이 JSFL 파일이 필요로 하는 하나 이상의 파일 이름입니다. 이 파일은 SWC 파일로 구성 요소를 내보낼 때 구성 요소 FLA와 동일한 폴더에 위치해야 합니다.

## 구성 요소 매개 변수 정의

구성 요소를 작성할 때 구성 요소의 모양과 비헤이비어를 정의하는 매개 변수를 추가할 수 있습니다. 가장 일반적으로 사용되는 매개 변수는 구성 요소 관리자와 속성 관리자에 제작 매개 변수로 표시됩니다. `ActionScript`를 사용하여 모든 관리 가능 및 컬렉션 매개 변수를 설정할 수도 있습니다. `Inspectable` 태그([146페이지의 “Inspectable 태그” 참조](#))를 사용하여 구성 요소 클래스 파일에서 이러한 속성을 정의합니다.

다음 예제에서는 `JellyBean` 클래스 파일에 여러 구성 요소 매개 변수를 설정하고 `Inspectable` 태그를 사용하여 구성 요소 관리자에 해당 매개 변수를 표시합니다.

```
class JellyBean{
    // 문자열 매개 변수
    [Inspectable(defaultValue="strawberry")]
    public var flavorStr:String;

    // 문자열 목록 매개 변수

    [Inspectable(enumeration="sour,sweet,juicy,rotten",defaultValue="sweet")
    ]
    public var flavorType:String;

    // 배열 매개 변수
    [Inspectable(name="Flavors", defaultValue="strawberry,grape,orange",
    verbose=1, category="Fruits")]
    var flavorList:Array;

    // 객체 매개 변수
    [Inspectable(defaultValue="belly:flop,jelly:drop")]
    public var jellyObject:Object;

    // 색상 매개 변수
    [Inspectable(defaultValue="#ffffff")]
    public var jellyColor:Color;

    // setter
    [Inspectable(defaultValue="default text")]
    function set text(t:String) {
    }
}
```

다음 매개 변수 유형 중 하나를 사용할 수 있습니다.

- Array
- Object
- List
- String
- Number
- Boolean
- Font Name
- Color

예제

JellyBean 클래스는 가상의 예제입니다. 실제 예제를 보려면 Flash와 함께 *language/First Run/Classes/mx/controls* 디렉토리에 설치되는 *Button.as* 클래스 파일을 찾아보십시오.

## 기본 함수

구성 요소 클래스 파일에서 `init()`, `createChildren()`, 생성자 함수, `draw()` 및 `size()`의 다섯 함수를 정의해야 합니다. 구성 요소가 `UIComponent`를 확장하면 클래스 파일에 있는 함수가 다음 순서로 호출됩니다.

### ■ `init()`

초기화는 항상 `init()` 함수 호출 시 발생합니다. 예를 들어, 인스턴스 멤버 변수를 이 시기에 설정할 수 있으며 구성 요소 경계 상자를 숨길 수 있습니다.

`init()`가 호출되면 `width` 및 `height` 속성이 자동으로 설정됩니다. 자세한 내용은 [156페이지의 “init\(\) 메서드 정의”](#)를 참조하십시오.

### ■ `createChildren()`

타임라인 안에서 프레임을 재생할 때 호출됩니다. 이 시간 동안 구성 요소 사용자는 메서드와 속성을 호출하여 구성 요소를 설정할 수 있습니다. 구성 요소가 만들어야 하는 하위 객체는 `createChildren()` 함수 내에서 만들어집니다. 자세한 내용은 [156페이지의 “createChildren\(\) 메서드 정의”](#)를 참조하십시오.

### ■ 생성자 함수

구성 요소의 인스턴스를 만들기 위해 호출됩니다. 구성 요소 생성자 함수는 초기화 충돌을 피하기 위해 일반적으로 비워둡니다. 자세한 내용은 [158페이지의 “생성자 함수 정의”](#)를 참조하십시오.

### ■ `draw()`

프로그래밍 방식으로 만들거나 수정하는 구성 요소의 시각적 요소는 `draw` 함수 내에서 발생해야 합니다. 자세한 내용은 [159페이지의 “draw\(\) 메서드 정의”](#)를 참조하십시오.

- `size()`

이 함수는 런타임에 구성 요소의 크기가 조절될 때 호출되며 구성 요소의 업데이트된 `width` 및 `height` 속성을 전달합니다. 구성 요소 하위 객체는 `size()` 함수 내에서 해당 구성 요소의 업데이트된 `width` 및 `height` 속성을 기준으로 크기가 조절되거나 이동됩니다.

자세한 내용은 [159페이지의 “size\(\) 메서드 정의”](#)를 참조하십시오.

이러한 기본 구성 요소 함수는 다음 단원에서 자세히 설명합니다.

## init() 메서드 정의

Flash에서는 클래스가 만들어질 때 `init()` 메서드를 호출합니다. 이 메서드는 구성 요소가 인스턴스화될 때 한 번 호출됩니다.

다음 작업을 수행하려면 `init()` 메서드를 사용해야 합니다.

- `super.init()`를 호출합니다.  
이것은 필수 사항입니다.
- `boundingBox_mc`가 표시되지 않도록 합니다.

```
boundingBox_mc.width = 0;
boundingBox_mc.height = 0;
boundingBox_mc.visible = false;
```
- 인스턴스 멤버 변수를 만듭니다.

`width`, `height` 및 `clip` 매개 변수는 이 메서드가 호출된 후에야 제대로 설정됩니다.

`init()` 메서드가 `UIObject`의 생성자에서 호출되므로 제어 방향은 `UIObject`에 도달할 때까지 생성자 체인을 따라 위로 향합니다. `UIObject`의 생성자는 가장 낮은 하위 클래스에 정의된 `init()` 메서드를 호출합니다. `init()`의 각 구현에서 `super.init()`를 호출해야 해당 기본 클래스가 초기화를 완료할 수 있습니다. `init()` 메서드를 구현하고 `super.init()`를 호출하지 않으면 기본 클래스에 대해 `init()` 메서드가 호출되지 않으므로 사용 가능한 상태가 되지 않습니다.

## createChildren() 메서드 정의

구성 요소는 `createChildren()` 메서드를 구현하여 구성 요소에 하위 객체(예: 다른 구성 요소)를 만듭니다. `createChildren()` 메서드에서 하위 객체의 생성자를 호출하는 대신 `createClassObject()` 또는 `createObject()`를 호출하여 구성 요소의 하위 객체를 인스턴스화합니다.

초기에 모든 자식이 올바른 크기로 설정되도록 `createChildren()` 메서드 내에서 `size()`를 호출하는 것이 좋습니다. 또한 `createChildren()` 메서드 내에서 `invalidate()`를 호출하여 스크린을 새로 고칩니다. 자세한 내용은 [160페이지의 “무효화”](#)를 참조하십시오.

`createClassObject()` 메서드의 구문은 다음과 같습니다.

```
createClassObject(className, instanceName, depth, initObject)
```

다음 표에서는 매개 변수에 대해 설명합니다.

매개 변수	유형	설명
<i>className</i>	Object	클래스 이름입니다.
<i>instanceName</i>	String	인스턴스 이름입니다.
<i>depth</i>	Number	인스턴스 깊이입니다.
<i>initObject</i>	Object	초기화 속성이 포함된 객체입니다.

`createClassObject()`를 호출하려면 `createClassObject()`를 호출할 때 객체의 이름과 유형, 초기화 매개 변수를 지정해야 하기 때문에 자식을 알아야 합니다.

다음 예제에서는 `createClassObject()`를 호출하여 구성 요소 내에서 사용할 새 `Button` 객체를 만듭니다.

```
up_mc.createClassObject(mx.controls.Button, "submit_btn", 1);
```

`createClassObject()`를 호출할 때 속성을 설정하려면 해당 속성을 `initObject` 매개 변수의 일부로 추가합니다. 다음 예제에서는 `label` 속성의 값을 설정합니다.

```
form.createClassObject(mx.controls.CheckBox, "cb", 0, {label:"Check  
this"});
```

다음 예제에서는 `TextInput` 및 `SimpleButton` 구성 요소를 만듭니다.

```
function createChildren():Void {  
    if (text_mc == undefined)  
        createClassObject(TextInput, "text_mc", 0, { preferredWidth: 80,  
            editable:false });  
        text_mc.addEventListener("change", this);  
        text_mc.addEventListener("focusOut", this);  
  
    if (mode_mc == undefined)  
        createClassObject(SimpleButton, "mode_mc", 1, { falseUpSkin:  
            modeUpSkinName, falseOverSkin: modeOverSkinName, falseDownSkin:  
            modeDownSkinName });  
        mode_mc.addEventListener("click", this);  
        size()  
        invalidate()  
}
```

## 생성자 함수 정의

생성자 함수는 구성 요소 클래스와 이름이 동일하므로 쉽게 알 수 있습니다. 예를 들어, 다음 코드는 `ScrollBar` 구성 요소의 생성자 함수를 보여 줍니다.

```
function ScrollBar() {  
}
```

이 경우 새로운 스크롤 막대가 인스턴스화될 때 `ScrollBar()` 생성자가 호출됩니다.

일반적으로 구성 요소 생성자는 비어 있어야 합니다. 생성자에 속성을 설정하면 초기화 호출 순서에 따라 기본값을 덮어쓰는 경우가 있습니다.

구성 요소가 `UIComponent`나 `UIObject`를 확장하면 `init()`, `createChildren()` 및 `size()` 메서드가 자동으로 호출되므로 다음과 같이 생성자 함수를 비워 둘 수 있습니다.

```
class MyComponent extends UIComponent {  
    ...  
    // 생성자 함수입니다.  
    function MyComponent() {  
    }  
}
```

모든 버전 2 구성 요소는 생성자가 호출된 후에 호출되는 `init()` 함수를 정의해야 합니다. 구성 요소의 `init()` 함수에 초기화 코드를 넣어야 합니다. 자세한 내용은 다음 단원을 참조하십시오.

구성 요소가 `MovieClip`을 확장하면 다음 코드 예제에서와 같이 `init()` 메서드, `createChildren()` 메서드, 생성자 함수에서 구성 요소를 배치하는 메서드를 호출할 수 있습니다.

```
class MyComponent extends MovieClip {  
    ...  
    function MyComponent() {  
        init()  
    }  
  
    function init():Void {  
        createChildren();  
        layout();  
    }  
    ...  
}
```

생성자에 대한 자세한 내용은 *Adobe Flash*에서 *ActionScript 2.0* 학습의 “생성자 함수 작성”을 참조하십시오.

## draw() 메서드 정의

draw() 메서드에서 코드를 작성하여 구성 요소의 시각적 요소를 만들거나 수정할 수 있습니다. 즉, draw() 메서드에서는 구성 요소가 자신의 상태 변수와 일치하도록 자기 자신을 그립니다. 마지막 draw() 호출 이후로 여러 속성이나 메서드가 호출되었을 수도 있으므로 draw()의 본문에서는 모든 것을 고려해 보아야 합니다.

하지만 draw() 메서드를 직접 호출하면 안 됩니다. 대신에 invalidate() 메서드를 호출하여 draw()에 대한 호출을 대기시켰다가 일괄적으로 처리할 수 있습니다. 이 방법을 사용하면 효율성이 높아지고 코드가 중앙 집중화됩니다. 자세한 내용은 [160페이지의 “무효화”](#)를 참조하십시오.

draw() 메서드 내에서 Flash 드로잉 API를 호출하여 테두리, 규칙 및 기타 그래픽 요소를 그릴 수 있습니다. 속성 값을 설정하고 메서드를 호출할 수도 있습니다. 보이는 객체를 제거하는 clear() 메서드를 호출할 수도 있습니다.

Dial 구성 요소([123페이지의 “첫 번째 구성 요소 만들기”](#) 참조)를 사용하는 다음 예제에서 draw() 메서드는 바늘의 회전을 value 속성으로 설정합니다.

```
function draw():Void {
    super.draw();
    dial.needle._rotation = value;
}
```

## size() 메서드 정의

componentInstance.setSize() 메서드를 사용하여 런타임에 구성 요소의 크기를 조절하면 size() 함수가 호출되어 width 및 height 속성에 전달됩니다. 구성 요소의 클래스 파일에서 size() 메서드를 사용하여 구성 요소의 내용을 배치할 수 있습니다.

적어도 size() 메서드는 슈퍼 클래스의 size() 메서드(super.size())를 호출해야 합니다.

Dial 구성 요소([123페이지의 “첫 번째 구성 요소 만들기”](#) 참조)를 사용하는 다음 예제에서 size() 메서드는 width 및 height 매개 변수를 사용하여 Dial 무비 클립의 크기를 조절합니다.

```
function size():Void {
    super.size();
    dial._width = width;
    dial._height = height;
    invalidate();
}
```

size() 메서드 내에서 invalidate() 메서드를 호출하여 draw() 메서드를 직접 호출하는 대신 자체를 다시 그리는 태그를 구성 요소에 설정합니다. 자세한 내용은 다음 단원을 참조하십시오.

## 무효화

대부분의 경우 구성 요소 자체를 즉시 업데이트하는 것보다는 새 속성 값의 복사본을 저장한 후 변경 내용을 나타내는 플래그를 설정한 다음 `invalidate()` 메서드를 호출하는 것이 좋습니다. 이 메서드는 객체의 시각적 요소만 변경되었고 하위 객체의 크기와 위치는 변경되지 않았음을 나타냅니다. 이 메서드는 `draw()` 메서드를 호출합니다.

구성 요소를 인스턴스화하는 동안 적어도 한 번은 무효화 메서드를 호출해야 합니다. 이 작업은 `createChildren()` 또는 `layoutChildren()` 메서드에서 가장 많이 수행됩니다.

## 이벤트 전달

구성 요소가 부모 클래스에서 상속되는 이벤트 이외의 이벤트를 브로드캐스팅하도록 하려면 구성 요소의 클래스 파일에서 `dispatchEvent()` 메서드를 호출해야 합니다.

`dispatchEvent()` 메서드는 `mx.events.EventDispatcher` 클래스에 정의되어 있고 `UIObject`를 확장하는 모든 구성 요소에 상속됩니다. 자세한 내용은 *ActionScript 2.0 구성 요소 언어 참조 설명서*의 “`EventDispatcher` 클래스”를 참조하십시오.

또한 클래스 파일 맨 위에 각각의 새 이벤트에 대한 `Event` 메타데이터 태그를 추가해야 합니다. 자세한 내용은 [148페이지](#)의 “`Event` 태그”를 참조하십시오.

예제

Flash 응용 프로그램의 구성 요소 이벤트 처리에 대한 자세한 내용은 [65페이지](#)의 제4장, “구성 요소 이벤트 처리”를 참조하십시오.

## `dispatchEvent()` 메서드 사용

구성 요소의 `ActionScript` 클래스 파일 본문에서 `dispatchEvent()` 메서드를 사용하여 이벤트를 브로드캐스팅합니다. `dispatchEvent()` 메서드의 구문은 다음과 같습니다.

```
dispatchEvent(eventObj)
```

`eventObj` 매개 변수는 이벤트에 대해 설명하는 `ActionScript` 객체입니다(이 단원 뒷부분의 예제 참조).

`dispatchEvent()` 메서드는 다음과 같이 호출 전에 코드에 선언되어야 합니다.

```
private var dispatchEvent:Function;
```

`dispatchEvent()`에 전달할 이벤트 객체도 만들어야 합니다. 이 이벤트 객체에는 리스너가 이벤트를 처리하는 데 사용할 수 있는 이벤트 정보가 있습니다.

다음 예제와 같이 이벤트를 전달하기 전에 이벤트 객체를 명시적으로 작성할 수 있습니다.

```
var eventObj = new Object();
eventObj.type = "myEvent";
eventObj.target = this;
dispatchEvent(eventObj);
```

한 행에서 type 속성과 target 속성의 값을 설정하고 이벤트를 전달하는 간략한 구문을 사용할 수도 있습니다.

```
ancestorSlide.dispatchEvent({type:"revealChild", target:this});
```

앞의 예제에서 target 속성은 암시적이므로 설정하지 않아도 됩니다.

Flash 설명서의 각 이벤트에 대한 설명에서는 선택 또는 필수 이벤트 속성 목록이 나열되어 있습니다. 예를 들어, ScrollBar.scroll 이벤트는 type 및 target 속성뿐 아니라 detail 속성을 받아들입니다. 자세한 내용은 *ActionScript 2.0 구성 요소 언어 참조 설명서*의 이벤트 설명을 참조하십시오.

## 일반 이벤트

다음 표는 다양한 클래스를 통해 브로드캐스팅되는 일반 이벤트 목록입니다. 모든 구성 요소는 필요에 따라 이러한 이벤트를 브로드캐스팅합니다. 이것은 모든 구성 요소에 대한 이벤트의 전체 목록이 아니며 다른 구성 요소에서 재사용될 수 있는 이벤트 목록일 뿐입니다. 일부 이벤트에서 매개 변수를 지정하지 않더라도 모든 이벤트에는 암시적 매개 변수, 즉 이벤트를 브로드캐스팅하는 객체에 대한 참조가 있습니다.

이벤트	용도
click	Button 구성 요소에 의해 사용되거나, 마우스 클릭이 아무 의미가 없을 때 사용됩니다.
change	List, ComboBox 및 기타 텍스트 입력 구성 요소에 의해 사용됩니다.
scroll	ScrollBar 및 스크롤을 일으키는 다른 컨트롤(스크롤 팝업 메뉴의 스크롤 "범퍼")에 의해 사용됩니다.

또한 기본 클래스에서 상속되므로 모든 구성 요소는 다음 이벤트를 브로드캐스팅합니다.

## UIComponent 설명 이벤트

load	구성 요소는 해당 하위 객체를 만들거나 로드합니다.
unload	구성 요소는 해당 하위 객체를 언로드합니다.
focusIn	구성 요소가 입력 포커스를 갖게 됩니다. 일부 HTML 포맷 구성 요소(ListBox, ComboBox, Button, Text)에서도 focus를 브로드캐스팅하지만, 이들 모두 DOMFocusIn을 브로드캐스팅합니다.
focusOut	구성 요소가 입력 포커스를 잃게 됩니다.
move	구성 요소가 새 위치로 이동됩니다.
resize	구성 요소의 크기가 조절됩니다.

다음 표에서는 일반 키 이벤트에 대해 설명합니다.

키 이벤트	설명
keyDown	키를 눌렀습니다. <code>code</code> 속성은 키 코드를 포함하고 <code>ascii</code> 속성은 누른 키의 ASCII 코드를 포함합니다. 이 이벤트가 <code>Key</code> 객체에 의해 생성되지 않았을 수 있으므로 하위 수준의 <code>Key</code> 객체를 사용해 확인하지 마십시오.
keyUp	키를 놓았습니다.

## 스킨 지정

UI(사용자 인터페이스) 구성 요소는 연결된 무비 클립만으로 구성됩니다. 즉, UI 구성 요소에 대한 모든 예셋은 UI 구성 요소 무비 클립 외부에 있을 수 있으므로 다른 구성 요소에서 사용할 수 있습니다. 예를 들어, 구성 요소에 체크 상자 기능이 필요하면 기존 `CheckBox` 구성 요소 예셋을 다시 사용할 수 있습니다.

`CheckBox` 구성 요소는 별개의 무비 클립을 사용하여 각 상태(`FalseUp`, `FalseDown`, `Disabled`, `Selected` 등)를 나타냅니다. 하지만 사용자 정의 무비 클립(스킨)을 이러한 상태에 연결할 수 있습니다. 런타임에 이전 무비 클립과 새 무비 클립은 `SWF` 파일로 내보내집니다. 간단히 이전 상태는 보이지 않게 되고 새 무비 클립이 표시됩니다. 제작하는 동안이나 런타임에 스킨을 변경하는 기능을 *스키닝*이라고 합니다.

구성 요소를 스킨하려면 구성 요소에서 사용하는 모든 스킨 요소(무비 클립 심볼)에 대해 변수를 만들어 심볼의 링크 ID로 설정합니다. 이렇게 하면 다음과 같이 구성 요소에서 매개 변수를 변경하는 것만으로 개발자가 다른 스킨 요소를 설정할 수 있습니다.

```
var falseUpIcon = "mySkin";
```

다음 예제에서는 `CheckBox` 구성 요소의 다양한 상태에 대한 스킨 변수를 보여 줍니다.

```
var falseUpSkin:String = "";
var falseDownSkin:String = "";
var falseOverSkin:String = "";
var falseDisabledSkin:String = "";
var trueUpSkin:String = "";
var trueDownSkin:String = "";
var trueOverSkin:String = "";
var trueDisabledSkin:String = "";
var falseUpIcon:String = "CheckFalseUp";
var falseDownIcon:String = "CheckFalseDown";
var falseOverIcon:String = "CheckFalseOver";
var falseDisabledIcon:String = "CheckFalseDisabled";
var trueUpIcon:String = "CheckTrueUp";
var trueDownIcon:String = "CheckTrueDown";
var trueOverIcon:String = "CheckTrueOver";
var trueDisabledIcon:String = "CheckTrueDisabled";
```

## 스타일

스타일을 사용하여 구성 요소의 모든 그래픽을 클래스에 등록할 수 있습니다. 그러면 해당 클래스가 런타임에 그래픽의 색상 체계를 제어할 수 있습니다. 스타일을 지원하기 위한 구성 요소 구현에는 특수한 코드가 필요하지 않습니다. 스타일은 기본 클래스(UIObject 및 UIComponent)와 스킨에서 완전히 구현됩니다.

구성 요소에 새 스타일을 추가하려면 구성 요소 클래스에서 `getStyle("styleName")`을 호출합니다. 인스턴스, 사용자 정의 스타일 시트 또는 전역 스타일 시트에 스타일을 설정했으면 값이 검색됩니다. 그렇지 않으면 전역 스타일 시트에 스타일의 기본값을 설치해야 합니다.

스타일에 대한 자세한 내용은 [80페이지의 “스타일을 사용하여 구성 요소 색상 및 텍스트 사용자 정의”](#)를 참조하십시오.

## 스타일에 스킨 등록

다음 예제에서는 `Shape`라는 구성 요소를 만듭니다. 이 구성 요소는 원과 정사각형의 두 스킨 모양 중 하나를 표시합니다. 스킨은 `themeColor` 스타일에 등록됩니다.

### 스타일에 스킨을 등록하려면:

1. 새 `ActionScript` 파일을 만들고 다음 코드를 복사해 넣습니다.

```
import mx.core.UIComponent;

class Shape extends UIComponent{

    static var symbolName:String = "Shape";
    static var symbolOwner:Object = Shape;
    var className:String = "Shape";

    var themeShape:String = "circle_skin"

    function Shape(){
    }

    function init(Void):Void{
        super.init();
    }

    function createChildren():Void{
        setSkin(1, themeShape);
        super.createChildren();
    }
}
```

2. `Shape.as`라는 이름으로 파일을 저장합니다.
3. 새 `Flash` 문서를 만들어 `Shape.as`와 동일한 폴더에 `Shape.fla`라는 이름으로 저장합니다.

4. 스테이지에서 원을 그리고 선택한 다음, F8 키를 눌러 무비 클립으로 변환합니다.  
원의 이름과 링크 식별자로 **circle\_skin**을 지정합니다.
  5. circle\_skin 무비 클립을 열고 프레임 1에 다음 ActionScript를 배치하여 themeColor라는 스타일 이름으로 심볼을 등록합니다.  
`mx.skins.ColoredSkinElement.setColorStyle(this, "themeColor");`
  6. 구성 요소의 새 무비 클립을 만듭니다.  
무비 클립과 링크 식별자의 이름을 **Shape**로 지정합니다.
  7. 두 개의 레이어를 만듭니다. 첫 번째 레이어의 첫 번째 프레임에는 stop() 액션을, 두 번째 프레임에는 circle\_skin 심볼을 배치합니다.  
이것은 구성 요소 무비 클립입니다. 자세한 내용은 [134페이지의 “구성 요소 무비 클립 만들기”](#)를 참조하십시오.
  8. StandardComponents.fla를 외부 라이브러리로 열고 UIComponent 무비 클립을 circle\_skin이 있는 Shape 무비 클립의 두 번째 프레임의 스테이지로 드래그합니다.
  9. StandardComponents.fla를 닫습니다.
  10. 라이브러리에서 Shape 무비 클립을 선택하고 **라이브러리** 컨텍스트 메뉴에서 **구성 요소 정의**를 선택하고(Windows: 마우스 오른쪽 버튼으로 클릭, Mac: Control 키를 누른 상태에서 클릭) AS 2.0 클래스 이름으로 **Shape**를 입력합니다.
  11. 스테이지에서 Shape 구성 요소를 사용하여 무비 클립을 테스트합니다.  
테마 색상을 변경하려면 인스턴스에 스타일을 설정합니다. 다음 코드는 인스턴스 이름이 shape인 Shape 구성 요소의 색상을 빨강으로 변경합니다.  
`shape.setStyle("themeColor",0xff0000);`
  12. 스테이지에서 정사각형을 그린 다음 무비 클립으로 변환합니다.  
링크 이름을 **square\_skin**으로 지정하고 **첫 프레임으로 내보내기**가 선택되어 있는지 확인합니다.
- |           |  |
|-----------|--|
| <b>예제</b> | 구성 요소에 무비 클립이 배치되지 않았으므로 첫 프레임으로 내보내기를 선택해야 초기화 전에 스킨을 사용할 수 있습니다. |
|-----------|--|
13. square\_skin 무비 클립을 열고 프레임 1에 다음 ActionScript를 배치하여 themeColor라는 스타일 이름으로 심볼을 등록합니다.  
`mx.skins.ColoredSkinElement.setColorStyle(this, "themeColor");`
  14. 기본 타임라인의 스테이지에 있는 Shape 구성 요소의 인스턴스에 다음 코드를 배치합니다.  
`onClipEvent(initialize){  
    themeShape = "square_skin";  
}`
  15. 스테이지에서 Shape를 사용하여 무비 클립을 테스트합니다. 결과적으로 빨강 정사각형이 표시되어야 합니다.

## 새 스타일 이름 등록

색상 스타일인 새 스타일 이름을 만든 경우 `StyleManager.as` 파일(`First Run\Classes\mx\styles\StyleManager.as`)에서 새 이름을 `colorStyles` 객체에 추가합니다. 이 예제에서는 `shapeColor` 스타일을 추가합니다.

```
// 상속하는 색상 스타일 집합을 초기화합니다.
static var colorStyles:Object =
{
    barColor: true,
    trackColor: true,
    borderColor: true,
    buttonColor: true,
    color: true,
    dateHeaderColor: true,
    dateRollOverColor: true,
    disabledColor: true,
    fillColor: true,
    highlightColor: true,
    scrollTrackColor: true,
    selectedDateColor: true,
    shadowColor: true,
    strokeColor: true,
    symbolBackgroundColor: true,
    symbolBackgroundDisabledColor: true,
    symbolBackgroundPressedColor: true,
    symbolColor: true,
    symbolDisabledColor: true,
    themeColor: true,
    todayIndicatorColor: true,
    shadowCapColor: true,
    borderCapColor: true,
    focusColor: true,
    shapeColor: true
};
```

다음과 같이 각 스킨 무비 클립의 프레임 1에서 원 및 정사각형 스킨에 새 스타일 이름을 등록합니다.

```
mx.skins.ColoredSkinElement.setColorStyle(this, "shapeColor");
```

다음과 같이 인스턴스에 스타일을 설정하여 색상을 새 스타일 이름으로 변경할 수 있습니다.

```
shape.setStyle("shapeColor",0x00ff00);
```

# 기존 구성 요소를 자신의 구성 요소에 통합

이 단원에서는 Label, TextInput 및 Button 구성 요소를 통합하는 간단한 LogIn 구성 요소를 만듭니다. 이 자습 과정에서는 컴파일되지 않은 Flash(FLA) 라이브러리 심볼을 추가하여 새 구성 요소에 기존 구성 요소를 통합하는 방법을 보여줍니다. 완성된 구성 요소 파일 샘플은 Flash 샘플 페이지([www.adobe.com/go/learn\\_fl\\_tutorials\\_kr](http://www.adobe.com/go/learn_fl_tutorials_kr))를 참조하십시오. 다음과 같은 샘플을 사용할 수 있습니다.

- LogIn fla
- LogIn as
- LogIn swf

LogIn 구성 요소는 이름과 암호를 입력하는 인터페이스를 제공합니다. LogIn용 API는 이름 및 암호 TextInput 필드에서 문자열 값을 설정하고 가져오기 위한 두 가지 속성, 즉 name 및 password를 가지고 있습니다. 또한 LogIn 구성 요소는 사용자가 “LogIn”이라는 레이블이 지정된 버튼을 클릭하면 “click” 이벤트를 전달합니다.

- 166페이지의 “LogIn Flash(FLA) 파일 만들기”
- 169페이지의 “LogIn 클래스 파일”
- 172페이지의 “LogIn 구성 요소 테스트 및 내보내기”

## LogIn Flash(FLA) 파일 만들기

먼저 구성 요소 심볼을 보유할 Flash(FLA) 파일을 만듭니다.

### LogIn FLA 파일을 만들려면:

1. Flash에서 **파일 > 새로 만들기**를 선택하고 새 문서를 만듭니다.
2. **파일 > 다른 이름으로 저장**을 선택하고 LogIn fla라는 이름으로 파일을 저장합니다.
3. **삽입 > 새 심볼**을 선택합니다. **LogIn**이라는 이름을 지정하고 **무비 클립** 라디오 버튼을 선택합니다.  
새 심볼 생성 대화 상자의 **링크** 섹션이 열려 있지 않으면 **고급** 버튼을 클릭하여 엽니다.
4. **ActionScript에 내보내기**를 선택하고 **첫 프레임으로 내보내기**는 선택 취소합니다.
5. 링크 식별자를 입력합니다.  
기본 링크 식별자는 LogIn입니다. 이 단계의 나머지 과정에서는 기본값을 사용한다고 가정합니다.
6. **AS 2.0 클래스 텍스트** 상자에 **LogIn**을 입력합니다. 이 값은 구성 요소 클래스 이름입니다. 클래스를 패키지에 넣을 경우에는 전체 패키지 이름을 입력합니다. 예를 들어, mx.controls.CheckBox는 mx.controls 패키지 안의 CheckBox 클래스를 표시합니다.

7. **확인**을 클릭합니다.

Flash가 심볼 편집 모드로 열립니다.

8. 새 레이어를 삽입합니다. 맨 위 레이어의 이름으로 **Actions**를, 맨 아래 레이어의 이름으로 **Assets**를 지정합니다.

9. Assets 레이어에서 프레임 2를 선택하고 키프레임(F6)을 삽입합니다.

이것은 구성 요소 무비 클립의 구조이며 Actions 레이어와 Assets 레이어로 구성됩니다. Actions 레이어에는 한 개의 키프레임이 Assets 레이어에는 두 개의 키프레임이 있습니다.

10. Actions 레이어에서 프레임 1을 선택하고 **액션** 패널(F9)을 엽니다. stop(); 전역 함수를 추가합니다.

이렇게 하면 무비 클립이 프레임 2로 진행하지 않게 됩니다.

11. **파일 > 가져오기 > 외부 라이브러리 열기**를 선택하고 Configuration/ComponentFLA 폴더에서 StandardComponents.flc 파일을 선택합니다.

- Windows의 경우: \Program Files\Adobe\Adobe Flash CS3\language\Configuration\ComponentFLA\StandardComponents.flc
- Macintosh의 경우: HD/Applications/Adobe Flash CS3/Configuration/ComponentFLA/StandardComponents.flc

**예제**

폴더 위치에 대한 자세한 내용은 Flash 사용 설명서의 “Flash와 함께 설치된 Configuration 폴더”를 참조하십시오.

12. Assets 레이어에서 프레임 2를 선택합니다. StandardComponents.flc 라이브러리 내에서 Flash UI Components 2 폴더로 이동합니다. Button, Label 및 TextInput 구성 요소 심볼을 Assets 레이어의 프레임 2로 드래그합니다.

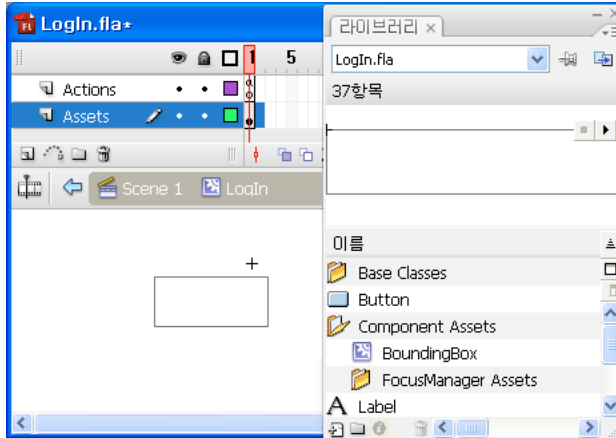
이 세 구성 요소의 예셋 종속 항목들이 자동으로 사용자의 LogIn.flc 라이브러리로 복사됩니다.

모든 구성 요소 예셋은 Assets 레이어의 프레임 2에 추가됩니다. Actions 레이어의 프레임 1에 stop() 전역 함수가 있기 때문에 프레임 2에 있는 예셋은 스테이지에 배열될 때 보이지 않습니다.

다음 두 가지 이유 때문에 프레임 2에 예셋을 추가합니다.

- 모든 예셋을 자동으로 라이브러리로 복사하여 동적으로 인스턴스화하고 해당 메서드, 속성 및 이벤트에 액세스할 수 있습니다.

- 에셋을 프레임에 배치하면 무비가 스트리밍될 때 더욱 매끄럽게 로드되므로 라이브러리 에셋을 첫 프레임에 내보내도록 설정할 필요가 없습니다. 이 메서드로 다운로드 지연 또는 장시간의 정지 상태를 일으키는 데이터 전송의 초기 부담을 피할 수 있습니다.



Button 구성 요소 심볼을 StandardComponents.fla의 라이브러리에서 LogIn.fla의 Assets 레이어 프레임 2로 드래그

13. StandardComponents.fla 라이브러리를 닫습니다.
14. Assets 레이어에서 프레임 1을 선택합니다. BoundingBox 무비 클립을 Component Assets 폴더 내에 있는 LogIn.fla 라이브러리에서 스테이지로 드래그합니다.
15. BoundingBox 인스턴스의 이름을 **boundingBox\_mc**로 지정합니다.
16. 정보 패널을 사용하여 BoundingBox의 크기를 LogInFinal 무비 클립의 크기(340, 150)로 조절하고 0, 0에 배치합니다.

BoundingBox 인스턴스는 제작하는 동안 구성 요소의 실시간 미리 보기를 만들고 크기를 조절하는 데 사용됩니다. 모든 그래픽 요소를 구성 요소에 넣을 수 있도록 경계 상자의 크기를 조절해야 합니다.

**예외** 구성 요소(버전 2 구성 요소 포함)를 확장하는 경우 해당 구성 요소에서 이미 사용 중인 인스턴스 이름이 있으면 해당 코드가 이러한 인스턴스 이름을 참조하기 때문에 유지해야 합니다. 예를 들어, boundingBox\_mc라는 인스턴스 이름을 사용하는 버전 2 구성 요소를 포함할 경우 이 이름을 변경하지 마십시오. 구성 요소를 위해 동일한 범위 내의 기존 이름과 충돌하지 않는 고유한 이름을 인스턴스 이름으로 사용할 수 있습니다.

17. 라이브러리에서 Login 무비 클립을 선택하고 라이브러리 컨텍스트 메뉴에서 **구성 요소 정의**를 선택합니다(Windows: 마우스 오른쪽 버튼으로 클릭, MAC: **Control** 키를 누른 상태에서 클릭).

18. AS 2.0 클래스 텍스트 상자에 **LogIn**을 입력합니다.

이 값은 **ActionScript** 클래스 이름입니다. 클래스가 패키지에 포함된 경우에는 전체 패키지 이름을 입력합니다. 예를 들어, `mx.controls.CheckBox`는 `mx.controls` 패키지 안의 `CheckBox` 클래스를 표시합니다.

19. 확인을 클릭합니다.

20. 파일을 저장합니다.

## LogIn 클래스 파일

다음 코드는 **LogIn** 구성 요소의 **ActionScript** 클래스입니다. 각 섹션에 대한 설명은 코드 주석을 참조하십시오. 구성 요소 클래스의 요소에 대한 자세한 내용은 [140페이지의 “구성 요소 클래스 파일 개요”](#)를 참조하십시오.

이 파일을 만들려면 **Flash**에서 새 **ActionScript** 파일을 만들거나 다른 텍스트 편집기를 사용할 수 있습니다. `LogIn.fla` 파일과 동일한 폴더에 `LogIn.as`라는 이름으로 파일을 저장합니다.

다음의 **LogIn** 구성 요소 **ActionScript** 클래스 코드를 새 `LogIn.as` 파일로 복사하거나 입력할 수 있습니다. 코드를 복사하지 않고 입력하면 구성 요소 코드의 각 요소를 빨리 익힐 수 있습니다.

```
/* 이 클래스에서 직접 참조할 수 있도록
   패키지를 가져옵니다. */
import mx.core.UIComponent;
import mx.controls.Label;
import mx.controls.TextInput;
import mx.controls.Button;

// Event 메타데이터 태그
[Event("change")]
[Event("click")]
class LogIn extends UIComponent
{
    /* 구성 요소는 이러한 멤버 변수가 구성 요소 프레임워크에서
       적절한 구성 요소임을 선언해야 합니다. */
    static var symbolName:String = "LogIn";
    static var symbolOwner:Object = LogIn;
    var className:String = "LogIn";

    // 그래픽으로 표현한 것입니다.
    private var name_label:MovieClip;
    private var password_label:MovieClip;
    private var name_ti:MovieClip;
    private var password_ti:MovieClip;
    private var login_btn:MovieClip;
    private var boundingBox_mc:MovieClip;
    private var startDepth:Number = 10;
```

```

/* getter/setter 를 통해 공개적으로 사용할 수 있는 private 멤버 변수
   name 및 password InputText 문자열 값을 나타냅니다. */
private var __name:String;
private var __password:String;

/* 생성자 :
   모든 클래스에 생성자가 필요하지만 v2 구성 요소의 경우에는
   생성자가 인수 없이 비어 있어야 합니다 .
   모든 모든 초기화는 클래스 인스턴스가 생성된 후
   생성자가 인수 없이 비어 있어야 합니다. */
function LogIn() {
}

/* 초기화 코드 :
   v2 구성 요소에 init 메서드가 필요합니다. 또한 init 메서드는
   super.init() 를 사용하여 해당 부모 클래스 init() 메서드를 호출합니다 .
   UIComponent 를 확장하는 구성 요소에 init 메서드가 필요합니다. */
function init():Void {
    super.init();
    boundingBox_mc._visible = false;
    boundingBox_mc._width = 0;
    boundingBox_mc._height = 0;
}

/* 시작할 때 필요한 자식 객체를 만듭니다 .
   UIComponent 를 확장하는 구성 요소에 createChildren
   메서드가 필요합니다. */
public function createChildren():Void {
    name_label = createObject("Label", "name_label", this.startDepth++);
    name_label.text = "Name:";
    name_label._width = 200;
    name_label._x = 20;
    name_label._y = 10;

    name_ti = createObject("TextInput", "name_ti",
this.startDepth++,{_width:200,_heigh:22,_x:20,_y:30});
    name_ti.html = false;
    name_ti.text = __name;
    name_ti.tabIndex = 1;
    /* 이 이 텍스트 입력 필드가 포커스를 갖도록 설정합니다 .
       참고 : 이미 선택되어 있지 않는 경우 Flash Debugger 에서
       컨트롤 키보드 단축키 비활성을 선택해야 합니다 . 이렇게 하지 않으면
       테스트할 때 포커스가 설정되지 않을 수 있습니다. */
    name_ti.setFocus();

    name_label = createObject("Label", "password_label",
this.startDepth++,{_width:200,_heigh:22,_x:20,_y:60});
    name_label.text = "Password:";

```

```

    password_ti = createObject("TextInput", "password_ti",
this.startDepth++,{_width:200,_heigh:22,_x:20,_y:80});
    password_ti.html = false;
    password_ti.text = __password;
    password_ti.password = true;
    password_ti.tabIndex = 2;

    login_btn = createObject("Button", "login_btn",
this.startDepth++,{_width:80,_heigh:22,_x:240,_y:80});
    login_btn.label = "LogIn";
    login_btn.tabIndex = 3;
    login_btn.addEventListener("click", this);

    size();
}

/* v2 구성 요소에 draw 메서드가 필요합니다 .
draw 메서드는 invalidate()를 호출하는 사용자에
의해 구성 요소가 무효화된 후 호출됩니다 .
이것은 변경 내용을 개별적으로 처리하지 않고
한 번의 그리기로 일괄 처리합니다 . 이를 통해
효율성을 높이고 코드를 중앙 집중화할 수 있습니다 . */
function draw():Void {
    super.draw();
}

/* size 메서드는 구성 요소의 크기가 변경될 때
호출됩니다 . 이를 통해 자식과 Dial 및 박늘
UIComponent를 확장하는 구성 요소에 size 메서드가 필요합니다 . */
function size():Void {
    super.size();
    // 필요한 경우 다시 그려지게 합니다 .
    invalidate();
}

/* 이벤트 핸들러 :
마우스 클릭을 받을 때 LogIn 버튼에서 호출됩니다 .
이 이벤트를 이 구성 요소 범위 밖에서 액세스할 수 있도록
dispatchEvent를 사용하여 click 이벤트를 전달합니다 . */
public function click(evt){
    // 입력 필드 내용으로 멤버 변수를 업데이트합니다 .
    __name = name_ti.text;
    __password = password_ti.text;
    // 버튼이 click 이벤트를 발생시킬 때 해당 이벤트를 전달합니다 .
    dispatchEvent({type:"click"});
}

/* 이는 값 속성에 대한 getter/setter입니다 .
[Inspectable] 메타데이터는 속성 관리자에 속성이

```

```

        나타나게 하고 기본 값을 설정할 수 있도록
        허용합니다. 값이 변경되면 getter/setter 를 사용하여 구성 요소를
        무효화하고 다시 그리도록 할 수 있습니다. */
[Bindable]
[ChangeEvent("change")]
[Inspectable(defaultValue="")]
function set name(val:String){
    __name = val;
    invalidate();
}

function get name():String{
    return(__name);
}

[Bindable]
[ChangeEvent("change")]
[Inspectable(defaultValue="")]
function set password(val:String){
    __password=val;
    invalidate();
}

function get password():String{
    return(__password);
}
}

```

## LogIn 구성 요소 테스트 및 내보내기

그래픽 요소와 기본 클래스 그리고 LogIn 구성 요소의 모든 기능이 포함된 클래스 파일로 이루어진 Flash 파일을 만들었습니다. 이제 구성 요소를 테스트해야 합니다.

작업하면서, 특히 클래스 파일을 작성하는 동안 구성 요소를 테스트하는 것이 가장 좋습니다. 작업하면서 테스트하는 가장 빠른 방법은 구성 요소를 컴파일된 클립으로 변환한 다음 구성 요소의 FLA 파일에서 해당 클립을 사용하는 것입니다.

구성 요소가 완성되면 SWC 파일로 내보냅니다. 자세한 내용은 [175페이지](#)의 “구성 요소 내보내기 및 배포”를 참조하십시오.

### LogIn 구성 요소를 테스트하려면:

1. LogIn.fla 파일의 라이브러리에서 Login 무비 클립을 선택하고 라이브러리 컨텍스트 메뉴에서 **컴파일된 클립으로 변환**을 선택합니다(Windows: 마우스 오른쪽 버튼으로 클릭, Mac: **Control** 키를 누른 상태에서 클릭).

컴파일된 클립이 **LogIn SWF**라는 이름으로 라이브러리에 추가됩니다. 무비 클립은 테스트용으로만 컴파일합니다. 그렇지 않은 경우 **Login** 무비 클립을 내보내려면 이 단원 후반부의 지시 사항을 따릅니다.

**예제** 두세 번 테스트한 적이 있거나 컴파일된 클립을 이미 만든 경우에는 라이브러리 충돌 해결 대화 상자가 표시됩니다. 기존 항목 교체를 선택하여 문서에 새 버전을 추가합니다.

2. **LogIn SWF**를 기본 타임라인의 프레임 1에 있는 스테이지로 드래그합니다(무비 클립 타임라인이 아닌 기본 타임라인, 장면 1이어야 합니다).

**매개 변수** 탭이나 구성 요소 관리자에서 이름 및 암호 속성을 설정할 수 있습니다. 이렇게 하면 아무 것도 입력하지 않은 상태에서도 “이름을 입력하세요”같이 기본 텍스트를 표시할 수 있어 유용합니다. 이름 및 암호 속성을 설정하는 경우 이름 및 암호 **InputText** 하위 구성 요소의 기본 텍스트는 런타임 시 설정한 대로 변경됩니다.

런타임에 **value** 속성을 테스트하려면 스테이지의 **Login** 인스턴스 이름을 **myLogin**으로 지정하고 기본 타임라인의 프레임 1에 다음 코드를 추가합니다.

```
// 로그인 값을 볼 수 있는 텍스트 필드를 만듭니다 .
createTextField("myLoginValues",10,10,10,340,40)
myLoginValues.border = true;
// 로그인 구성 요소 인스턴스의 디스패치된 클릭 이벤트에 대한 이벤트 핸들러에서
function click(evt){
/* 인증이 이루어집니다.
예를 들면 , 이름 및 암호를 인증하여 세션 ID 및 / 또는 권한 역할 속성을
사용자에게 반환하는 웹 서비스에 이름 및 암호가 전달됩니다 . */
myLoginValues.text = "Processing...\r";
myLoginValues.text += "Name: " + myLogin.name + " Password: " +
myLogin.password;
}

myLogin.addEventListener("click",this);
```

3. **컨트롤 > 무비 테스트**를 선택하여 **Flash Player**에서 구성 요소를 테스트합니다.

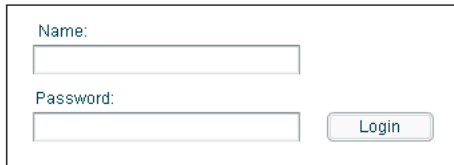
**예제** 이 구성 요소는 원본 문서 내에서 테스트하기 때문에 두 심볼이 동일한 링크 식별자를 가지고 있다는 경고 메시지가 표시될 수도 있습니다. 해당 구성 요소는 계속 작동합니다. 실제로 링크 식별자가 고유해야 하는 경우에도 다른 문서 내에서 새로운 구성 요소를 사용할 수 있습니다.

### LogIn 구성 요소를 내보내려면:

1. LogIn.fla 파일의 라이브러리에서 Login 무비 클립을 선택하고 **라이브러리** 컨텍스트 메뉴에서 **구성 요소 정의**를 선택합니다(Windows: 마우스 오른쪽 버튼으로 클릭, Mac: **Control** 키를 누른 상태에서 클릭).
2. **구성 요소 패널**에 표시를 선택합니다.
3. **확인**을 클릭합니다.
4. LogIn.fla 파일의 라이브러리에서 Login 무비 클립을 선택하고 **라이브러리** 컨텍스트 메뉴에서 **SWC 파일 내보내기**를 선택합니다(Windows: 마우스 오른쪽 버튼으로 클릭, Mac: **Control** 키를 누른 상태에서 클릭).
5. SWC 파일을 저장할 위치를 선택합니다.

사용자 수준의 configuration 폴더에 있는 Components 폴더에 저장하면 Flash를 다시 시작하지 않고도 **구성 요소** 패널을 다시 로드할 수 있으며 이 경우 패널에 구성 요소가 표시됩니다.

<b>예</b> <b>10</b>	폴더 위치에 대한 자세한 내용은 <i>Flash 사용 설명서</i> 의 “Flash와 함께 설치된 Configuration 폴더”를 참조하십시오.
-----------------------	---



The image shows a simple login form. It has two text input fields. The first is labeled 'Name:' and the second is labeled 'Password:'. To the right of the password field is a button labeled 'Login'.

완성된 Login 구성 요소

# 구성 요소 내보내기 및 배포

Flash에서 구성 요소를 구성 요소 패키지(SWC 파일)로 내보냅니다. 구성 요소는 SWC 파일이나 FLA 파일로 배포될 수 있습니다. 구성 요소를 FLA로 배포하는 작업에 대한 자세한 내용은 Adobe DevNet 문서([www.adobe.com/support/flash/applications/creating\\_comps/creating\\_comps12.html](http://www.adobe.com/support/flash/applications/creating_comps/creating_comps12.html))를 참조하십시오.

구성 요소를 배포하는 가장 좋은 방법은 SWC 파일로 배포하는 것으로, 구성 요소를 사용하는 데 필요한 ActionScript, SWF 파일 및 기타 선택적 파일이 모두 SWC 파일에 포함되어 있기 때문입니다. SWC 파일은 구성 요소와 해당 구성 요소를 사용하는 응용 프로그램에 대해 동시에 작업할 경우에도 유용합니다.

SWC 파일을 사용하여 Flash, Macromedia Dreamweaver MX 2004 및 Macromedia Director MX 2004에서 사용할 구성 요소를 배포할 수 있습니다.

자신이 사용할 구성 요소를 개발하던 다른 사람을 위해 구성 요소를 개발하던 구성 요소를 개발하는 과정에서 직접 SWC 파일을 테스트해야 합니다. 예를 들어, FLA 파일에 나타나지 않는 문제가 구성 요소의 SWC 파일에 발생할 수 있습니다.

이 단원에서는 SWC 파일에 대해 설명하고 Flash에서 SWC 파일을 가져오고 내보내는 방법에 대해 설명합니다.

## SWC 파일의 이해

SWC 파일은 Flash 제작 도구로 생성한 zip 형식 파일로, PKZIP 압축 형식으로 압축 및 해제할 수 있습니다.

다음 표에서는 SWC 파일의 내용에 대해 설명합니다.

파일	설명
catalog.xml	(필수) 구성 요소 패키지의 내용과 개별 구성 요소를 나열하며 SWC 파일에 있는 다른 파일에 대하여 디렉토리의 역할을 합니다.
ActionScript (AS) 파일	Flash로 구성 요소를 만드는 경우 구성 요소에 대한 클래스 선언이 포함된 하나 이상의 ActionScript 파일이 소스 코드입니다. 컴파일러는 구성 요소가 확장될 때 소스 코드를 사용하여 유형을 검사합니다. 컴파일된 바이트코드가 이미 구현하는 SWF 파일에 있으므로 AS 파일은 제작 도구를 통해 컴파일되지 않습니다. 소스 코드는 함수 본문이 없고 오로지 유형 검사만을 위해 제공되는 고유한 클래스 정의를 포함할 수 있습니다.
SWF 파일	(필수) 구성 요소를 구현하는 SWF 파일입니다. 단일 SWF 파일에서 하나 이상의 구성 요소를 정의할 수 있습니다. 구성 요소를 Flash 8 이상에서 만든 경우 SWF 파일당 하나의 구성 요소만 내보내집니다.

파일	설명
실시간 미리 보기 SWF 파일	(선택 사항) 이 파일을 지정하면 이러한 SWF 파일이 제작 도구의 실시간 미리 보기에 사용됩니다. SWF 파일을 생략하면 구성 요소를 구현하는 SWF 파일이 실시간 미리 보기에 대신 사용됩니다. 실시간 미리 보기 SWF 파일은 거의 모든 경우에 생략할 수 있습니다. 이 파일은 구성 요소의 모양이 동적 데이터(예: 웹 서비스 호출의 결과를 보여주는 텍스트 필드)에 의존하는 경우에만 필요합니다.
SWD 파일	(선택 사항) 구현하는 SWF 파일에 상응하는 SWD 파일로, SWF 파일을 디버깅하는 데 사용됩니다. 파일 이름은 항상 SWF 파일의 이름과 같지만 확장명은 .swd입니다.
PNG 파일	(선택 사항) 18 x 18 크기의 8비트/픽셀 아이콘이 포함된 PNG 파일로, 제작 도구 사용자 인터페이스에 구성 요소 아이콘을 표시하는 데 사용됩니다. 아이콘을 제공하지 않으면 기본 아이콘이 표시됩니다. 자세한 내용은 <a href="#">178페이지의 “아이콘 추가”</a> 를 참조하십시오.
속성 관리자 SWF 파일	(선택 사항) 제작 도구에서 사용자 정의 속성 관리자로 사용되는 SWF 파일입니다. 이 파일을 생략하면 기본 속성 관리자가 표시됩니다.

Flash 환경에서 SWC 파일을 생성한 후에는 해당 SWC 파일에 다른 파일을 선택적으로 포함시킬 수 있습니다. 예를 들어, 읽어보기 파일을 포함시킬 수 있으며 사용자들이 구성 요소의 소스 파일에 액세스할 수 있도록 FLA 파일을 포함시킬 수도 있습니다. 다른 파일을 추가하려면 Adobe Extension Manager([www.adobe.com/exchange/em\\_download/](http://www.adobe.com/exchange/em_download/) 참조)를 사용합니다.

여러 SWC 파일이 한 디렉토리로 압축이 풀릴 수 있으므로 충돌하지 않도록 각 구성 요소에 고유한 파일 이름을 지정해야 합니다.

## SWC 파일 내보내기

Flash에서는 무비 클립을 SWC 파일로 내보내는 방식으로 SWC 파일을 내보낼 수 있습니다. SWC 파일을 내보내면 Flash 응용 프로그램을 테스트할 때처럼 컴파일 시 오류가 보고됩니다.

SWC 파일을 내보내는 이유는 다음과 같습니다.

- 완성된 구성 요소 배포
- 개발 과정에서 테스트

## 완성된 구성 요소의 SWC 내보내기

구성 요소를 사용하는 데 필요한 ActionScript, SWF 파일 및 기타 선택적 파일이 모두 포함되어 있는 SWC 파일로 구성 요소를 내보낼 수 있습니다.

### 완성된 구성 요소의 SWC 파일을 내보내려면:

1. Flash 라이브러리에서 구성 요소 무비 클립을 선택합니다.
2. 마우스 오른쪽 버튼으로 클릭(Windows)하거나 Control 키를 누른 상태에서 클릭(Mac)하여 라이브러리 컨텍스트 메뉴를 엽니다.
3. 라이브러리 컨텍스트 메뉴에서 **SWC 파일 내보내기**를 선택합니다.
4. SWC 파일을 저장합니다.

## 개발 과정에서 SWC 테스트

개발 과정의 여러 단계에서 구성 요소를 SWC 파일로 내보낸 다음 응용 프로그램에서 테스트 하는 것이 좋습니다. SWC를 사용자 수준의 Configuration 폴더에 있는 Components 폴더로 내보내면 Flash를 종료했다가 다시 시작하지 않고도 구성 요소 패널을 다시 로드할 수 있습니다.

### 개발 과정에서 SWC를 테스트하려면:

1. Flash 라이브러리에서 구성 요소 무비 클립을 선택합니다.
2. 마우스 오른쪽 버튼으로 클릭(Windows)하거나 Control 키를 누른 상태에서 클릭(Mac)하여 라이브러리 컨텍스트 메뉴를 엽니다.
3. 라이브러리 컨텍스트 메뉴에서 **SWC 파일 내보내기**를 선택합니다.
4. 사용자 수준의 Configuration 폴더에 있는 Components 폴더로 이동합니다.  
Configuration/Components

예제

폴더 위치에 대한 자세한 내용은 *Flash 사용 설명서*의 “Flash와 함께 설치된 Configuration 폴더”를 참조하십시오.

5. SWC 파일을 저장합니다.
6. 구성 요소 패널의 옵션 메뉴에서 **다시 로드**를 선택합니다.  
구성 요소 패널에 구성 요소가 표시됩니다.
7. 구성 요소 패널에서 문서로 구성 요소를 드래그합니다.

## 구성 요소 SWC 파일을 Flash로 가져오기

다른 개발자에게 구성 요소를 배포할 때 즉시 설치하여 사용할 수 있도록 다음과 같은 지침을 포함시킬 수 있습니다.

### SWC 파일을 가져오려면:

1. SWC 파일을 Configuration/Components 디렉토리에 복사합니다.
2. Flash를 다시 시작합니다.  
구성 요소의 아이콘이 **구성 요소** 패널에 나타납니다.

## 구성 요소 개발의 마지막 단계

구성 요소를 만들어 패키지화할 수 있게 준비했다면 아이콘과 도구 설명을 추가합니다. 모든 필수 단계를 완료했는지 확인하려면 [179페이지의 “구성 요소 개발 검사 목록”](#)을 참조하십시오.

## 아이콘 추가

Flash 제작 환경의 **구성 요소** 패널에 사용자 구성 요소를 나타내는 아이콘을 추가할 수 있습니다.

### 구성 요소에 대한 아이콘을 추가하려면:

1. 새 이미지를 만듭니다.  
이미지는 18픽셀 크기의 정사각형이며 PNG 포맷으로 저장되어야 합니다. 또한 알파 투명도가 포함된 8비트 색상으로 되어 있고 왼쪽 위 픽셀은 마스킹을 지원할 수 있도록 투명해야 합니다.
2. 구성 요소의 ActionScript 클래스 파일에서 클래스 정의 앞에 다음 정의를 추가합니다.  
`[IconFile("component_name.png")]`
3. FLA 파일과 동일한 디렉토리에 이미지를 추가합니다. SWC 파일을 내보낼 경우 압축 파일의 루트에 해당 이미지가 포함됩니다.

## 도구 설명 추가

도구 설명은 사용자가 Flash 제작 환경의 **구성 요소** 패널에 있는 구성 요소 이름이나 아이콘 위로 마우스를 이동할 때 나타납니다.

**구성 요소 정의** 대화 상자에서 도구 설명을 정의합니다. 구성 요소 FLA 파일의 **라이브러리** 옵션 메뉴에서 이 대화 상자에 액세스 할 수 있습니다(Windows: 마우스 오른쪽 버튼으로 클릭, Mac: **Control** 키를 누른 상태에서 클릭).

### 구성 요소 정의 대화 상자에서 도구 설명을 추가하려면:

1. Flash에서 구성 요소의 FLA 파일을 열어 놓은 상태에서 라이브러리가 표시되는지 확인해야 합니다(윈도우 > 라이브러리 메뉴).
2. 라이브러리 옵션 메뉴를 클릭합니다(Windows: 마우스 오른쪽 버튼으로 클릭, Mac: **Control** 키를 누른 상태에서 클릭).  
라이브러리 옵션 메뉴는 라이브러리 제목 막대 오른쪽에 있으며 세 줄이 있는 역삼각형 모양의 아이콘으로 나타납니다.
3. **구성 요소 정의** 옵션을 선택합니다.
4. **구성 요소 정의** 대화 상자의 **옵션** 아래에서 **구성 요소 패널에 표시**를 선택합니다.  
도구 설명 텍스트 상자가 편집 가능한 상태로 바뀝니다.
5. **도구 설명** 텍스트 상자에 구성 요소의 도구 설명 텍스트를 입력합니다.
6. **확인**을 클릭하여 변경 내용을 저장합니다.

## 구성 요소 개발 검사 목록

구성 요소를 디자인할 때는 다음 사항에 주의하십시오.

- 파일 크기를 최대한 작게 유지합니다.
- 기능을 일반화하여 구성 요소를 최대한 재사용할 수 있도록 만듭니다.
- 그래픽 요소 대신 `RectBorder` 클래스(`mx.skins.halo.RectBorder`)를 사용하여 객체 주위에 테두리를 그립니다. 자세한 내용은 *ActionScript 2.0 구성 요소 언어 참조 설명서*의 “`RectBorder` 클래스”를 참조하십시오.
- 태그 기반 스키닝을 사용합니다.
- `symbolName`, `symbolOwner` 및 `className` 변수를 정의합니다.
- 초기 상태를 가정합니다. 이제 스타일 속성이 객체에 지정되므로, 스타일 및 속성의 초기 값을 지정하여 사용자가 기본 상태를 재정의하지 않는 한, 객체가 생성될 때 이러한 설정 값을 지정할 필요가 없게 할 수 있습니다.

- 심볼을 정의할 때는 반드시 필요한 경우에만 **첫 프레임으로 내보내기** 옵션을 선택하십시오. Flash에서는 구성 요소를 Flash 응용 프로그램에서 사용되기 직전에 로드하므로 이 옵션을 선택하면 부모 구성 요소의 첫 번째 프레임에서 해당 구성 요소가 미리 로드됩니다. 첫 번째 프레임에서 구성 요소를 미리 로드하지 않는 일반적인 이유는 웹에서의 상황을 고려하기 때문입니다. 웹에서는 프리로더가 시작되기 전에 구성 요소가 로드되므로 프리로더가 필요 없게 됩니다.
- 무비 클립에 여러 프레임이 포함되지 않게 합니다(두 개의 프레임이 있는 Assets 레이어 제외).
- 항상 `init()` 및 `size()` 메서드를 구현하고 각각 `Super.init()` 및 `Super.size()`를 호출합니다. 그렇지 않으면 간단한 메서드로 유지합니다.
- `_root.myVariable` 같은 절대 참조의 사용을 피합니다.
- `attachMovie()` 대신 `createClassObject()`를 사용합니다.
- `draw()`를 명시적으로 호출하는 대신 `invalidate()` 및 `invalidateStyle()`을 사용하여 `draw()` 메서드를 호출합니다.
- Flash 구성 요소를 자신의 구성 요소에 통합할 때는 `Configuration/ComponentFLA` 폴더의 `StandardComponents.fla` 파일의 라이브러리에 있는 컴파일되지 않은 무비 심볼을 사용합니다.

## 컬렉션 속성

Flash에서 새 사용자 정의 구성 요소를 만들면 사용자가 편집하는 데 사용할 수 있는 속성 값을 만들 수 있습니다. 이러한 속성을 **컬렉션 속성**이라고 합니다. 이 속성 값은 값 대화 상자(구성 요소의 **매개 변수** 탭에 있는 텍스트 상자에서 열림)에서 편집할 수 있습니다.

구성 요소에는 대개 구성 요소 사용자의 다양한 요구 사항을 수용하기 위한 유연성을 유지하면서 특정 작업을 처리할 수 있도록 일련의 기능이 포함되어 있습니다. 구성 요소가 유연하려면 구성 요소 내에 표시되는 속성이 유연해야 합니다(즉, 일부 구성 요소의 경우 구성 요소 사용자가 속성 값은 물론 속성 자체를 변경할 수도 있습니다).

컬렉션 속성을 사용하면 객체 모델에서 확정되지 않은 개수의 편집할 수 있는 속성을 만들 수 있습니다. Flash에서 제공하는 **Collection** 클래스를 사용하면 구성 요소 관리자를 통해 이러한 속성을 손쉽게 관리할 수 있습니다.

구체적으로 **Collection** 클래스는 각각 **컬렉션 항목**이라고 하는 관련 객체 그룹을 관리하는 데 사용되는 도우미 클래스입니다. 구성 요소의 속성을 컬렉션 항목으로 정의하여 사용자가 구성 요소 관리자를 통해 사용할 수 있도록 만들면 해당 사용자는 제작하는 동안 값 대화 상자에서 컬렉션 항목을 추가, 삭제 및 수정할 수 있습니다.



다음과 같이 컬렉션 및 컬렉션 항목을 정의합니다.

- 구성 요소의 ActionScript 파일에서 Collection 메타데이터 태그를 사용하여 컬렉션 속성을 정의합니다. 자세한 내용은 [151페이지](#)의 “Collection 태그”를 참조하십시오.
- 고유의 관리 가능 속성이 있는 각 ActionScript 파일에서 컬렉션 항목을 클래스로 정의합니다.

Flash에서 컬렉션을 사용하면 관련 항목의 그룹을 프로그래밍 방식으로 관리할 수 있습니다. 이전 버전의 Flash에서는 구성 요소 제작자가 프로그래밍 방식으로 동기화된 여러 배열을 통해 관련 항목의 그룹을 관리했습니다.

값 대화 상자 외에도 Flash는 프로그래밍 방식으로 Collection 인스턴스와 값을 관리하기 위한 Collection 및 Iterator 인터페이스를 제공합니다. 자세한 내용은 *ActionScript 2.0 구성 요소 언어 참조 설명서*의 “Collection 인터페이스” 및 “Iterator 인터페이스”를 참조하십시오.

이 장에서 설명하는 단원은 다음과 같습니다.

컬렉션 속성 정의 .....	182
간단한 컬렉션 예제 .....	183
컬렉션 항목의 클래스 정의 .....	185
프로그래밍 방식으로 컬렉션 정보 액세스 .....	185
컬렉션이 있는 구성 요소를 SWC 파일로 내보내기 .....	188
컬렉션 속성이 있는 구성 요소 사용 .....	189

## 컬렉션 속성 정의

구성 요소의 ActionScript 파일에서 Collection 태그를 사용하여 컬렉션 속성을 정의합니다. 자세한 내용은 [151페이지](#)의 “Collection 태그”를 참조하십시오.



이 단원에서는 구성 요소와 관리 가능 구성 요소 속성을 만드는 방법을 알고 있다고 가정합니다.

### 컬렉션 속성을 정의하려면:

1. 구성 요소의 FLA 파일을 만듭니다. 자세한 내용은 [134페이지](#)의 “구성 요소 무비 클립 만들기”를 참조하십시오.
2. ActionScript 클래스를 만듭니다. 자세한 내용은 [138페이지](#)의 “ActionScript 클래스 파일 만들기”를 참조하십시오.
3. ActionScript 클래스에 Collection 메타데이터 태그를 삽입합니다. 자세한 내용은 [151페이지](#)의 “Collection 태그”를 참조하십시오.
4. 구성 요소의 ActionScript 파일에서 컬렉션의 get 및 set 메서드를 정의합니다.

5. 윈도우 > 공용 라이브러리 > 클래스를 선택하고 UtilsClasses를 구성 요소 라이브러리로 드래그하여 FLA 파일에 유틸리티 클래스를 추가합니다.

UtilsClasses에는 Collection 인터페이스용 mx.utils.\* 패키지가 들어 있습니다.

예제

UtilsClasses는 ActionScript 클래스가 아니라 FLA 파일과 연관되기 때문에 구성 요소의 ActionScript 클래스를 보는 동안 구문을 검사하면 컴파일러 오류가 발생합니다.

6. 컬렉션 항목 속성이 있는 클래스를 코딩합니다.

자세한 내용은 [185페이지](#)의 “컬렉션 항목의 클래스 정의”를 참조하십시오.

## 간단한 컬렉션 예제

다음은 MyShelf.as라는 구성 요소 클래스 파일의 간단한 예제입니다. 이 예제에는 UIObject 클래스로부터 상속 받은 구성 요소에 대하여 컬렉션 속성과 함께 최소한의 가져오기, 메서드 및 선언이 들어 있습니다.

이 예제에서 mx.utils.\*를 가져오면 mx.utils의 클래스 이름은 더 이상 전체 이름이 아니어도 됩니다. 예를 들어, mx.utils.Collection을 Collection으로 기록할 수 있습니다.

```
import mx.utils.*;
// 표준 클래스 선언
class MyShelf extends mx.core.UIObject
{
    // 모든 클래스에 필요한 변수
    static var symbolName:String = "MyShelf";
    static var symbolOwner:Object = Object(MyShelf);
    var className:String = "MyShelf";

    // Collection 메타데이터 태그 및 속성
    [Collection(variable="myCompactDiscs",name="My Compact
    Discs",collectionClass="mx.utils.CollectionImpl",
    collectionItem="CompactDisc", identifier="Title")]

    // 컬렉션에 대한 get 및 set 메서드
    public function get MyCompactDiscs():mx.utils.Collection
    {
        return myCompactDiscs;
    }
    public function set MyCompactDiscs(myCDs:mx.utils.Collection):Void
    {
        myCompactDiscs = myCDs;
    }

    // 전용 클래스 멤버
    private var myCompactDiscs:mx.utils.Collection;
```

```

// 컴파일러가 SWC 내에 컬렉션 항목 클래스를
// 종속으로 포함하도록 그에 대한 참조를 코딩해야
// 합니다.
private var collItem:CompactDisc;

// 컴파일러가 SWC 내에 mx.utils.CollectionImpl 클래스를
// 종속으로 포함하도록 그에 대한 참조를 코딩해야
// 합니다.
private var coll:mx.utils.CollectionImpl;

// 모든 클래스에 필요한 메서드
function init(Void):Void {
    super.init();
}
function size(Void):Void {
    super.size();
}
}

```

### 테스트 목적으로 이 클래스와 함께 제공할 FLA 파일을 만들려면:

1. Flash에서 **파일 > 새로 만들기**를 선택하고 Flash 문서를 만듭니다.
2. **삽입 > 새 심볼**을 선택합니다. 심볼 이름은 링크 식별자로 지정하고, AS 2.0 클래스 이름은 **MyShelf**로 지정합니다.
3. **첫 프레임으로 내보내기**의 선택을 취소하고 **확인**을 클릭합니다.
4. 라이브러리에서 MyShelf 심볼을 선택하고 **라이브러리 옵션 메뉴**에서 **구성 요소 정의**를 선택합니다. ActionScript 2.0 클래스 이름으로 **MyShelf**를 입력합니다.
5. **윈도우 > 공용 라이브러리 > 클래스**를 선택하고 MyShelf fla의 라이브러리로 UtilsClasses를 드래그합니다.
6. MyShelf 심볼의 타임라인에서 한 레이어의 이름을 **Assets**로 지정합니다. 다른 레이어를 만들고 이름을 **Actions**로 지정합니다.
7. Actions 레이어의 프레임 1에 stop() 함수를 배치합니다.
8. Assets 레이어의 프레임 2를 선택하고 **삽입 > 타임라인 > 키프레임**을 선택합니다.
9. Configuration/ComponentFLA 폴더에서 StandardComponents fla를 열고 UIObject의 인스턴스를 Assets 레이어의 프레임 2에 있는 MyShelf의 스테이지로 드래그합니다. 위 클래스 파일에서 알 수 있듯이 MyShelf가 UIObject를 확장하므로 구성 요소의 FLA 파일에 UIObject를 포함시켜야 합니다.
10. Assets 레이어의 프레임 1에서 선반을 그립니다.  
이것은 학습 목적으로 사용할 MyShelf 구성 요소의 시각적 표현일 뿐이므로 간단한 사각형이면 됩니다.

11. 라이브러리에서 MyShelf 무비 클립을 선택하고 **컴파일된 클립으로 변환**을 선택합니다. 이렇게 하면 MyShelf SWF 파일(라이브러리에 추가된 컴파일된 클립)을 MyShelf.fla 파일로 드래그하여 구성 요소를 테스트할 수 있습니다. 이미 라이브러리에 이전 버전의 구성 요소가 있으므로 구성 요소를 다시 컴파일할 때마다 항상 **라이브러리 충돌 해결** 대화 상자가 나타납니다. 기존 항목을 바꾸도록 선택합니다.

예제 10

이미 CompactDisc 클래스를 만들었어야 합니다. 그렇지 않으면 컴파일된 클립으로 변환할 때 컴파일러 오류가 발생합니다.

## 컬렉션 항목의 클래스 정의

컬렉션 항목 속성은 별도의 ActionScript 클래스에 코딩하며 다음과 같이 정의합니다.

- UIObject 또는 UIComponent를 확장하지 않도록 클래스를 정의합니다.
- Inspectable 태그를 사용하여 모든 속성을 정의합니다.
- 모든 속성을 변수로 정의합니다. get 및 set(getter/setter) 메서드를 사용하면 안 됩니다.

다음은 CompactDisc.as라는 컬렉션 항목 클래스 파일의 간단한 예제입니다.

```
class CompactDisc{
    [Inspectable(type="String", defaultValue="Title")]
    var title:String;
    [Inspectable(type="String", defaultValue="Artist")]
    var artist:String;
}
```

CompactDisc.as 클래스 파일을 보려면 [183페이지의 “간단한 컬렉션 예제”](#)를 참조하십시오.

## 프로그래밍 방식으로 컬렉션 정보 액세스

Flash에서는 Collection 및 Iterator 인터페이스를 통해 프로그래밍 방식으로 컬렉션 데이터에 액세스할 수 있습니다. Collection 인터페이스를 사용하면 컬렉션에 항목을 추가, 수정 및 제거할 수 있습니다. Iterator 인터페이스를 사용하면 컬렉션의 항목을 하나씩 차례로 반복할 수 있습니다.

Collection 및 Iterator 인터페이스를 사용하는 시나리오에는 다음 두 가지가 있습니다.

- [186페이지의 “구성 요소 클래스\(AS\) 파일에서 컬렉션 정보 액세스”](#)
- [187페이지의 “Flash 응용 프로그램에서 런타임에 컬렉션 항목 액세스”](#)

고급 개발자가 프로그래밍 방식으로 컬렉션을 만들어 채우고 액세스하고 삭제할 수도 있습니다. 자세한 내용은 *ActionScript 2.0 구성 요소 언어 참조 설명서*의 “Collection 인터페이스”를 참조하십시오.

## 구성 요소 클래스(AS) 파일에서 컬렉션 정보 액세스

구성 요소 클래스 파일에서 제작하는 동안이나 런타임에 정의된 컬렉션 항목과 상호 작용하는 코드를 작성할 수 있습니다.

구성 요소 클래스 파일에서 컬렉션 항목 정보에 액세스하려면 다음 방법 중 하나를 사용합니다.

- **Collection** 태그에는 `mx.utils.Collection` 유형의 변수를 지정하는 `variable` 속성이 포함되어 있습니다. 이 예제와 같이 이 변수를 사용하여 컬렉션에 액세스합니다.

```
[Collection(name="LinkButtons", variable="__linkButtons",
  collectionClass="mx.utils.CollectionImpl", collectionItem="ButtonC",
  identifier="ButtonLabel")]
public var __linkButtons:mx.utils.Collection;
```

- 이 예제와 같이 `Collection.iterator()` 메서드를 호출하여 컬렉션 항목의 반복기 인터페이스에 액세스합니다.

```
var itr:mx.utils.Iterator = __linkButtons.iterator();
```

- **Iterator** 인터페이스를 사용하여 컬렉션의 항목을 하나씩 차례로 반복합니다. `Iterator.next()` 메서드가 `Object`를 반환하므로 이 예제와 같이 컬렉션 항목 유형을 정의해야 합니다.

```
while (itr.hasNext()) {
  var button:ButtonC = ButtonC(itr.next());
  ...
}
```

- 이 예제와 같이 응용 프로그램에 맞게 컬렉션 항목 속성에 액세스합니다.

```
item.label = button.ButtonLabel;

if (button.ButtonLink != undefined) {
  item.data = button.ButtonLink;
}
else {
  item.enabled = false;
}
```

# Flash 응용 프로그램에서 런타임에 컬렉션 항목 액세스

Flash 응용 프로그램에서 컬렉션 속성이 있는 구성 요소를 사용하는 경우 런타임에 컬렉션 속성에 액세스할 수 있습니다. 이 예제에서는 **값** 대화 상자를 사용하여 컬렉션 속성에 여러 항목을 추가하고 Collection 및 Iterator API를 사용하여 런타임에 해당 항목을 표시합니다.

## 런타임에 컬렉션 항목에 액세스하려면:

1. 앞서 만든 MyShelf.fla 파일을 엽니다.  
자세한 내용은 [183페이지의 “간단한 컬렉션 예제”](#)를 참조하십시오.  
이 예제는 MyShelf 구성 요소와 CompactDisc 컬렉션을 기반으로 합니다.
2. 라이브러리 패널을 열고 구성 요소를 스테이지로 드래그한 다음 인스턴스 이름을 지정합니다.  
이 예제에서는 인스턴스 이름으로 myShelf를 사용합니다.
3. 구성 요소를 선택하고 구성 요소 관리자를 연 다음 **매개 변수** 탭을 표시합니다. 컬렉션 속성이 있는 행을 클릭하고 해당 행 오른쪽에 있는 돋보기 아이콘을 클릭합니다. **값** 대화 상자가 표시됩니다.
4. **값** 대화 상자를 사용하여 컬렉션 속성에 값을 입력합니다.
5. 스테이지에서 구성 요소를 선택한 상태에서 **액션** 패널을 열고 다음 코드(구성 요소에 첨부되어 있어야 함)를 입력합니다.

```
onClipEvent (mouseDown) {  
    import mx.utils.Collection;  
    import mx.utils.Iterator;  
    var myColl:mx.utils.Collection;  
    myColl = _parent.myShelf.MyCompactDiscs;  
  
    var itr:mx.utils.Iterator = myColl.getIterator();  
    while (itr.hasNext()) {  
        var cd:CompactDisc = CompactDisc(itr.next());  
        var title:String = cd.Title;  
        var artist:String = cd.Artist;  
        trace("Title: " + title + " Artist: " + artist);  
    }  
}
```

컬렉션에 액세스하려면 `componentName.collectionVariable` 구문을 사용합니다.

반복기에 액세스하여 컬렉션 항목을 하나씩 차례로 반복하려면

`componentName.collectionVariable.getIterator()`를 사용합니다.

6. **컨트롤 > 무비 테스트**를 선택하고 선반을 클릭하여 **출력** 패널에서 컬렉션 데이터를 확인합니다.

# 컬렉션이 있는 구성 요소를 SWC 파일로 보내기

컬렉션이 있는 구성 요소를 배포할 때 SWC 파일에는 다음 종속 파일이 있어야 합니다.

- Collection 인터페이스
- 컬렉션 구현 클래스
- 컬렉션 항목 클래스
- Iterator 인터페이스

개발자 코드에서는 대개 이들 파일 중 Collection 및 Iterator 인터페이스를 사용합니다. Collection 및 Iterator 인터페이스는 종속 클래스로 표시됩니다. Flash에서는 종속 파일이 SWC 파일과 출력 SWF 파일에 자동으로 포함됩니다.

하지만 컬렉션 구현 클래스(`mx.utils.CollectionImpl`)와 구성 요소에 한정된 컬렉션 항목 클래스는 SWC 파일에 자동으로 포함되지 않습니다.

컬렉션 구현 클래스와 컬렉션 항목 클래스를 SWC 파일에 포함시키려면 다음 예제와 같이 구성 요소의 ActionScript 파일에서 전용 변수를 정의합니다.

```
// 컬렉션 항목 클래스
private var collItem:CompactDisc;
// 컬렉션 구현 클래스
private var coll:mx.utils.CollectionImpl;
```

SWC 파일에 대한 자세한 내용은 [175페이지의 “SWC 파일의 이해”](#)를 참조하십시오.

# 컬렉션 속성이 있는 구성 요소 사용

컬렉션 속성이 있는 구성 요소를 사용할 때는 대개 값 대화 상자를 사용하여 컬렉션의 항목을 설정합니다.

## 컬렉션 항목이 있는 구성 요소를 사용하려면:

1. 구성 요소를 스테이지에 추가합니다.
2. 속성 관리자를 사용하여 구성 요소 인스턴스의 이름을 지정합니다.
3. 구성 요소 관리자를 열고 **매개 변수** 탭을 표시합니다.
4. 컬렉션 속성이 있는 행을 클릭하고 해당 행 오른쪽에 있는 돋보기 아이콘을 클릭합니다. 값 대화 상자가 표시됩니다.
5. **추가(+)** 버튼을 클릭하고 컬렉션 항목을 정의합니다.
6. **추가(+)**, **삭제(-)** 및 화살표 버튼을 클릭하여 컬렉션 항목을 추가, 수정, 이동 및 삭제합니다.
7. **확인**을 클릭합니다.

프로그래밍 방식으로 컬렉션에 액세스하는 방법에 대한 자세한 내용은 [187페이지의 “Flash 응용 프로그램에서 런타임에 컬렉션 항목 액세스”](#)를 참조하십시오.



# 색 인

## 가

- 값 대화 상자 181
- 격자. DataGrid 구성 요소 참조
- 고급 엔터엘리어싱은 지원되지 않음 19
- 관리 가능 매개 변수 146
- 구성 요소 138
  - 개발 검사 목록 179
  - 구성 요소 만들기 예제 123, 166
  - 구조 122
  - 내보내기 및 배포 175
  - 도구 설명 추가 179
  - 런타임에 추가 50
  - 로드 62
  - 매개 변수 정의 154
  - 메타데이터 태그 144
  - 메타데이터, ComponentTask 태그 153
  - 무비 클립 만들기 134
  - 무비 클립 편집 135
  - 무효화 160
  - 미리 로드 61
  - 미리 보기 60
  - 범주, 설명 16
  - 변수 정의 143
  - 부모 클래스 선택 131
  - 삭제 55
  - 상속 18
  - 설치 12
  - 소스 파일 121
  - 스킨 지정 162
  - 스타일 163
  - 스타일에 스킨 등록 163
  - 심볼 이름 선택 142
  - 아이콘 추가 178
  - 아키텍처 17
  - 응용 프로그램에서 사용(자습 과정) 21
  - 이벤트 65
  - 이벤트 전달 160
  - 일반 이벤트 161
  - 컬렉션이 있는 클래스 파일의 예제 183
  - 클래스 개요 140
  - 클래스 파일 예제 139
  - 클래스 확장 133
  - ActionScript 클래스 138
  - className 변수 142
  - draw() 메서드 정의 159
  - Flash 문서에 추가 48
  - Flash MX 버전에서 사용 가능 12
  - Flash Player 지원 17
  - getter/setter 메서드 143
  - init() 메서드 정의 156
  - size() 메서드 정의 159
  - SWC 파일 가져오기 178
  - SWC 파일 내보내기 176
  - SWC 파일 테스트 177
  - SWC로 구성 요소 내보내기 177
  - symbolOwner 변수 142
  - 개별 구성 요소 이름 참조
- 구성 요소 개발을 위한 최선의 방법 179
- 구성 요소 관리자
  - 매개 변수 설정 52
  - 바인딩 탭 29
- 구성 요소 내보내기 175
- 구성 요소 마법사 153
- 구성 요소 매개 변수
  - 관리 가능 146
  - 보기 52
  - 설정 52
  - 정보 52
  - 정의 154
  - 개별 구성 요소 이름 참조
- 구성 요소 미리 로드 61
- 구성 요소 미리 보기 60
- 구성 요소 삭제 55
- 구성 요소 설치 12

구성 요소 스키닝 93  
구성 요소 클래스 파일. 참고 사항 클래스 파일  
구성 요소 패널 48  
구성 요소의 시스템 요구 사항 8

## 다

데이터 격자. DataGrid 구성 요소 참조  
데이터 바인딩, XML 파일(자습 과정) 28  
데이터 유형, 인스턴스의 설정(자습 과정) 26  
디스페처(이벤트 브로드캐스터) 66

## 라

라이브러리  
라이브러리 패널 52  
컴파일된 클립 52  
StandardComponents 135  
로드, 구성 요소 62  
리소스, 추가 Macromedia 9  
리스너  
객체 67  
구성 요소 인스턴스와 함께 사용(자습 과정) 35  
구성 요소와 함께 사용(자습 과정) 31  
범위 72  
정보 66  
함수 70

## 마

마법사 153  
매개 변수, 구성 요소 매개 변수 참조  
메타데이터  
정보 144  
태그, 목록 145  
Collection 태그 151  
ComponentTask 태그 153  
Event 태그 148  
Inspectable 태그 146  
무비 클립  
구성 요소로 지정 137  
만들기 134

## 바

바인딩 탭, 샘플 응용 프로그램(자습 과정) 29  
버전 1 구성 요소 업그레이드 63  
버전 2 구성 요소  
상속 18  
장점 16

Flash Player 17  
변수, 정의 143  
부모 클래스, 선택 131  
브로드캐스터 66  
비트맵 캐싱은 지원되지 않음 19

## 사

사용자 정의  
텍스트 80  
상속, 버전 2 구성 요소 18  
색상  
사용자 정의 80  
스타일 속성 설정 89  
색상 및 텍스트 사용자 정의, 스타일 시트 사용 80  
설명 텍스트, 추가 179  
설명서  
개요 8  
용어 안내 9  
설명서 용어 9  
속성 관리자 52  
수퍼 클래스 키워드 142  
스크린  
구성 요소 사용 58  
등록 포인트 56  
인스턴스 이름 56  
ActionScript 사용 56  
ActionScript와 상호 작용 57  
스크린 판독기, 액세스 가능성 20  
스킨  
구성 요소에 적용 99  
링크 식별자 93  
변수 생성 162  
편집 96  
하위 구성 요소에 적용 100  
개별 구성 요소 이름 참조  
스킨 속성  
설정 93  
프로토타입에서 변경 103  
스킨의 링크 식별자 93  
스타일  
구성 요소 만들기 163  
사용자 정의 설정 85  
사용(자습 과정) 27  
설정 80  
우선 순위 결정 89  
인스턴스에 설정 82  
전역 설정 84  
정보 80  
개별 구성 요소 이름 참조  
스타일 선언

- 기본 클래스 87
- 사용자 정의 85
- 전역 84
- 클래스 설정 87
- 스타일 속성, 색상 89
- 스타일 시트
  - 사용자 정의 80
  - 클래스 80
- 실시간 미리 보기 기능 60

## 아

- 아이콘, 구성 요소 178
- 액세스 가능성 20
- 열
  - 추가(자습 과정) 30
- 웹 서비스, 연결(자습 과정) 27
- 이 설명서의 대상 8
- 이벤트
  - 메타데이터 148
  - 범위 위임 74
  - 이벤트 객체 77
  - 일반 161
  - 전달 160
  - 정보 65
  - 핸들러 함수 65
  - 개별 구성 요소 이름 참조
- 이벤트 리스너, 리스너 참조
- 이벤트 전달 160
- 인쇄 규칙 9
- 인스턴스
  - 스타일 선언 80
  - 스타일 설정 82

## 자

- 전역 스타일 선언 84

## 카

- 컬렉션 속성
  - 구성 요소 내보내기 188
  - 사용 189
  - 예제 183
  - 정의 182
  - 클래스 정의 185
  - 프로그래밍 방식으로 액세스 185
- 컬렉션 항목 185
- 컴파일된 클립
  - 라이브러리 패널 52

- 정보 19
- 코드 힌트, 트리거 55
- 클래스
  - 가져오기 141
  - 구성 요소 상속 18
  - 만들기, 참고 사항 구성 요소 만들기
  - 부모 클래스 선택 131
  - 정의 142
  - 참조 만들기(자습 과정) 25
  - 확장 133
  - UIComponent 133
  - UIObject 132
- 클래스 스타일 시트 80
- 클래스 키워드 142
- 클래스 파일
  - 예제 139, 183
  - 정보 140
- 클래스 확장 133

## 타

- 태그, 참고 사항 메타데이터
- 테마
  - 만들기 108
  - 적용 110
  - 정보 104
- 텍스트, 사용자 정의 80

## 파

- 패키지 18
- 프로토타입 103

## 하

- 하위 구성 요소, 스킨 적용 100
- 하위 클래스, 스킨을 바꾸는 데 사용 103
- 핸들러 함수 65

## A

- ActionScript 클래스 파일 138

## C

- className 변수 142
- Collection 태그 151
- components
  - creating subobjects 156

ComponentTask 태그  
  JavaScript(JSFL) 153  
createClassObject() method 156  
CSSStyleDeclaration 84, 85

## D

DataGrid 구성 요소  
  열 추가(자습 과정) 30  
  DataSet에 바인딩(자습 과정) 28  
DataSet 구성 요소, XMLConnector 및 DataGrid에 바인  
  딩(자습 과정) 28  
defaultPushButton 속성 59  
Delegate 클래스(자습 과정) 74  
DepthManager 클래스, 개요 59  
Dial 구성 요소 123, 166  
draw() 메서드, 정의 159

## E

Event 메타데이터 태그 148

## F

Flash 파일, ActionScript 파일 및 SWC 파일 아이콘 122  
Flash JavaScript(JSFL), ComponentTask 태그 153  
Flash MX 버전 및 사용 가능한 구성 요소 12  
Flash Player  
  구성 요소 17  
  지원 63  
FocusManager 클래스, 포커스 탐색 기능 만들기 58

## G

getter/setter 메서드 143

## H

Halo 테마 104  
handleEvent 콜백 함수 70

## I

import 문 141  
init() 메서드, 정의 156  
invalidate() 메서드 160

## L

Label 구성 요소  
  자습 과정 38

## M

MovieClip 클래스, 확장 134

## O

on() 이벤트 핸들러 78

## S

Sample 테마 104  
ScrollPane 구성 요소  
  자습 과정 38  
size() 메서드, 정의 159  
StandardComponents 라이브러리 135  
subobjects, creating 156  
SWC 파일  
  가져오기 178  
  내보내기 176  
  정보 19  
  컬렉션 속성 내보내기 188  
  컴파일된 클립 19  
  테스트 177  
  파일 포맷 175  
SWC 파일 테스트 177  
symbolName 변수 142  
symbolOwner 변수 142

## T

Tab 키 누르기 58  
TextInput 구성 요소(자습 과정) 38

## U

UIComponent 클래스  
  개요 133  
  구성 요소 상속 18  
UIObject 클래스  
  정보 132

## W

WebService 클래스(자습 과정) 27

## X

XMLConnector 구성 요소

스키마 지정(자습 과정) 28

외부 XML 파일 로드(자습 과정) 30

DataSet 구성 요소에 바인딩(자습 과정) 28

## 숫자

9-슬라이스는 지원되지 않음 19

