2부

액션스크립트 응용

2부에서는 저작 환경과 디버거를 사용하는 방법 등 플래시 프로그래밍 실전에서 필요한 내용을 배울 것이다. 또한 플래시 폼을 만드는 법, 온스크린 텍스트 필드를 만드는 법과 같은 플래시 프로그래밍의 두 가지 특별한 부분에 대해서도 알아보자.

- 16장. 액션스크립트 저작 환경
- 17장. 플래시 폼
- 18장. 온스크린 텍스트 필드
- 19장. 디버깅

16

액션스크립트 저작 환경

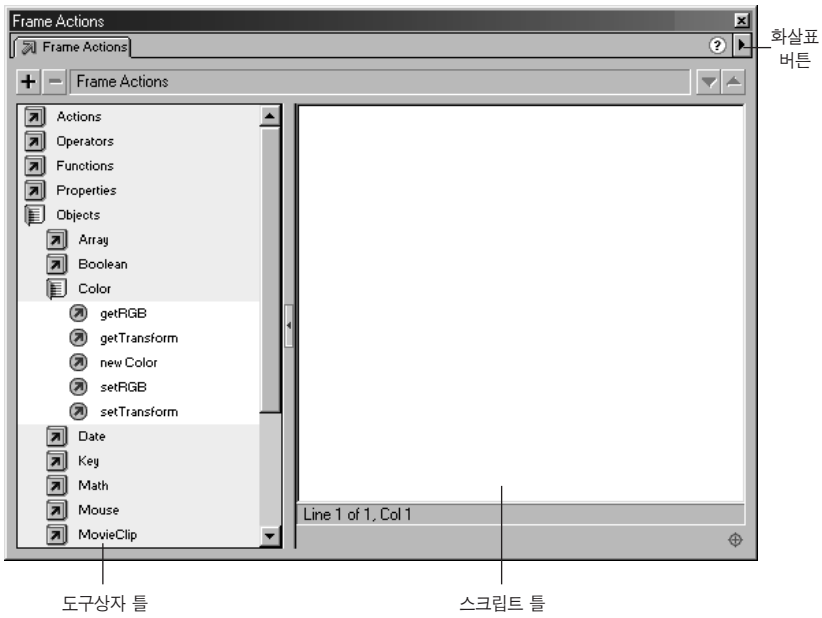
이 장에서는 액션스크립트 코드를 만드는 방법을 자세히 알아보자. 여기서 다룰 주제는 다음과 같다.

- 액션 패널을 이용하여 버튼, 무비 클립, 프레임에 코드를 추가하는 법
- 외부 파일로부터 코드를 불러오는 법
- 코드를 패키징하여 재사용할 수 있는 저작 컴포넌트로 만드는 법

액션 패널

'액션 패널(Actions panel)'은 플래시의 액션스크립트 편집 환경이다. 무비에 있는 각 스크립트는 액션 패널에서 만든다. 액션 패널을 열 때는 Windows → Actions를 선택하면 된다.

액션 패널은 〔그림 16-1〕에 나온 것처럼 '도구상자 틀(Toolbox pane)'과 '스크립트 틀(Script pane)'의 두 부분으로 나눌 수 있다.



〔그림 16-1〕 액션 패널

스크립트 틀에는 현재 선택한 프레임, 버튼 또는 무비 클립에 들어가는 코드가 표시된다. 도구상자는 간단한 액션을 찾아볼 수 있는 퀵 레퍼런스 가이드 역할도 하며 스크립트 틀에 코드를 추가하는 도구로도 쓰인다. 도구상자에 있는 아이템을 더블클릭하면 그 아이템이 스크립트 틀에 추가된다. 또한 도구상자 틀에서 스크립트 틀로 아이템을 끌어다 놓을 수도 있다.

액션 패널의 타이틀을 보면 지금 작업 중인 코드가 프레임에 속하는지(Frame Actions) 아니면 버튼이나 무비 클립에 속하는지(Object Actions) 알 수 있다. 프레임을 선택하면 액션 패널의 타이틀이 자동으로 Frame Actions로 바뀌고, 무비 클립이나 버튼을 선택하면 액션 패널의 타이틀이 Object Actions로 바뀐다.

도구상자에서 아이템을 열거하는 방법은 필자가 이 책에서 액션스크립트의 기능을 설명하면서 사용했던 구분법과는 약간 다르다. 가장 큰 차이를 보이는 것으로 도구상자에서는 statements(선언문)가 별도의 폴더에 들어가지 않는다는 것과 클래스와 객체가 Objects 폴더에 하나의 그룹으로 들어 있다는 점을 들 수 있다. 이 책에

서는 제대로 된 프로그래밍 용어를 사용하기 위해 선언문, 클래스, 객체를 분명하게 구분하여 사용한다.

편집 모드

액션 패널에는 '일반 모드(Normal Mode)'와 '전문가 모드(Expert Mode)'의 두 가지 작업 모드가 있다. 각 모드에 따라 스크립트 틀에 코드를 추가하는 방법이 다르다.

일반 모드

일반 모드에서는 도구상자에서 새로운 선언문을 만들기 위한 용도로 스크립트 틀을 사용한다. 일반 모드에서 스크립트 틀에 새로운 선언문을 추가할 때는 원하는 액션을 더블클릭하거나 그 액션을 도구상자에서 스크립트 틀로 끌어다 놓으면 된다. 사용자가 직접 코드를 만들어 선언문을 추가할 때는 evaluate 액션을 선택하고, 액션 패널 아래쪽에 있는 매개변수 틀(Parameters pane)의 Expression 필드(그림에 나오지 않았음)에 선언문을 입력하면 된다.

스크립트 틀에 선언문을 추가하면 액션 패널의 아래쪽에 있는 매개변수 틀을 통해 선언문을 수정할 수 있다. 매개변수 틀의 형태나 내용은 스크립트 틀에서 선택한 선언문의 종류에 따라 달라진다. 액션 패널을 일반 모드 상태로 사용하면 스크립트 틀에 코드를 직접 입력할 수 없다. 일반 모드에서는 스크립트 틀에서 코드를 직접 편집할 수 없고 단지 선언문의 목록을 보여주는 역할만 할 뿐이다. 스크립트 틀에 있는 선언문을 고치고 싶다면 매개변수 틀을 이용해야 한다.

플래시 5의 일반 모드는 플래시 4의 액션 패널과 겉모양은 비슷하지만, 일반 모드에서 플래시 4와 호환되는 액션스크립트만을 만들 수 있는 것은 아니다. 일반 모드와 전문가 모드 중 어떤 편집 모드를 사용하더라도 플래시 5 전용 코드를 만들 수 있다('부록 C. 하위 호환성' 참조). 일반 모드는 초보 프로그래머가 사용하기는 편하겠지만 프로그램이 조금이라도 복잡해지면, 여러 가지 제약조건 때문에 오히려 더 사용하기 불편해진다. 따라서 이 책에서는 전문가 모드만을 사용한다. 하지만 여러 줄짜리 주석은 전문가 모드에서만 사용할 수 있다는 점을 제외하면 일반 모드에서

만든 코드를 전문가 모드에서도 만들 수 있고 거꾸로 전문가 모드에서 만든 코드를 일반 모드에서도 만들 수 있다.

전문가 모드

전문가 모드에서는 스크립트 틀이 일반적인 텍스트 편집기 역할을 한다. 전문가 모드에서 코드를 만들거나 고칠 때는 스크립트 틀에 직접 내용을 입력할 수 있다. 물론 도구상자의 아이템을 더블클릭하여 스크립트 틀에 추가하는 방법도 여전히 사용할 수 있다. 전문가 모드에서는 매개변수 틀을 쓰지 않는다. 그 대신 스크립트 틀에 매개변수를 바로 입력할 수 있다.

편집 모드 설정법

액션 패널의 편집 모드는 프레임이나 객체별로 따로 설정할 수 있다. 플래시에서는 무비에 있는 각 객체와 프레임에 그에 해당하는 액션 패널 편집 모드를 저장해 두기 때문이다. 예를 들어 2번 프레임에서는 일반 모드를 선택하고 3번 프레임에서는 전문가 모드를 선택하면, 나중에 2번 프레임에 있는 액션스크립트를 열었을 때 자동으로 일반 모드로 전환되고 3번 프레임에 있는 코드를 편집할 때는 자동으로 전문가 모드로 전환된다.

액션 패널의 오른쪽 위에 있는 화살표 버튼([그림 16-1] 참조)을 클릭하면 전문가 모드나 일반 모드 중 하나를 선택할 때 각 프레임이나 객체의 액션 패널 모드를 설정할 수 있다. 무비에 있는 모든 프레임과 객체에 적용되는 액션 패널 모드의 기본값을 설정하고 싶다면, Edit → Preferences → General → Actions Panel → Mode에서 전문가 모드나 일반 모드 가운데 하나를 선택하면 된다. 기본 모드를 새로 설정하면 그 기본 모드는 편집 모드를 별도로 선택하지 않은 프레임이나 객체에만 적용된다. 따라서 기본 모드는 무비를 만들기 시작할 때 처음부터 설정해 두는 것이 좋다. 일단 프레임이나 객체에서 스크립트를 만들고 나면 원래 있던 모드를 무시하고 새로운 모드를 일괄적으로 적용할 수 없다.

전문가 모드에서 일반 모드로 전환하면 모든 소스 코드 형식이 바뀐다. 코드의 모양이 플래시 표준 형식에 따라 다시 구성되므로 필요 없는 공백문자는 삭제되고 모든 주석은 한 줄짜리 주석으로 바뀌며, 선언문 블록의 들여쓰기는 이 책에서 사용하는 형식과 같은 형식으로 통일된다.

프레임에 스크립트 추가하기

플래시 문서는 기본적으로 애니메이션 개념을 기준으로 구성되어 있다. 모든 플래시 문서는 일련의 프레임이나 비주얼, 오디오 컨텐츠로 이루어진다. 거의 모든 액션스크립트 코드가 이러한 프레임 기반 구조에 연결되어 있다. 어떤 프레임에 코드 블록을 집어넣으면 그 블록이 실행되는 타이밍은 무비가 재생되는 순서에 의해 결정된다. 프레임에 코드를 추가할 때는 코드에서 해야 할 일 외에도 그 코드를 실행할 시기에 대해서도 생각해 보아야 한다.

어떤 무비의 20번째 프레임이 화면에 나타날 때 새로운 무비 클립을 스테이지에 추가하는 코드를 만든다고 해보자.

```
_root.attachMovie("myClip", "clip1", 0);
```

20번째 프레임이 화면에 표시될 때 이 코드가 실행되도록 하려면 위 코드를 무비의 20번 프레임에 있는 키프레임에 넣어야 한다. 코드를 키프레임에 넣을 때는 타임라인에서 키프레임을 선택하고 액션 패널을 연 다음 원하는 코드를 스크립트 틀에 추가하면 된다. 무비를 재생하면 프레임의 내용이 화면에 표시되기 전에 키프레임에 있는 코드가 실행된다.

키프레임에 있는 코드는 무비의 재생 과정과 동기화된(즉 무비 재생 순서와 같은 순서로 실행되는) 작업을 처리하는 데 쓰이며, 무비나 무비 클립 전체에 걸쳐서 사용할 변수, 함수, 객체와 같은 프로그램 요소를 만드는 데도 쓰인다. 동기화된 작업의 대표적인 예로 타임라인 루프를 생각할 수 있다. 다음과 같은 코드를 무비의 15번 프레임에 추가한다고 가정해 보자.

```
gotoAndPlay(10);
```

무비의 플레이헤드가 15번 프레임에 도착하면 이 코드가 실행되며 무비가 10번 프레임부터 다시 재생된다. 그 다음에 15번 프레임으로 돌아오면 또 다시 10번 프레임부터 재생이 시작된다. 즉 무비가 10번 프레임과 15번 프레임 사이를 계속 맴돌게 된다.

하지만 키프레임에 있는 코드에서 언제나 무비의 플레이헤드를 제어하거나 동기화시키는 작업만 처리하는 것은 아니다. 함수나 변수, 객체 및 다른 프로그램 요소를 저장하기 위한 용도로 키프레임을 사용할 수도 있다. 예를 들어 다음과 같은 moveTo()

함수를 무비 클립의 1번 프레임에 추가하면 나중에 버튼에서도 이 함수를 호출할 수 있다.

```
function moveTo (x, y) {
  _x = x;
  _y = y;
}
```

주의 사항: 프레임에 있는 코드는 무비 재생 순서에 따라 실행되므로, 코드를 실행할 때 필요한 모든 변수, 함수 및 기타 프로그램 요소가 준비되어 있는지 확인해야 한다. 예를 들면 어떤 함수가 10번 프레임에서 정의된다면 3번 프레임에서는 그 함수를 사용할 수 없다. 따라서 무비 전반에 걸쳐서 쓰이는 코드는 메인 타임라인의 첫 번째 프레임에 넣어두어야 한다.

또한 무비의 일부가 아직 완전히 로딩되지 않은 상태에서 그 부분에 있는 내용을 실행하지 않도록 주의해야 한다. 무비의 특정 부분이 로딩되었는지 확인하고 싶다면 무비 클립의 _framesloaded 속성이나 getBytesLoaded() 메소드를 이용하면 된다. 샘플 코드를 보고 싶다면 '3부. 레퍼런스'의 Movieclip._framesloaded 부분에 있는 내용을 참조하기 바란다.

버튼에 스크립트 추가하기

사용자 이벤트에 의해 코드를 실행하고 싶다면 버튼에 코드를 추가해야 한다. 예를 들어 어떤 버튼을 클릭할 때 무비의 플레이헤드를 section1이라는 레이블이 붙은 프레임으로 움직이고 싶다면 그 버튼에 다음과 같은 코드를 추가하면 된다.

```
on (release) {
  gotoAndStop("section1");
}
```

어떤 버튼에 코드를 추가하려면 스테이지에서 버튼을 선택하고 액션 패널의 스크립트 틀에 코드를 추가하면 된다. 버튼의 코드는 반드시 그 코드가 실행되는 조건에 해당하는 이벤트 핸들러에 들어가야 한다. 대부분의 버튼 액션을 유발하는 이벤트는 release 이벤트이다. rollOver 이벤트를 이용하면 마우스를 클릭할 때가 아니라 마우스가 버튼 위를 지나갈 때 원하는 코드를 실행시킬 수 있다.

```
on (rollOver) {
  gotoAndStop("section1");
}
```

버튼 이벤트 핸들러에 대한 자세한 내용은 '10장. 이벤트 및 이벤트 핸들러'를 참조하기 바란다.

한 버튼에 수천 줄의 코드를 입력해도 문법적으로는 문제가 없지만 버튼에 그렇게 긴 코드를 집어넣는 것은 권장할만한 방법은 아니다. 될 수 있다면 버튼 코드의 기능을 일반화하여 하나의 함수로 만들고 그 함수를 타임라인에 집어넣는 것이 좋다. 다음과 같은 코드를 버튼에 직접 추가할 수도 있다.

```
on (release) {
  title._xscale = 20;
  title._yscale = 20;
  title._alpha  = 50;
  title._gotoAndPlay("fadeout");
}
```

하지만 위와 같은 코드를 함수에 집어넣고 버튼에서는 그 함수만 호출하도록 하는 것이 훨씬 더 좋은 방법이다.

```
// 버튼의 타임라인에 있는 1번 프레임에 들어갈 코드
function transformClip(clip, scale, transparency, framelabel) {
  clip._xscale = scale;
  clip._yscale = scale;
  clip._alpha = transparency;
  clip.gotoAndPlay(framelabel);
}
// 버튼에 들어가는 코드
on (release) {
  transformClip(title, 20, 50, "fadeout");
}
```

이렇게 하면 모든 코드를 한 군데로 모을 수 있으므로 코드를 관리하기도 쉽고 여러 개의 버튼에 비슷한 기능을 추가할 때도 매우 편리하다.

버튼 코드는 스테이지에 있는 버튼 객체에 추가해야 하며 그 코드에는 이벤트 핸들러가 있어야 한다. 버튼 코드에 이벤트 핸들러가 없으면 오류가 생긴다. 또한 버튼 심벌의 내부 프레임인 UP, OVER, DOWN, HIT 프레임에는 코드를 추가할 수 없다.

무비 클립에 스크립트 추가하기

앞에서 무비 클립의 타임라인에 있는 프레임에 코드를 추가하는 방법을 배웠다. 하지만 무비 클립 객체 자체에 코드를 추가할 수도 있다. 이 때는 스테이지에서 무비 클립 인스턴스를 선택하고 액션 패널의 스크립트 틀에 코드를 입력하면 된다. 버튼의 경우와 마찬가지로 무비 클립 객체에 추가되는 모든 코드는 이벤트 핸들러 안에 넣어야 한다. 이벤트 핸들러는 인터프리터에 무비 클립 코드를 실행하는 시기를 알려주는 역할을 한다. 예를 들어 다음과 같은 코드를 사용하면 무비 클립이 로딩되었을 때 변수 x의 값을 10으로 설정할 수 있다.

```
onClipEvent (load) {
  var x = 10;
}
```

무비 클립 이벤트 핸들러는 마우스나 키보드 관련 이벤트 또는 데이터 로딩, 프레임 렌더링, 무비 클립의 생성 및 삭제와 관련된 이벤트를 처리하는 데 쓰인다. 무비 클립과 이벤트 핸들러에 대한 자세한 내용은 '10장. 이벤트 및 이벤트 핸들러'와 '13장. 무비 클립'을 참조하기 바란다.

코드는 어디에?

숙련된 플래시 사용자라도 무비에서 코드가 어디에 들어있는지 찾아내기 힘든 경우가 종종 있다. 코드는 어떤 타임라인의 어떤 프레임에도 추가할 수 있으며, 모든 버튼과 무비 클립에도 추가할 수 있기 때문에 코드를 찾아낸다는 것이 간단한 일은 아니다. 물론 처음부터 구조를 짜임새 있게 구성하고 코드에 자세한 주석을 추가한다면, 주어진 프로젝트를 처리하는 데 걸리는 시간을 수십 시간에서 수백 시간까지 줄일 수 있다. 하지만 무비에서 필요한 코드를 찾아내기 힘들다면 일단 액션 패널을 열고 다음과 같은 방법을 적용해 보자.

- 타임라인에 조그만 동그라미 아이콘이 있는 프레임을 하나씩 클릭해 본다. 동그라미 아이콘은 그 타임라인에 액션스크립트 코드가 있다는 것을 나타낸다.
- 스테이지에서 검은 테두리로 된 흰색 원을 찾아서 선택한다. 이러한 모양의 원은 비어있는 클립을 나타내므로 그 안에는 코드만 들어있을 가능성이 높

다. 비어있는 클립에 코드가 없다면 더블클릭하여 그 클립을 편집하는 상태로 들어가서 그 안에 있는 프레임을 확인한다.

- 무비에 들어있는 각 버튼을 하나씩 확인해 본다. 코드를 한 군데로 모으지 않고 꽤 덩치가 크고 중요한 코드를 버튼에 직접 집어넣는 프로그래머도 있다.
- 타임라인에 숨겨진 레이어나 다른 레이어에 덮여 있는 레이어가 없는지 확인한다. 옆에 빨간 X자 모양의 아이콘이 있는 레이어에는 플래시 저작 환경에서는 숨겨져 있지만 무비를 재생하면 실행되는 코드가 들어간 클립이나 버튼이 들어있을 수도 있다. 마찬가지로 덮여 있는 레이어에도 코드가 들어있는 객체가 숨겨져 있을 수 있다. 덮여 있는 레이어의 내용을 보고 싶다면 잠겨 있는 레이어를 해제하면 된다.
- 잠겨 있는 모든 레이어를 해제한다. 레이어가 잠겨 있으면 그 안에 들어있는 비어있는 무비 클립(검은색 테두리로 된 흰색 원이 있는 클립)이 보이지 않는다.

위와 같은 방법을 모두 적용했는데도 원하는 코드를 찾지 못할 수도 있다. 그 무비를 만든 사람이 어떤 코드를 숨기기로 마음먹었다면 플래시에는 코드를 숨길만한 장소가 무궁무진하다. 예를 들어 비어있는 클립을 스테이지의 외곽에서 아주 멀리 떨어진 곳에 숨겨 놓는다면 그 클립을 찾는 것이 거의 불가능하다. 하지만 그렇다고 해서 코드를 찾을 방법이 전혀 없는 것은 아니다. Movie Explorer를 이용하면 무비에 있는 어떤 스크립트라도 찾아낼 수 있다. Window → Movie Explorer를 선택하면 프레임, 버튼, 무비 클립에 들어있는 스크립트를 포함한 무비에 있는 모든 요소를 한 눈에 볼 수 있다. 스크립트는 액션 패널의 도구상자에 있는 아이템 아이콘과 같은 파란색 화살표로 된 액션 아이콘으로 표시된다. 또한 Movie Explorer에서 필터를 적용하여 스크립트만 화면에 나타나게 할 수도 있다. 탐색기 패널의 맨 위에 있는 Show 메뉴에서 Actions 아이콘만 선택하고 나머지는 모두 해제하면 스크립트만 골라서 찾아볼 수 있다.

생산성

액션스크립트 소스 코드를 만드는 작업을 효율적으로 처리하고 싶다면 다음과 같은 팁을 활용해 보자.

- 모든 타임라인 스크립트를 scripts라는 별도의 레이어에 넣어둔다. scripts 레이어에는 다른 내용은 넣지 말고 코드만 집어넣는 것이 좋다. 이 레이어는 레이어 구조에서 맨 위에 (또는 Load Order를 Top Down으로 설정했다면 맨 아래) 놓아서 다른 모든 레이어가 로딩된 후에 scripts 레이어에 있는 코드가 실행되도록 하자. scripts 레이어를 언제나 같은 위치에 놓는 습관이 익숙해지면 코드를 찾기도 쉬워진다(앞 절에서 코드를 빨리 찾는 것이 상당히 중요하다는 것을 느꼈을 것이다).

- 모든 프레임 레이블은 labels라는 별도의 레이어에 모아둔다. 대신 labels 레이어에는 다른 내용은 집어넣지 말고 프레임 레이블을 모아두기 위한 레이어로 활용하자.

- 액션 패널의 오른쪽 위에 있는 화살표 버튼의 메뉴를 활용하자. 그 메뉴에는 찾기 및 바꾸기, 소스 코드 출력, 스크립트 틀 폰트 조절 등의 기능이 들어 있다.

- 여러 개의 프로젝트에서 공유하는 코드 라이브러리를 이용할 때는 코드를 외부 파일에 저장한다. 자세한 내용은 다음 절인 '액션스크립트 코드 외부화'에서 알아보기로 하자.

- 불필요한 키보드 입력을 줄이기 위해 액션에서 많이 쓰이는 단축키(예를 들면 gotoAndPlay()는 Esc-G-P 키로 입력할 수 있다)를 활용하자. 단축키 목록은 액션 패널의 + 버튼을 누르면 찾을 수 있다.

- 액션 패널을 잘 설정하여 Instance, Frame, Text Options와 같이 코드와 관련된 패널과 함께 놓도록 하자. Window → Panels를 선택한 후 원하는 패널을 열고 액션 패널을 그 옆으로 끌어다 놓으면 저절로 착 달라붙는다.

- 도구상자의 크기를 조절해 보자. 도구상자와 스크립트 틀 사이의 경계선을 마우스로 드래그하여 도구상자 크기를 줄이거나 아예 숨겨버리면 코딩하는 데 필요한 공간을 더 확보할 수 있다.

액션스크립트 코드 외부화

액션스크립트 코드는 외부 텍스트 파일(보통 .as라는 확장자를 사용한다)이나 .swf 파일로 저장할 수 있으며, 그 파일은 다시 플래시 문서에서 불러올 수 있다. 코드를 외부 파일로 저장해 두면 여러 프로젝트에서 하나의 표준 코드 라이브러리를 활용할 수 있다는 장점이 있다. 외부 코드를 플래시로 불러올 때는 Import From File을 선택하거나 #include를 사용하거나 공유 라이브러리를 이용하면 된다.

Import From File(무비를 만들 때 가져오는 법)

.fla 파일을 편집하는 과정에서 Import From File을 선택하면 액션 패널의 스크립트 틀로 코드를 불러올 수 있다. Import From File은 액션 패널의 오른쪽 위에 있는 화살표 버튼(〔그림 16-1〕 참조)에 있는 메뉴에서 찾을 수 있다. Import From File은 외부 파일의 내용을 액션 패널로 복사해오는 작업이며, 이 때 원래 액션 패널에 들어있던 내용을 모두 지워버린다. Import From File을 이용하여 불러온 코드는 바로 .fla 파일에 연결되지는 않는다. 원래 있던 스크립트를 지우지 않고 외부 파일의 스크립트 텍스트를 추가하고 싶다면, 외부 텍스트 에디터에서 원하는 파일을 불러와서 직접 복사하여 붙여야 한다.

#include(컴파일할 때 불러오기)

#include 명령어를 이용하면 .fla 파일로부터 .swf 파일을 만들 때(컴파일할 때) 외부 텍스트 파일로부터 코드를 가져올 수 있다. #include문의 사용법은 3부를 참조하기 바란다.

공유 라이브러리(실행 중에 불러오기)

무비가 실제로 재생되는 동안에 외부 소스로부터 코드를 가져오고 싶다면 가져올 코드가 들어있는 무비 클립으로부터 공유 라이브러리 .swf 파일을 만들어야 한다. 실행 중에 외부 파일을 불러오는 방법은 여러 개의 무비에서 코드를 공유하는 방법 가운데 가장 강력한 유연성을 자랑하는 방법이다. 공유 코드를 변경하더라도

그 코드를 사용하는 다른 무비를 다시 컴파일하지 않아도 되기 때문이다. 공유 라이브러리 .swf 파일이 업데이트되면 그 파일에 연결된 무비에도 자동으로 업데이트 내용이 반영된다.

아래에 열거한 내용은 codeLibrary.swf라는 무비의 간단한 테스트용 함수를 myMovie.swf라는 무비에서 공유하는 방법을 적어놓은 것이다. 우선 codeLibrary .swf 무비를 만들자.

1. 새로운 플래시 문서를 만든다.
2. sharedFunctions라는 새로운 무비 클립 심벌을 만든다.
3. sharedFunctions 클립의 1번 프레임에 다음과 같은 코드를 추가한다.

```
function test () {
trace("The shared function, test, was called.");
}
```

4. 라이브러리에서 sharedFunctions 클립을 선택한다.
5. 라이브러리 패널에서 Options → Linkage를 선택한다.
6. Export This Symbol을 선택한다.
7. Identifier 상자에 sharedFunctions라고 입력한다.
8. 이 문서를 codeLibrary.fla라는 이름으로 저장한다.
9. File → Export Movie를 선택하여 codeLibrary.fla로부터 codeLibrary.swf를 만든다.
10. codeLibrary.swf와 codeLibrary.fla 파일을 닫는다.

이제 codeLibrary.swf에서 가져온 코드를 실행시키는 myMovie.swf 파일을 만들자.

1. 새로운 플래시 문서를 만든다.
2. 새 문서를 codeLibrary.fla와 같은 디렉토리에 myMovie.fla라는 이름으로 저장한다.

3. Layer 1의 이름을 sharedCode로 바꾼다.
4. File → Open As Shared Library를 선택한 후 codeLibrary.fla를 선택한다. codeLibrary.fla의 라이브러리가 나타난다.
5. codeLibrary.fla 라이브러리에서 sharedFunctions 클립의 인스턴스를 myMovie.fla의 1번 스테이지의 프레임에 마우스로 끌어다 놓는다.
6. 스테이지에 있는 인스턴스를 선택하고 Modify → Instance를 선택한다.
7. 인스턴스의 이름을 sharedFunctions로 설정한다.
8. myMovie.fla의 메인 타임라인에 scripts라는 새로운 레이어를 만든다.
9. scripts 레이어의 2번 프레임에 키프레임을 추가한다.
10. sharedCode 레이어에 새로운 프레임을 추가한다.
11. scripts 레이어의 2번 프레임에 다음과 같은 코드를 추가한다.

```
stop();
sharedFunctions.test();
```

12. myMovie.swf로 내보낸다(File → Export Movie).
13. 이 무비를 실행시키면 "The shared function, test, was called."라는 문장이 Output 창에 출력된다.

공유 라이브러리는 .gif 파일이 .html 파일에 연결된 것과 같은 식으로 무비에 연결되어 있다. 따라서 공유 라이브러리의 .swf 파일은 그 라이브러리를 사용하는 파일들과 함께 업로드해야 한다. 외부에서 가져온 심벌에서 공유 라이브러리로 연결되는 링크의 위치를 바꾸고 싶다면 다음과 같이 하면 된다.

1. Library에서 심벌을 선택한다.
2. 라이브러리 패널에서 Options → Linkage를 선택한다.
3. Import This Symbol from URL에 새로운 위치를 적는다.

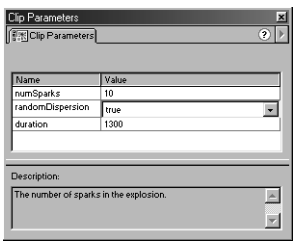
컴포넌트를 스마트 클립으로 만드는 법

스마트 클립은 플래시 저작 도구에서 특별한 그래픽 사용자 인터페이스를 통해 변수 값을 대입할 수 있는 무비 클립이다. 스마트 클립을 이용하면 프로그래머가 아닌 사람도 프로그램을 통해 제어하는 무비 클립을 자신이 원하는 형태로 만들 수 있다. 스마트 클립에서는 그 특징을 결정하는 변수를 클립의 실제 코드와 분리하여 '블랙 박스' 처럼 사용할 수 있도록 되어있다. 즉 어떤 내용을 입력하면 그 내용에 따라 일정한 결과가 나온다면 그 블랙 박스 안에서 행해지는 실제 작업은 자세히 알지 못해도 상관없다.

일반적으로 변수는 무비 클립의 소스 코드에서 초기화한다. 아래의 예에서는 불꽃놀이 효과를 제어하는 데 쓰이는 변수를 초기화한다.

```
// 사용자 정의 변수
var numSparks = 10;                 // 폭발시에 필요한 스파크 클립의 개수
var randomDispersion = true;        // 폭발 형태(true - 무작위,
                                    // false - 일정한 형태)
var duration = 1300;                // 폭발 시간(밀리초 단위)
```

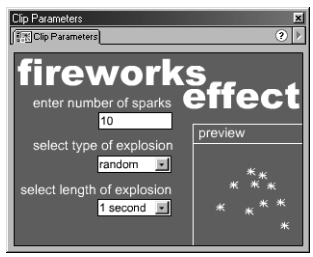
하지만 프로그래밍을 잘 모르는 사람들에게는 이러한 간단한 소스 코드를 수정하는 것도 상당히 부담스러운 일이다. 하지만 시스템을 스마트 클립으로 만들면 프로그래밍을 모르는 사람이라도 간편한 애플리케이션 형태의 인터페이스를 통해 불꽃 효과를 설정할 수 있다. 변수 초기화 코드 대신 사용할 수 있는 스마트 클립 인터페이스의 모양은 [그림 16-2]와 같다.



[그림 16-2] 스마트 클립 - 설정 인터페이스

스마트 클립 인터페이스에서는 각 변수의 이름과 값이 각각 화면에 나타난다. 변수 이름은 편집할 수 없기 때문에 실수로 변수 이름을 건드려서 시스템이 엉망이 되는 일은 없다. 또한 각 변수에 그 변수의 역할과 그 변수를 설정하는 법에 대한 자세한 설명을 집어넣을 수도 있다. 사용할 수 있는 값이 몇 가지로 제한된 변수(앞에 있는 randomDispersion같은 변수) 값을 대입할 때는 드롭다운 메뉴를 이용할 수도 있다.

프로그래밍을 모르는 사람에게는 소스 코드보다는 〔그림 16-2〕에 나온 것과 같은 인터페이스를 사용하는 것이 훨씬 편하다. 더 예쁜 스마트 클립을 만들 수도 있다. 기본 스마트 클립 인터페이스 대신 〔그림 16-3〕에 나온 것처럼 사용자가 직접 만든 인터페이스를 사용할 수 있기 때문이다. 사용자 정의 스마트 클립 인터페이스를 이용하면 시스템의 변수를 완전히 감출 수 있으므로 프로그래밍을 모르는 사용자라도 불꽃놀이 효과의 각 인스턴스를 텍스트 필드와 풀다운 메뉴만으로 마음대로 설정할 수 있다. 또한 이 인터페이스에서는 사용자가 설정한 효과를 미리 보여주는 프리뷰(preview) 기능도 제공한다.



〔그림 16-3〕 사용자 정의 스마트 클립 - 설정 인터페이스

이제 스마트 클립을 어떻게 만드는지 알아보자.

표준 인터페이스로 스마트 클립 만들기

방금 배웠듯이 스마트 클립에는 기본 시스템 인터페이스 또는 사용자 정의 인터페이스를 적용할 수 있다. 우선 표준 인터페이스를 만드는 법을 알아보자.

스마트 클립을 만드는 첫 번째 단계는 하나 이상의 변수를 통해 제어할 수 있는 일반적인 무비 클립을 만드는 것이다. 예를 들어 다음 코드에서는 xPos와 yPos라는 변수를 통해 스테이지상에서 클립의 위치를 결정한다.

```
_x = xPos;
_y = yPos;
```

무비 클립을 본인 또는 다른 사람이 사용할 수 있도록 스마트 클립으로 만들 때는 클립이 스테이지에 올라갔을 때 특정한 변수를 스마트 클립 인터페이스를 통해 설정할 수 있도록 만들어야 한다. 이렇게 스마트 클립 인터페이스를 통해 설정하는 변수를 '클립 매개변수(clip parameter)'라고 부른다. 하나 이상의 클립 매개변수를 통해 기능을 바꿀 수 있는 클립을 만들고 나면 그 매개변수를 설정하는 데 사용할 스마트 클립 인터페이스를 만들어야 한다.

스마트 클립에 표준 인터페이스 추가하기

무비에 기본 스마트 클립 인터페이스를 추가할 때는 다음과 같이 하면 된다.

1. 라이브러리에서 클립을 선택한다.
2. Options → Define Clip Parameters를 선택한다(그러면 Define Clip Parameters 대화상자가 나타난다).
3. Parameters 틀에서 + 버튼을 클릭하여 클립 매개변수를 추가한다.
4. 3번 단계를 반복하여 모든 클립 매개변수를 추가한다.
5. 다음 절 내용을 참고하여 매개변수를 설정한다.

표준 클립 매개변수 설정

스마트 클립에 클립 매개변수를 추가할 때는 우선 매개변수의 이름을 설정하고 필요하면 기본값을 입력하면 된다. 변수와 마찬가지로 클립 매개변수에는 여러 유형의 데이터를 넣을 수 있다. 하지만 클립 매개변수로 사용할 수 있는 데이터형은 변수에서 사용할 수 있는 데이터형과는 조금 다르다. 클립 매개변수에는 문자열, 숫자, 배열, 객체, 리스트만을 사용할 수 있다. 이러한 데이터형은 두 가지 면에서 변수에서 사용할 수 있는 데이터형과 차이점을 보인다.

- 클립 매개변수에서는 인터페이스에서만 사용할 수 있는 데이터형인 '리스트(list)'를 지원한다. 리스트는 매개변수의 값으로 미리 정해진 옵션 중 하나만을 사용할 수 있도록 할 때 쓰인다. 예를 들어 difficulty라는 매개변수에 "hard", "normal", "easy" 가운데 한 가지 값만 사용하도록 할 수도 있다. 리스트를 사용하면 사용자가 클립 매개변수에 원하지 않는 값을 입력하는 것을 방지할 수 있다.

- 원시 데이터형 중에서 부울, null, undefined 값은 클립 매개변수의 값으로 바로 사용할 수 없다. 이는 클립 매개변수 때문이 아니라 스마트 클립 인터페이스의 한계 때문에 생기는 문제이다. 물론 클립 안에 있는 코드에서는 부울, null, undefined 값을 변수에 대입하여 클립 매개변수를 초기화할 수 있다. 클립 매개변수에 true나 false와 같은 부울 값을 사용하는 경우에는 문자열 true나 false 대신 숫자 1 또는 0을 사용한다. 숫자 1과 0은 부울형이 들어갈 자리에 사용하면 각각 true와 false로 변환되기 때문이다.

클립 매개변수에 이름과 기본값을 지정하려면 다음과 같은 단계를 따라가면 된다.

1. 매개변수 이름(Name)을 더블클릭하여 적절한 인식자를 입력한다.
2. 매개변수 유형(Type)을 더블클릭하고 다음 중 하나를 선택한다.
    a. 매개변수가 문자열이나 숫자인 경우에는 Default
    b. 매개변수가 배열인 경우에는 Array
    c. 매개변수가 객체인 경우에는 Object
    d. 매개변수로 사용할 수 있는 숫자나 문자값이 미리 정해져 있는 경우에는 List
3. 기본값이 필요한 경우에는 매개변수 값(Value)을 더블클릭하고 기본값을 입력한다. 이렇게 기본값을 설정해 두면 스마트 클립 인터페이스에서 이 값이 기본값으로 나타난다. 기본값을 입력하는 방법은 다음과 같이 매개변수 유형에 따라 달라진다.
    a. Default 매개변수의 경우에는 매개변수 값을 더블클릭하고 문자열 또는 숫자를 입력한다.

b. Array, Object, List 매개변수의 경우에는 우선 매개변수 값을 더블클릭한다. 그러면 Values 대화상자가 나타나는데, 여기서 +, -, 화살표 버튼을 이용하여 아이템을 추가, 제거하거나 아이템의 순서를 바꿀 수 있다. 설정이 끝나면 OK를 클릭한다.

4. 매개변수에 대한 설명을 집어넣고 싶다면 Description 상자에 내용을 입력한다.

5. 스마트 클립을 사용할 때 매개변수의 이름이 바뀌는 것을 방지하려면 맨 아래에 있는 Lock in Instance를 체크한다.

6. Define Clip Parameters 대화상자에서 OK를 클릭하여 매개변수 설정을 마친다.

표준 클립 매개변수 제거 및 순서 바꾸기

때때로 스마트 클립의 매개변수를 제거하거나 그 순서를 바꿔야 하는 경우도 있다.

클립 매개변수를 제거하려면 다음과 같은 단계를 따라하면 된다.

1. 라이브러리에서 수정할 스마트 클립을 선택한다.
2. Options → Define Clip Parameters를 선택한다.
3. 제거할 매개변수를 선택한다.
4. - 버튼을 클릭한다.
5. OK를 클릭한다.

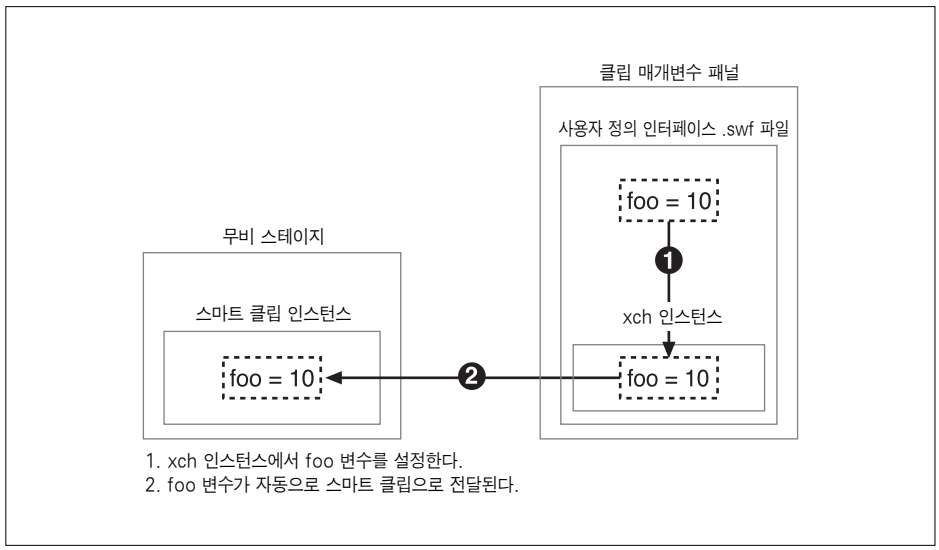
클립 매개변수의 순서를 바꾸려면 다음과 같이 하면 된다.

1. 라이브러리에서 수정할 스마트 클립을 선택한다.
2. Options → Define Clip Parameters를 선택한다.
3. 순서를 바꿀 매개변수를 선택한다.
4. 화살표 버튼을 이용하여 매개변수의 위치를 조절한다.
5. OK를 선택한다.

사용자 정의 인터페이스로 스마트 클립 만들기

사용자 정의 인터페이스를 사용하는 스마트 클립을 만들려면, 우선 앞에서 설명한 방법에 따라 몇 가지 매개변수를 통해 특성이 달라지는 일반 무비 클립을 만들어야 한다. 그리고 나서 클립 매개변수 패널의 인터페이스로 사용할 .swf 파일(인터페이스 .swf라고도 부름)을 따로 만들어야 한다. 보통 사용자가 매개변수 값을 입력할 수 있도록 해주는 그래픽 인터페이스를 갖추고 있는 .swf 파일을 만들면 된다(텍스트 박스, 메뉴, 버튼과 같은 아이템을 활용하면 된다). 사용자가 이 무비에 입력한 값들은 자동으로 스마트 클립의 매개변수로 전달된다.

스마트 클립은 xch(exchange를 줄여쓴 단어) 인스턴스를 통해 인터페이스 .swf 파일과 데이터를 주고받는다. xch 인스턴스는 인터페이스 .swf 파일 안에 들어가는 특별한 인스턴스이다(xch 인스턴스를 만드는 방법은 잠시 후에 배울 것이다). 매개변수 이름과 값이 인터페이스 .swf 파일에서 스마트 클립으로 전달되는 과정은 〔그림 16-4〕와 같다.



〔그림 16-4〕 사용자 정의 스마트 클립에서 데이터가 전달되는 과정

인터페이스 .swf 파일과 스마트 클립 사이의 데이터 전달은 어떤 주기에 따라 이루어진다. 스테이지에서 스마트 클립 인스턴스를 선택하면 그에 해당하는 .swf 파일이 Clip Parameters 패널로 로딩된다. 그러면 스마트 클립 인스턴스의 현재 매개

변수가 .swf 파일의 xch 인스턴스로 전달된다. .swf 파일에서는 그러한 매개변수를 받아서 현재 인터페이스의 상태를 설정한다. 그리고 나서 .swf 파일의 xch에 있는 변수 값이 설정되면, 그 값은 자동으로 스마트 클립의 매개변수로 전달된다. 스마트 클립 인스턴스의 선택을 해제하면 인터페이스 .swf 파일이 Clip Parameters 패널에서 없어진다. 하지만 매개변수 값은 없어지지 않고 스마트 클립에 저장된다. 매번 스마트 클립 인스턴스를 선택할 때마다 그 스마트 클립에서는 .swf 파일의 xch 클립으로 매개변수를 전달한다. 이러한 주기를 돌기 때문에 인터페이스 .swf 파일의 내용은 언제나 스마트 클립 매개변수와 연동된다.

이제 사용자 정의 인터페이스 .swf 파일을 만드는 방법과 그 .swf 파일을 스마트 클립에 연결하는 방법을 알아보자. 온라인 코드 창고의 "Playhead Control" 부분에서 사용자 정의 사용자 인터페이스를 사용하는 스마트 클립의 샘플을 구할 수 있다.

사용자 정의 인터페이스 .swf 파일 만들기

스마트 클립의 사용자 인터페이스로 사용할 .swf 파일을 만들려면 다음과 같은 단계를 거치면 된다.

1. 새로운 플래시 문서를 시작한다.
2. xchLayer라는 이름으로 새로운 레이어를 만든다.
3. Insert → New Symbol을 선택하여 무비 클립 심벌을 만든다.
4. 새로운 심벌의 이름을 xchClip으로 바꾼다.
5. xchLayer 레이어에 xchClip 심벌의 인스턴스를 추가한다.
6. 그 인스턴스의 이름을 xch로 설정한다.
7. 변수의 값을 설정하기 위한 버튼, 텍스트 필드 및 기타 인터페이스 요소를 만든다.
8. xch 인스턴스에서 설정한 변수는 자동으로 스마트 클립의 매개변수로 들어간다. 다음과 같은 코드를 입력하면 두 개의 사용자 정의 매개변수의 값을 설정할 수 있다.

```
xch.param1 = value1;  // 어떤 데이터형의 값이라도 사용할 수 있다.
xch.param2 = value2;
```

9. 사용자 정의 인터페이스 .swf 파일에서 스마트 클립 매개변수에 저장된 값을 집어넣기 위해 그 매개변수에 xch의 속성을 대입한다. 다음과 같은 코드를 사용하면 param1Input 텍스트 필드에 스마트 클립에 있는 param1의 값을 대입할 수 있다.

```
param1Input = xch.param1;
```

10. .swf 파일을 저장한다(File → Export Movie).

사용자 정의 인터페이스를 스마트 클립에 추가하기

이제 인터페이스 .swf 파일을 만들었으니 이 파일을 스마트 클립에 추가하자.

1. 사용자 정의 인터페이스로 사용할 .swf 파일을 선택하고 스마트 클립이 들어있는 원래의 .fla 파일로 돌아온다.
2. 라이브러리에서 원하는 스마트 클립을 선택한다.
3. Options → Define Clip Parameters를 선택하면 Define Clip Parameters 대화상자가 나타난다.
4. Link to Custom UI 박스에 사용자 정의 인터페이스로 사용할 .swf 파일의 위치를 입력한다. 이 때 현재 .fla 파일이 저장된 디렉토리를 기준으로 한 상대경로를 입력해야 한다(또는 폴더 버튼을 클릭하여 .swf 파일을 직접 선택할 수도 있다).

스마트 클립 사용법

무비 클립에 클립 매개변수를 추가하고 나면 그 무비 클립은 공식적인 '스마트 클립'이 된다. 스마트 클립은 라이브러리에서 특별한 아이콘()으로 표시된다.

무비에서 스마트 클립 인스턴스를 사용하려면 다음과 같이 하면 된다.

1. 라이브러리에서 스테이지로 스마트 클립을 끌어다 놓는다.
2. Window → Panels → Clip Parameters를 선택한다.
3. 클립에서 표준 인터페이스를 사용한다면 다음과 같은 단계를 거쳐서 각 매개변수 값을 설정한다.
   a. Default 매개변수인 경우에는 매개변수 값을 더블클릭하고 적절한 문자열이나 숫자를 입력한다.
   b. Array 매개변수인 경우에는 매개변수 값을 더블클릭한다. Values 대화상자가 나타나면 각 배열 원소의 값을 더블클릭하고 적절한 문자열이나 숫자를 입력한다. 배열 원소의 값을 모두 설정하고 나면 OK를 클릭한다.
   c. Object 매개변수인 경우에도 매개변수 값을 더블클릭한다. Values 대화상자가 나타나면 각 객체 속성 값을 더블클릭하고 적절한 문자열이나 숫자를 입력한다. 값을 모두 설정하고 나면 OK를 클릭한다.
   d. List 매개변수인 경우에도 매개변수 값을 더블클릭하고 주어진 옵션 가운데 한 가지를 선택한다.
4. 클립에서 사용자 정의 인터페이스를 사용한다면 사용자 정의 인터페이스에 주어진 도구를 이용하여 클립의 매개변수를 설정한다.

앞으로 배울 내용

다음 장에서는 액션스크립트와 간단한 서버 애플리케이션을 결합하여 플래시 폼을 만들어 보자.