18

온스크린 텍스트 필드

플래시는 기본적으로 시각 위주 환경이기 때문에 무비에서 사용자에게 화면을 통해 사용자에게 정보(온스크린 정보)를 제공한다. 또한 인터랙티브한 플래시의 성격으로 인해 무비에서 GUI를 통해 사용자의 정보를 얻는 경우도 많이 있다. 텍스트 필드는 플래시에서 어떤 값을 화면에 표시하거나 사용자가 플래시 무비에 어떤 값을 입력할 수 있게 해주는 도구이다.

텍스트 필드는 화면에 출력되는 변수의 값을 설정하거나 현재 화면에 출력된 값을 알아내는 도구로 사용할 수 있다. 텍스트 필드에는 두 가지 종류가 있다. 하나는 사용자에게 정보를 보여주기 위한 도구인 동적 텍스트 필드이고, 나머지 하나는 사용자가 입력한 정보를 알아내기 위한 사용자 입력 텍스트 필드이다.

동적 텍스트 필드

동적 텍스트 필드는 변수를 들여다볼 수 있게 해주는 유리창과 비슷한 것으로, 특정 변수의 값을 텍스트 문자열로 화면에 표시해주는 역할을 한다. 동적 텍스트 필드는 플래시의 Text 도구를 이용하여 만든다. 하지만 일반적인 정적 텍스트와는 달

리 동적 텍스트 필드의 내용은 어떤 변수에 연결되어 있기 때문에, 액션스크립트를 통해 그 값을 변경시키거나 저장된 값을 알아낼 수도 있다.

동적 텍스트 필드 만들기

동적 텍스트 필드는 다음과 같이 만들 수 있다.

1. Text 도구를 선택한다.
2. 스테이지 위에 텍스트를 출력할 위치를 클릭하고 마우스를 끌어서 직사각형을 만든다. 이렇게 만든 직사각형에 의해 새로 만드는 텍스트 필드의 크기가 결정된다.
3. Text → Options를 선택한다. Text Options 패널이 화면에 나타난다.
4. Text Type 메뉴에서 Dynamic Text를 선택한다(이외에 Static Text와 Input Text 옵션이 있다).
5. Variable에 동적 텍스트 필드의 이름을 입력한다. 이 때 '2장. 변수'에서 배운 변수 명명법을 기준으로 이름을 만들면 된다.

동적 텍스트 필드를 만들고 나면 새로운 필드의 옵션을 설정해야 한다. 옵션을 설정하는 방법은 잠시 후에 알아보자.

동적 텍스트 필드의 내용

텍스트 필드를 만들고 나면 그 텍스트 필드를 이용하여 화면에 어떤 값을 출력할 수 있다. 예를 들어 myText라는 동적 텍스트 필드를 만들었다면 다음과 같은 선언문을 이용하여 그 텍스트 필드의 내용을 설정할 수 있다.

```
myText = 10;                    // myText 텍스트 필드에 숫자를 출력한다.
myText = "Welcome to my web site"; // 문자열을 출력한다.

var msg = "Please make a selection";
myText = msg;                   // myText에 msg 변수의 값을 출력한다.
```

myText 변수의 값이 바뀔 때마다 myText 동적 텍스트 필드가 자동으로 업데이트되어 새로운 값이 화면에 표시된다. 하지만 플래시에서는 어떤 값을 화면에 표시하기 전에 모든 값을 문자열 형태로 변환한다. 따라서 실제 출력되는 내용은 〔표 3-2〕에 나온 문자열 변환 규칙에 따라 결정된다.

일반적인 변수와 마찬가지로 텍스트 필드도 그 필드가 속한 무비 클립 타임라인에 속한다. 원격 무비 클립 타임라인에 있는 동적 텍스트 필드를 사용하려면 2장의 '다른 타임라인에 있는 변수 사용하기' 부분에서 설명한 방법을 이용해야 한다.

동적 텍스트 필드 값 알아내기

동적 텍스트 필드의 값을 확인할 때는 그냥 그 필드의 이름을 사용하기만 하면 된다. 예를 들어 myTextField가 무비에 포함된 동적 텍스트 필드라면 다음과 같은 식으로 그 필드 안에 들어있는 값을 다른 변수에 대입할 수 있다.

```
welcomeMessage = myTextField;
```

텍스트 필드 값을 가져와서 그 값을 다른 변수에 대입하는 작업은 보통 선언문 한 개로 모두 처리한다. += 연산자를 이용하면 현재 텍스트 필드의 내용에 다른 텍스트를 추가할 수도 있다.

```
// 텍스트 필드 값을 설정한다.
myTextField = "Today's Headlines...";
// 새로운 메시지를 만든다.
var newText = "Update! The Party Has Been Cancelled!"
// 원래 있던 텍스트 필드에 저장된 값 뒤에 새로운 메시지를 덧붙인다.
myTextField += newText;
```

사용자 입력 텍스트 필드

사용자 입력 텍스트 필드는 무비가 실행될 때 사용자가 그 텍스트 필드를 편집할 수 있다는 점을 제외하면 동적 텍스트 필드와 똑같다. 사용자 입력 텍스트 필드에는 사용자가 어떤 값을 입력하면 새로운 값이 그 필드에 해당하는 변수에 저장된

다. 이렇게 하면 액션스크립트에서는 그 값을 확인하고 그 값으로 다른 작업을 처리할 수 있다. 사용자 입력 텍스트 필드는 방명록, 폼, 비밀번호 입력 필드를 비롯하여 사용자가 입력한 정보가 필요한 곳이라면 어디든지 사용할 수 있다.

사용자 입력 텍스트 필드 만들기

사용자 입력 텍스트 필드를 만드는 방법은 '동적 텍스트 필드 만들기'에서 설명한 방법과 거의 똑같다. Text Type 메뉴에서 Dynamic Text 대신 Input Text를 선택하기만 하면 사용자 입력 텍스트 필드가 만들어진다.

사용자 입력 텍스트 필드의 내용 바꾸기

동적 텍스트 필드와 마찬가지로 사용자 입력 텍스트 필드에서도 다음과 같이 대입 선언문을 이용하여 텍스트 필드 값을 설정하기만 하면 그 필드의 값을 변경할 수 있다.

```
myInputText = "Type your name here";
```

사용자 입력 텍스트 필드는 보통 데이터를 출력하기보다는 데이터를 입력받기 위한 용도로 주로 쓰이기 때문에, 사용자가 입력할 내용에 대한 기본값을 설정하는 경우를 제외하면 액션스크립트에서 값을 설정하는 일이 거의 없다.

사용자 입력 텍스트 필드 값 알아내기

텍스트 필드 값을 이용할 때는 스크립트에서 그 필드의 이름을 그대로 사용하면 된다. 예를 들어 myInput이라는 입력 텍스트 필드 값을 화면에 표시하고 싶다면 다음과 같은 코드를 사용하면 된다.

```
trace(myInput);
```

사용자가 사용자 입력 텍스트 필드에 입력한 내용은 모두 문자열 데이터로 간주되기 때문에, 다음과 같이 두 개의 사용자 입력 텍스트 필드에서 입력받은 값을 더하여 그 합을 구하는 프로그램을 만들 때는 직접 숫자형으로 값을 변환해야 한다.

```
// myFirstInput을 5로, mySecondInput을 10으로 설정했다고 가정하고
// 두 필드의 값의 합을 구해 보자.
// + 연산자가 문자열을 합치는 연산자로 작용하기 때문에 "Total: 510"이라는
// 엉뚱한 결과가 나온다. 따라서 다음과 같이 하면 안 된다.
myOutput = "Total: " + (myFirstInput + mySecondInput);
// 제대로 된 결과를 얻고 싶다면 다음과 같이 필드의 값을 숫자로 변환해야 한다.
myOutput = "Total: " + (parseFloat(myFirstInput) + parseFloat(mySecondInput));
```

사용자 입력 텍스트 필드와 폼

사용자 입력 텍스트 필드는 펄 스크립트와 같은 서버측 애플리케이션에 보낼 내용을 입력받는 폼에서 많이 쓰인다. loadVariables()를 이용하여 변수를 서버에 보낼 때는 현재 무비 클립에 있는 변수만 전송된다. 따라서 어떤 폼에 사용자 입력 텍스트 필드가 있으면 그 필드를 서버에 한꺼번에 보낼 수 있도록 한 무비 클립에 모두 모아두는 것이 좋다. loadVariables()에 대한 내용은 '17장. 플래시 폼'과 '3부. 레퍼런스'에 있다.

텍스트 필드 옵션

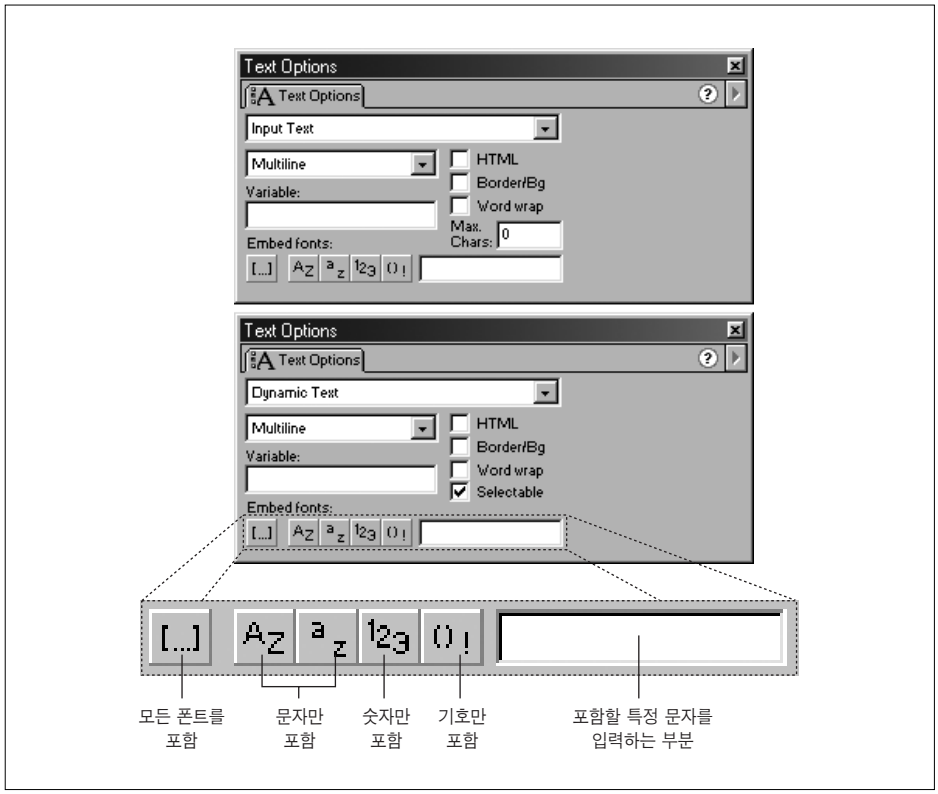
동적 텍스트 필드와 사용자 입력 텍스트 필드에서 화면 출력 및 입력 기능과 관련된 옵션은 거의 같다(물론 완전히 같진 않다). 사용자 입력 텍스트 필드와 동적 텍스트 필드의 Text Options 패널은 〔그림 18-1〕과 같다.

Line Display 옵션

텍스트 필드의 레이아웃과 입력 스타일을 설정하거나 사용자가 입력한 내용을 화면에 보이지 않도록 하려면 Line Display 메뉴를 사용하면 된다.

Single Line

Single Line 옵션을 선택하면 사용자가 필드의 텍스트에 한 줄까지만 입력할 수 있다. 이렇게 하면 입력할 때 엔터 키를 사용할 수 없다.



〔그림 18-1〕 Text Options 패널

Single Line 옵션을 사용하면 저작 도구에서 줄을 바꾸지 않고 입력한 텍스트에도 영향을 미친다. 플래시 무비를 만들 때는 자동으로 줄바꿈이 되지만 플레이어에서 재생될 때는 텍스트의 줄이 바뀌지 않는다. 따라서 필드가 오른쪽 경계선을 넘어가도 텍스트가 한 줄로만 표시된다. 하지만 플래시를 만들 때 줄바꿈 문자를 따로 넣어주면 Single Line 설정을 이용한 경우에도 저작 도구에서 입력한 내용이 그대로 화면에 표시된다.

Single Line 옵션은 주로 사용자 입력 텍스트 필드에서만 사용하는 옵션이다. 동적 텍스트 필드에서는 Word Wrap 옵션을 따로 선택하지 않으면 Multiline 옵션과 똑같다. 예를 들어 텍스트 필드 변수 값을 "this is\na test"라고 설정하면, "this is"와 "a test"가 서로 다른 줄에 표시된다.

Multiline

Multiline 옵션을 선택하면 사용자가 텍스트 필드에 입력할 때 여러 줄에 걸쳐서 입력할 수 있다. 또한 사용자 입력 필드에서 엔터 키를 사용할 수도 있다.

Multiline을 사용하더라도 Word Wrap 옵션을 사용하지 않으면 동적 텍스트 필드로 출력하는 내용에는 전형 영향을 미치지 않는다. Word Wrap 옵션을 선택하지 않은 상태에서는 Multiline으로 설정한 텍스트 필드도 Single Line으로 설정한 텍스트 필드와 다를 것이 없다.

Password

Password 옵션은 사용자가 입력한 문자가 화면에 출력되지 않도록 하는 용도로 쓰이며 사용자 입력 텍스트 필드에만 적용된다. 스페이스를 포함한 모든 문자가 별표(*)로 표시된다는 점을 제외하면 Single Line 옵션을 사용한 텍스트 필드와 똑같다. 예를 들어 "hi there"는 "********"로 출력된다.

플래시 인터페이스의 특이한 성질을 이용하면 Password로 설정한 텍스트 필드에서도 줄을 바꿀 수 있다. Line Display를 Multiline으로 설정하고 Word Wrap을 선택한 후에 Line Display를 Password로 설정하면, Word Wrap 설정이 유지되므로 자동 줄바꿈 기능을 이용할 수 있다. 하지만 여러 줄에 걸쳐서 비밀번호를 입력할 수 있도록 하면 대부분의 사용자가 혼란스러워할 수 있기 때문에 가급적이면 이렇게 하지 않는 것이 좋다.

Variable

Text Options 패널의 Variable 옵션은 동적 텍스트 필드 또는 사용자 입력 텍스트 필드의 이름을 적어주는 곳이다. 텍스트 필드를 액션스크립트로 조작하고 싶다면 이름을 꼭 정해주어야 한다. 텍스트 필드 이름을 정할 때도 '2장. 변수' 및 '14장. 렉시컬 구조'에서 나온 규칙을 지켜야 한다.

Border/Bg

Text Options 패널에서 Border/Bg를 설정하면 텍스트 필드 주변에 검은색 테두리가 생기고 필드 부분은 흰색이 된다. 하지만 색깔과 스타일은 마음대로 변경할 수

없다. 텍스트 필드의 배경을 고치고 싶다면 Border/Bg 옵션을 선택하지 않은 상태로 수동으로 텍스트 필드 뒤에 도형을 그리는 방법을 사용해야 한다.

Word Wrap

Line Display 옵션으로 Multiline을 선택하고 이 옵션을 선택하면 필드의 너비보다 넓은 텍스트는 자동으로 줄을 바꿔서 텍스트 너비를 넘어서지 않도록 할 수 있다. 이 설정은 사용자가 입력한 텍스트와 액션스크립트를 통해 출력되는 텍스트에 모두 적용된다.

Multiline으로 설정된 상태에서 Word Wrap 옵션을 선택하고 나서 Single Line을 다시 선택하면 Word Wrap이 계속해서 적용된다. 만약 한 줄을 꽉 채울 때 줄이 자동으로 바뀌지 않도록 하고 싶다면, 잊지 말고 Sinle Line을 선택하기 전에 Word Wrap 옵션 선택을 해제한다.

Selectable

동적 텍스트 필드에 들어있는 텍스트를 마우스나 키보드를 이용하여 선택하려면 필드의 Selectable 옵션이 설정되어 있어야 한다. 하지만 필드를 선택하더라도 복사만 할 수 있을 뿐 잘라내거나 다른 방식으로 편집할 수는 없다. 사용자 입력 텍스트 필드는 언제나 선택할 수 있고 사용자가 원하는 대로 텍스트를 복사하거나 잘라내거나 편집할 수 있기 때문에 Selectable 옵션이 무의미하다.

플래시에서 텍스트를 복사하거나 잘라내거나 붙여넣을 때는 윈도우에서 오른쪽 클릭을 했을 때 나오는 메뉴(매킨토시에서는 Ctrl 키를 누른 상태에서 클릭했을 때 나오는 메뉴)를 이용해야 한다. Ctrl-C, Ctrl-V(윈도우)나 Cmd-C, Cmd-V(매킨토시)와 같은 단축키는 사용할 수 없다.

Max Characters

Max Characters 옵션은 사용자 입력 텍스트 필드에서만 사용할 수 있으며, 사용자가 텍스트 필드에 입력할 수 있는 텍스트의 최대 길이를 제한하는 데 쓰인다.

Max Characters의 기본값은 0이다(0으로 설정하면 텍스트를 무제한으로 입력할 수 있다). 다른 숫자를 입력하면 입력할 수 있는 문자의 개수를 그 숫자만큼으로 제한할 수 있다.

Max Characters는 데이터의 형식이 따로 정해져 있는 폼에서 주로 쓰인다. 예를 들어 날짜를 입력하는 부분에서 연, 월, 일을 각각 두 자리 숫자로 입력하도록 할 때 이 옵션을 사용할 수 있다.

Embed Fonts

기본적으로 모든 동적 텍스트 필드와 사용자 입력 텍스트 필드에서는 사용자의 시스템에 있는 폰트를 사용한다. 시스템 폰트를 사용할 때 텍스트 필드의 Character 패널에서 정해준 폰트가 사용자의 시스템에도 있다면 무비를 만들 때 사용한 폰트가 그대로 쓰인다(하지만 안티앨리어싱은 적용되지 않는다). 사용자의 시스템에 지정된 폰트가 없으면 다른 폰트를 사용하게 되는데, 상황에 따라 다른 폰트를 사용하면 무비가 이상해지는 경우도 있다.

항상 같은 폰트를 사용하고 싶다면 Embed Fonts 옵션([그림 18-1]에서 확대된 부분)을 이용하여 폰트를 무비에 포함시킬 수 있다.

폰트를 무비에 포함시키는 방식은 크게 세 종류로 나눌 수 있다.

- [...] 버튼을 선택하여 폰트 전체를 포함시키는 방식
- AZ, az, 123과 ()! 버튼을 이용하여 문자, 숫자, 기호를 필요에 따라 포함시키는 방식
- 주어진 필드에 입력한 특정 문자만을 포함시키는 방식

영문 폰트를 통째로 집어넣으면 무비의 파일 크기가 20~30KB 정도 더 커진다(아시아권 폰트를 포함시키면 훨씬 더 커진다). 일부 문자만을 사용하는 경우에는 필요한 문자만 포함시켜 파일 크기를 줄일 수 있다. 하지만 무비에 포함시키지 않은 문자는 액션스크립트를 통해 화면에 출력할 수도 없고 사용자가 그러한 문자를 입력할 수도 없다. 따라서 특정 문자를 입력하지 못하도록 하고 싶다면 이런 기능을 활용해도 된다.

여러 텍스트 필드에서 같은 폰트를 사용한다고 하더라도 각 텍스트 필드별로 Embed Fonts 옵션을 따로 설정해야 한다. 하지만 여러 텍스트 필드에서 같은 폰트를 사용하면 무비에 한 벌의 폰트만 포함되기 때문에 텍스트 필드의 개수와는 상관없이 늘어나는 파일의 크기는 일정하다. 한꺼번에 여러 개의 텍스트 필드에 대한 Embed Fonts 옵션을 적용하고 싶다면 원하는 필드를 한꺼번에 선택하고 나서 Embed Fonts 옵션을 설정하면 된다.

임베드된 폰트를 사용하는 텍스트 필드에서는 안티앨리어싱이 적용된다. 따라서 크기가 10포인트 미만인 폰트는 무비에 포함시키지 않는 것이 좋다. 대부분의 폰트에서 10포인트 미만인 글자에 안티앨리어싱을 적용하면 글씨를 알아보기 힘들어지기 때문이다. 폰트에 안티앨리어싱이 적용되지 않도록 하려면 Embed Fonts 옵션을 선택하지 않으면 된다. 이렇게 하면 시스템 폰트를 사용하게 되고, 시스템 폰트에는 안티앨리어싱이 적용되지 않기 때문이다.

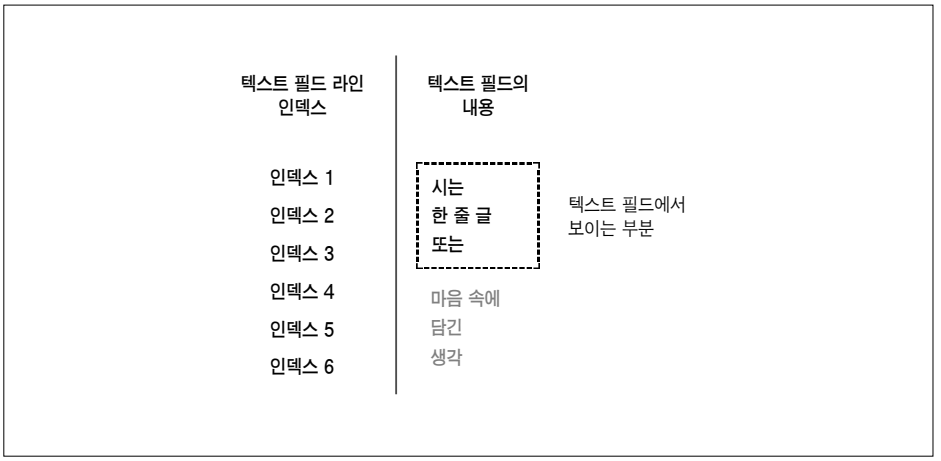
텍스트 필드에 들어가는 내용을 회전시키거나 매스킹을 적용하면 폰트를 무비에 포함시켜야만 그 내용이 화면에 출력된다. 따라서 시스템 폰트를 이용하면 텍스트 필드를 회전시키거나 매스킹할 수 없다.

텍스트 필드의 폰트에 대한 자세한 내용은 'HTML 형식 출력' 부분을 참조하기 바란다.

텍스트 필드 속성

텍스트 필드에 들어가는 텍스트 분량이 필드로 볼 수 있는 영역보다 더 많다면 텍스트의 일부가 화면에 표시되지 않을 수도 있다. 하지만 화면에 나타나지 않는 부분도 여전히 텍스트 필드에 포함된다. 가려진 부분을 보고 싶다면 필드를 클릭하고 화살표 키를 누르면 모든 내용을 볼 수 있다. 하지만 사용자가 텍스트 필드에 있는 텍스트를 일일이 클릭해서 화살표 키로 가려져 있는 부분을 확인하도록 하는 것보다는 scroll과 maxscroll 속성을 이용하여 텍스트를 스크롤할 수 있도록 버튼을 만들어 주는 것이 좋다. 이 두 속성에서는 모두 인덱스 번호를 이용하여 텍스트 필드의 특정 라인을 가리키도록 한다. 맨 윗줄의 번호가 1부터 시작하며 아래로 한 줄씩

내려갈수록 인덱스 번호가 1씩 커진다. 화면으로 볼 수 없는 부분의 인덱스도 같은 식으로 정해진다.



〔그림 18-2〕 텍스트 필드 라인 인덱스

scroll 속성

scroll 속성은 현재 텍스트 필드에 표시되는 내용 중 가장 윗줄의 인덱스를 나타낸다. 이 값은 textFieldName.scroll과 같은 식으로 사용하면 된다.

텍스트 필드에 들어있는 내용이 화면에 표시할 수 있는 것보다 많다면 scroll 속성을 설정하여 화면에 표시되는 영역을 조절할 수 있다. 예를 들어 〔그림 18-2〕에 나온 텍스트 필드의 scroll 속성을 3으로 설정하면 텍스트 필드에는 다음과 같이 나타난다.

```
또는
마음 속에
담긴
```

maxscroll 속성

필드에서 스크롤할 수 있는 가장 큰 인덱스(즉 마지막 줄이 화면에 표시되는 인덱스)는 maxscroll 속성으로 알 수 있다. maxscroll 속성의 값은 필드의 마지막 줄의 인덱스에서 한 번에 화면에 표시할 수 있는 줄 수를 뺀 값에 1을 더한 숫자가 된다. 예를 들어 [그림 18-2]에 나온 텍스트 필드의 maxscroll 속성은 4이다(마지막 줄의 인덱스인 6에서 한 번에 화면에 표시할 수 있는 줄 수인 3을 빼고 1을 더하면 4가 되기 때문이다). maxscroll은 텍스트 줄 수와 같지 않다는 점에 주의하자.

maxscroll 속성 값은 testFieldName.maxscroll과 같은 식으로 알아낼 수 있다(이 속성에는 다른 값을 대입할 수 없다).

텍스트 스크롤 코드

scroll과 maxscroll 속성을 이용하면 텍스트 필드를 스크롤할 수 있다. 다음과 같은 코드를 이용하면 사용자가 버튼을 클릭할 때마다 텍스트를 한 줄씩 아래로 스크롤할 수 있다.

```
on (press) {
  if (textField.scroll < textField.maxscroll) {
    textField.scroll++;
  }
}
```

아래 코드는 버튼을 클릭하면 텍스트를 한 줄씩 위로 스크롤하는 코드이다.

```
on (press) {
  if (textField.scroll > 1) {
    textField.scroll--;
  }
}
```

무비에서 사용할 수 있는 간단한 스크롤 버튼의 예제를 보고 싶다면 온라인 코드 창고에 있는 스크롤러의 샘플을 다운로드하여 실행해 보자.

플래시 저작 도구와 함께 발표된 플래시 5 플레이어 빌드 30에는 텍스트 필드 표시 부분에 버그가 있다. 안티앨리어싱을 적용한 텍스트 필드를 스크롤하면 스크롤된 텍스트가 제대로 지워지지 않는 경우가 종종 있다. 이러한 문제를 해결하려면 텍스

트 필드에 테두리를 추가하여 제대로 지워지지 않고 남아 있는 텍스트를 지워야 한다. 이 버그는 2000년 12월에 공개된 플래시 5 플레이어 빌드 41에서 수정되었다. getVersion()이라는 전역 함수를 사용하면 플레이어의 버전을 확인할 수 있다.

_changed 이벤트

플래시 4와 플래시 5에서는 사용자 입력 텍스트 필드의 내용이 바뀌는 것을 _changed 이벤트로 알아낼 수 있다(이 이벤트는 공식적으로 문서화되지 않는 이벤트이다). _changed 이벤트는 사용자 입력 텍스트 필드의 내용이 바뀔 때마다 플래시 4 형식의 서브루틴을 실행시키는 역할을 한다. 텍스트 필드에 대한 _changed 이벤트는 다음과 같이 만든다.

1. 임의의 타임라인에 입력 텍스트 필드를 만든다.
2. 텍스트 필드의 이름을 myField로 바꾼다.
3. 텍스트 필드와 같은 타임라인에 myField_changed라는 프레임을 만든다.
4. myField_changed 프레임에 아무 코드나 입력한다. 예를 들어 다음과 같은 코드를 넣어도 된다.

```
trace("myField was changed");
```

5. Control → Test Movie를 선택하여 무비를 실행시킨다.
6. myField 텍스트 필드에 문자열을 입력한다. 그러면 myField_changed 프레임의 코드가 실행되므로 Output 창에 "myField was changed"라는 문자열이 출력된다.

물론 myField 대신 아무 이름이나 사용해도 된다. 필드에 어울릴만한 이름을 붙이고 그 필드 이름 뒤에 _changed를 덧붙인 프레임 레이블을 만들기만 하면 된다. 액션스크립트를 이용하여 텍스트 필드의 값을 설정할 때는 _changed 이벤트가 발생되지 않는다. _changed 이벤트는 사용자가 키를 누를 때만 생성된다.

_changed 이벤트는 공식적으로 발표된 기능이 아니기 때문에 플래시의 다음 버전이 나올 때는 텍스트 필드와 관련된 이벤트를 처리하는 새로운 표준이 정해질 수도 있다.

HTML 지원

Character 패널을 이용하면 텍스트 필드의 폰트 크기, 종류, 스타일을 설정할 수는 있지만 텍스트 필드 전체의 속성을 한꺼번에 정해야만 한다. 텍스트에 각 문자별로 스타일을 설정하고 하이퍼텍스트 링크를 추가하려면 HTML을 사용해야 한다(이 기능은 플래시 5에서 새로운 텍스트 필드 관련 기능으로 도입되었다).

동적 텍스트 필드나 사용자 입력 텍스트 필드에 HTML을 모두 사용할 수 있긴 하지만 사용자 입력 텍스트 필드에서는 HTML을 잘 쓰지 않는다. Text Options 패널에서 HTML 옵션을 선택하면 텍스트 필드에 HTML 기능을 추가할 수 있다.

플래시의 텍스트 필드에서는 HTML 태그 중에서 <B>, <I>, <U>, <FONT>, <P>, <BR>, <A>만 사용할 수 있다.

〈B〉 (볼드)

<B> 태그를 사용하면 텍스트가 볼드체로 표시된다. 하지만 사용 중인 폰트에서 볼드체가 지원되는 경우에만 제대로 작동한다.

```
<B>This is bold text</B>
```

〈I〉 (이탤릭)

<I> 태그를 사용하면 텍스트가 이탤릭체로 표시된다. 이 기능도 해당 폰트에서 이탤릭체를 지원해야만 제대로 작동한다.

```
<I>This is italic text</I>
```

〈U〉 (밑줄)

<U> 태그를 사용하면 텍스트에 밑줄이 첨가된다. 예를 들면 다음과 같다.

```
<U>This is underlined text</U>
```

플래시에서는 텍스트에 링크를 연결해도 밑줄이 자동으로 들어가지 않기 때문에 하이퍼링크를 사용할 때는 <U> 태그를 따로 추가해 주는 것이 좋다.

```
<A HREF="http://www.thesquarerootof-1.com"><U>Click here</U>
</A> to visit a neat site.
```

〈FONT〉 (폰트 제어)

<FONT> 태그에서는 아래 세 가지 속성을 지원한다.

FACE

FACE 속성은 사용할 폰트의 이름을 지정하는 데 쓰인다. 하지만 HTML과는 달리 플래시에서는 여러 개의 폰트 종류를 지정할 수 없다. 만약 FACE 속성에 여러 폰트를 지정하면 그 중 첫 번째 폰트만 사용한다. 예를 들어 <FONT FAC=“Arial, Helvetica”>my text</FONT> 같은 코드를 사용하면 Arial 폰트가 없더라도 Helvetica 폰트를 사용하지 않는다. 맨 앞에 지정한 폰트가 없으면 기본 폰트를 사용한다.

SIZE

SIZE 속성은 텍스트의 크기를 고정된 포인트 크기(예: <FONT SIZE=“18”>) 또는 상대적인 크기로 설정할 수 있다. 상대적인 크기를 사용할 때는 숫자 앞에 + 또는 - 부호를 덧붙여서 지정할 수 있으면 Character 패널에서 선택한 텍스트 크기를 기준으로 폰트의 크기를 조절한다. 예를 들어 Character 패널에서 14포인트를 선택했을 때 <FONT SIZE=“-2”>라는 태그를 사용하면 12포인트짜리 글씨로 출력된다.

COLOR

COLOR 속성은 텍스트의 색을 바꾸는 데 쓰이며, 이 때 색은 # 부호 뒤에 여섯 자리 16진수로 지정한다(예: <FONT COLOR=“#FF0000”>this is red text</FONT>). 16진수 색 표현에서는 세 개의 두 자리 16진수(00부터 FF까지)로 RGB(빨간색, 녹색, 파란색) 값을 각각 표시한다. 플래시의 COLOR 속성은 HTML의 COLOR 속성보다 더 까다롭다. # 부호를 반드시 사용해야 하며 “green”, “blue” 같은 이름을 쓸 수도 없다.

<FONT> 태그를 사용한 예는 다음과 같다.

```
<FONT FACE="Arial">this is Arial</FONT>
<FONT FACE="Arial" SIZE="12">this is 12pt Arial</FONT>
<FONT FACE="Lucida Console" SIZE="+4" COLOR="#FF0000">this is red,
+4pt Lucida Console</FONT>
```

플래시의 폰트와 관련된 자세한 내용은 잠시 뒤에 나오는 'HTML 형식 출력'에서 다루고 있다.

〈P〉 (문단 나누기)

<P> 태그는 문단을 나누는 기능을 하지만 플래시의 <P> 태그는 HTML의 <P> 태그와는 약간 다르다. 무엇보다도 <P> 태그를 사용하더라도 </P> 태그를 이용하여 태그를 종료하지 않으면 줄이 바뀌지 않는다는 점에서 HTML에서의 <P>와 차이가 난다. 아래 예를 통해 <P> 태그의 기능이 플래시와 웹 브라우저에서 어떻게 다른지 확인해 보자.

```
I hate filling out forms. <P>So sometimes I don't.
// 플래시
I hate filling out forms. So sometimes I don't.
// 웹 브라우저
I hate filling out forms.
So sometimes I don't.
```

줄을 바꾸고 싶다면 </P> 태그를 이용하여 <P> 태그를 종료시켜야 한다.

```
<P> I hate filling out forms.</P> So sometimes I don't.
```

또한 웹 브라우저에서는 <P> 태그를 이용해도
 태그를 사용한 것과 마찬가지로 줄바꿈 문자가 한 번만 들어간 효과가 나타난다. 일반적으로 웹 브라우저에서는 줄바꿈 문자가 두 번 들어간 것과 같은 효과가 나타난다. 아래 예를 살펴보자.

```
<P>This is line one.</P><P>This is line two.</P>
```

플래시에서는 위와 같은 텍스트가 다음과 같은 형식으로 출력된다.

```
This is line one.
This is line two.
```

하지만 웹 브라우저에서는 다음처럼 줄이 두 번 바뀐다.

```
This is line one.

This is line two.
```

플래시의 <P> 태그는 웹 브라우저의 <P> 태그와는 약간 다르게 동작하기 때문에 <P> 대신
 태그를 사용하는 경우가 많다. 하지만 가운데, 오른쪽, 왼쪽 정렬을 선택할 수 있다는 장점 때문에 <P> 태그도 요긴하게 쓰인다.

```
<P ALIGN="CENTER">Centered text</P>
<P ALIGN="RIGHT">Right-justified text</P>
<P ALIGN="LEFT">Left-justified text</P>
```

〈BR〉 (줄 바꾸기)

 태그는 텍스트 본체에서 줄을 바꿔주는 역할을 하며, newline 키워드나 \n 이스케이프 시퀀스와 똑같은 기능을 한다. 아래 예를 살펴보자.

```
This is line one. <BR>This is line two.
This is line one. \nThis is line two.
```

위 두 코드 중 어느 쪽을 사용하더라도 플래시에서는 다음과 같이 출력된다.

```
This is line one.
This is line two.
```

〈A〉 (앵커 또는 하이퍼텍스트 링크)

<A> 태그는 하이퍼텍스트 링크를 만드는 데 쓰인다. 사용자가 <A> 태그로 둘러싼 텍스트를 클릭하면 태그의 HREF 속성에서 지정한 문서가 브라우저에 나타난다. 만약 플레이어가 독립 모드로 돌아가고 있다면 시스템의 기본 웹 브라우저가 실행되면서 그 웹 브라우저에서 그 문서를 연다.

<A> 태그는 보통 다음과 같은 형식으로 쓰인다.

```
<A HREF="documentToLoad.html">linked text</A>
```

예를 들어 퀘이크 3 사이트로 가는 링크를 만들고 싶다면 다음과 같은 태그를 사용하면 된다.

```
<A HREF="http://www.quake3arena.com/">nice game</A>
```

HTML에서와 마찬가지로 플래시의 앵커 태그에서 사용하는 URL에서도 절대 경로와 상대 경로를 모두 사용할 수 있다. 일반적으로 앵커 태그로 연결된 링크를 선택하면 현재 무비가 로딩된 웹 브라우저에 새로운 링크가 바로 연결되므로 현재 무비가 닫힌다. 하지만 앵커 태그를 이용할 때 새 창에서 링크를 열도록 할 수도 있다. 아래와 같이 TARGET 속성을 이용하면 새로 열리는 창의 이름을 지정할 수 있다.

```
<A HREF="documentName" TARGET="windowName">linked text</A>
```

windowName이라는 이름을 가진 창이 없다면 브라우저에서 새로운 창을 열고 그 창의 이름을 windowName로 설정한다. 모든 문서를 새로운 창에서 열고 싶다면 다음과 같이 _blank 키워드를 사용하면 된다.

```
<A HREF="mypage.html" TARGET="_blank">linked text</A>
```

TARGET 속성을 이용하여 창을 선택하더라도 새로운 창의 크기나 도구막대 상태와 같은 것은 마음대로 조절할 수 없다. 링크를 통해 새로 열리는 창의 크기를 조절하려면 자바스크립트를 이용해야 한다. 자바스크립트를 이용하여 창의 크기나 특성을 조절하는 방법은 아래 사이트에 나와 있다.

```
http://www.moock.org/webdesign/flash
```

액션스크립트에서 자바스크립트를 사용하는 방법을 알고 싶다면 3부에서 전역 함수인 fscommand()와 getURL()에 대한 내용을 읽어보거나 이 장의 뒷부분에 나와 있는 'HTML 링크에서 자바스크립트 실행시키기'를 참조하기 바란다.

TARGET 속성을 이용하면 다른 프레임에서 문서를 불러올 수도 있다.

```
<A HREF="documentName" TARGET="frameName">linked text</A>
```

플래시의 앵커 태그도 HTML의 앵커 태그와는 약간 다르다. 예를 들어 플래시에서는 NAME 속성을 사용할 수 없기 때문에, 텍스트 내부의 다른 위치로 링크를 연결할 수 없다. 게다가 플래시에서는 링크 부분에 밑줄을 긋거나 하는 식으로 링크를 자동으로 강조하지 않는다. 따라서 링크에 밑줄을 추가하거나 링크의 색깔을 바꾸

려면 앞에서 설명한 <U> 태그나 <FONT> 태그를 이용하여 직접 링크를 강조하는 수밖에 없다.

앵커 태그 탭 순서

플래시 5에서 앵커 태그는 무비의 탭 순서에 포함되지 않기 때문에 키보드에서 탭 키를 누르는 것으로는 링크를 선택할 수 없다. 키보드만으로 링크를 선택할 수 있도록 만들려면 앵커 태그로 링크를 만드는 대신 버튼을 만들어서 그 버튼에 링크를 연결해야 한다.

속성 값에서 따옴표 사용하기

일반적으로 HTML 속성 값을 표기할 때는 작은따옴표나 큰따옴표 중 어느 쪽을 사용해도 상관없고 괄호를 아예 사용하지 않아도 된다. 대부분의 웹 브라우저에서는 아래에 나온 세 가지 형식 중 어느 것을 사용해도 무방하다.

```
<P ALIGN=RIGHT>
<P ALIGN='RIGHT'>
<P ALIGN="RIGHT">
```

하지만 플래시에서는 모든 속성 값을 지정할 때 반드시 따옴표를 사용해야 한다. 예를 들면 플래시에서는 <P ALIGN=RIGHT>와 같은 태그는 사용할 수 없다. 하지만 속성 값을 둘러싸는 따옴표로 둘러싸기만 하면 큰따옴표나 작은따옴표 중 아무거나 사용해도 된다. HTML을 이용하여 텍스트 필드 값을 설정할 때는 작은따옴표와 큰따옴표 중 하나는 문자열 전체를 감싸는 데, 나머지 하나는 속성 값을 감싸는 데 사용해야 한다. 예를 들면 다음과 같다.

```
/// 아래 두 구문은 모두 문제없이 실행된다.
myText = "<P ALIGN='RIGHT'>hi there</P>";
myText = '<P ALIGN="RIGHT">hi there</P>';
// 다음과 같이 하면 오류가 생긴다. 문자열 전체를 감쌀 때도 큰따옴표를 사용하고
// 속성 값을 감싸는 데도 큰따옴표를 사용했기 때문이다.
myText = "<P ALIGN="RIGHT">hi there</P>";
```

문자열을 만들 때 따옴표를 사용하는 방법에 대한 자세한 내용은 '4장. 원시 데이터형'의 '문자열 리터럴'에 나와 있다.

인식할 수 없는 태그와 속성

웹 브라우저와 마찬가지로 플래시에서도 인식할 수 없는 태그와 속성은 무시해 버린다. 예를 들어 플래시의 HTML 텍스트 필드에 다음과 같은 값을 대입했다고 가정해 보자.

```
<P>Please fill in and print this form</P>
<FORM><INPUT TYPE="TEXT"></FORM>
<P>Thank you!</P>
```

위와 같은 텍스트를 플래시에서 출력하면 다음과 같은 내용만 화면에 표시된다.

```
Please fill in and print this form
Thank you!
```

플래시에서는 <FORM>과 <INPUT> 태그를 지원하지 않기 때문에 이와 같은 태그는 그냥 무시한다. 마찬가지로 <TD>와 같은 태그를 사용하면 태그로 둘러싸인 내용은 출력되지만 태그는 무시된다.

```
myTextField = "<TABLE><TR><TD>table cell text</TD></TR></TABLE>";
```

위와 같은 코드를 사용하더라도 테이블 형식으로 출력되지 않고 일반적인 텍스트 형식으로 다음과 같이 출력된다.

```
table cell text
```

HTML 형식 출력

Text 도구를 이용하여 직접 텍스트 필드에 입력한 HTML 텍스트는 HTML 문서로 인식되지 않는다. HTML 형식으로 입력한 텍스트가 화면에 그대로 표시되기 때문에, HTML 텍스트는 액션스크립트에서 직접 동적 텍스트 필드에 대입해야 한다.

```
myTextField = "<P><B>Error!</B> You <I>must</I> supply an email address!</P>";
```

HTML 텍스트 필드에서 어떤 필드를 포함시키면 한 가지 폰트가 한 가지 스타일로만 추가된다. 예를 들어 Character 패널에서 Arial 굵은 글씨를 선택한다면, Arial 굵은 글씨만 지원될 뿐 일반 Arial이나 Arial 이탤릭은 지원되지 않는다. HTML을 이용하여 (이탤릭과 같은) 다른 스타일의 Arial 폰트를 선택하거나 아예 다른 폰트를 사용하면, 필요한 폰트를 무비에 포함시키지 않으면 텍스트가 화면에 표시되지 않는다.

예를 들어 output이라는 텍스트 필드를 만든다고 가정해 보자. output 텍스트 필드의 Character 패널에서 Arial 이탤릭을 선택하고 Text Options 패널에서 Arial 이탤릭 폰트 전체를 무비에 포함시키자. 그리고 나서 output 텍스트 필드에서 HTML을 출력할 수 있도록 설정하자. 마지막으로 다음과 같은 값을 텍스트 필드에 대입하자.

```
output = '<P><I>My</I>, what <B>lovely</B>'
         + '<FONT SIZE="24">eyes</FONT> you have!</P>';
```

이렇게 한 다음에 무비를 재생하면 다음과 같은 텍스트가 텍스트 필드에 출력된다.

My

output에 꽤 많은 내용을 입력했는데도 다른 건 다 빠지고 My만 출력된다. HTML에서 이탤릭으로 지정한 부분만 화면에 표시되기 때문이다. "what", "eyes", "you have"는 이탤릭이 아니고 "lovely"는 굵은 글씨이다. 따라서 무비에 포함시킨 이탤릭 폰트인 "My"만 출력된다.

HTML 텍스트 필드에서 다양한 폰트와 그 폰트에 해당하는 굵은 글씨, 이탤릭체와 같이 다양한 스타일을 사용하고 싶다면, 각 폰트, 스타일을 따로 무비에 포함시켜야 한다. 여러 폰트와 스타일을 포함시키려면 아래 두 방법 중 하나를 적절히 활용하면 된다.

- 아무 내용도 없는 텍스트 필드를 만들고 그 필드는 화면에 나타나지 않도록 한 다음, Character 패널에서 원하는 폰트를 선택하고 Text Options 패널에서 Embed fonts 옵션을 선택하여 그 폰트를 무비에 포함시킨다.
- 무비의 라이브러리에 새로운 폰트 심벌을 추가하고 그 폰트를 무비와 함께 저장한다.

텍스트 필드에서 사용할 Arial 볼드체를 무비에 포함시키고 싶다면 다음과 같이 하면 된다.

1. Window → Library 선택
2. Options → New Font 선택. Font Symbol Properties 대화상자가 화면에 뜬다.
3. Font에서 Arial을 선택한다.
4. Style에서 Bold를 선택한다.
5. Name에 ArialBold(이 이름은 라이브러리 안에서만 쓰이는 이름이다)라고 입력한다.
6. 라이브러리에서 ArialBold 폰트 심벌을 선택한다.
7. Options → Linkage를 선택한다.
8. Symbol Linkage Properties 대화상자에서 Export This Symbol을 선택한다.
9. Identifier 부분에 ArialBold라고 입력한다. 이 때 입력하는 이름은 사실 별로 중요하지 않다. 이렇게 만든 심벌 인식자는 공유 라이브러리에서만 쓰인다.

각 폰트별로 서로 다른 스타일에 해당하는 폰트를 따로따로 무비에 포함시켜야 한다는 점에 주의하자. Arial 굵은 글씨와 Arial 이탤릭과 Arial 굵은 이탤릭을 텍스트 필드에서 사용하려면 세 가지 변형된 폰트를 모두 무비에 포함시켜야 한다. 밑줄을 그었거나 색, 크기를 변경한 폰트는 변형된 폰트로 간주되지 않으므로 신경쓰지 않아도 된다.

Text Options 패널에서 Embed Fonts 옵션을 전혀 사용하지 않으면 자동으로 사용자의 시스템 폰트가 쓰인다. 이런 경우에는 사용자의 시스템에 모든 폰트와 그 폰트의 변형된 폰트가 없다면 텍스트가 화면에 제대로 표시되지 않는다.

사용자의 플랫폼이나 시스템에 상관없이 텍스트가 어디서나 똑같이 출력되도록 하려면, 텍스트 필드에서 쓰이는 모든 폰트를 무비에 포함시키도록 하자.

HTML 형식 입력

HTML은 보통 텍스트 필드를 출력하는 데 쓰이긴 하지만 HTML을 사용할 수 있는 사용자 입력 텍스트 필드를 통해서 HTML 형식 텍스트를 입력받는 경우도 있다(또는 일반적인 사용자 입력 텍스트 필드에서 HTML 형식 텍스트를 사용하는 경우도 있다).

일반적인 텍스트를 HTML을 사용할 수 있는 사용자 입력 텍스트 필드에 입력하면 HTML 태그가 자동으로 추가된다. 예를 들어 "Hi there"라는 텍스트를 입력한다면 자동으로 다음과 같은 HTML 형식으로 변환된다.

```
'<P ALIGN="LEFT"><FONT FACE="Arial" SIZE="10" COLOR="#000000">Hi
there</FONT></P>'
```

HTML을 사용할 수 있는 사용자 입력 텍스트 필드에 HTML 태그를 입력하면 < 와 > 문자는 각각 >와 <로 변환된다. 예를 들어 <B>hi there</B>라는 텍스트는 다음과 같은 값으로 변환된다.

```
'<P ALIGN="LEFT"><FONT FACE="Arial" SIZE="10"
 COLOR="#000000">&lt;B&gt;hi there&lt;/B&gt;</FONT></P>'
```

HTML을 사용할 수 있는 사용자 입력 텍스트 필드를 활용하면 간단한 HTML 데이터 입력 시스템을 만들 수도 있다.

일반 텍스트나 HTML 텍스트를 일반(HTML이 아닌) 사용자 입력 텍스트 필드에 입력하면 입력한 텍스트 내용이 그대로 대입된다. 일반 사용자 입력 텍스트 필드를 이용하면 HTML 코드를 있는 그대로 무비에 입력할 수 있다.

HTML을 사용할 수 있는 사용자 입력 텍스트 필드와 일반 사용자 입력 텍스트 필드를 다룬 예제는 온라인 코드 창고에서 구할 수 있다.

HTML 링크에서 자바스크립트 실행하기

거의 모든 자바스크립트가 돌아가는 웹 브라우저에서는 HREF 속성에 javascript: 프로토콜을 사용하면 앵커 태그에서 자바스크립트를 실행시킬 수 있다.

```
<A HREF="javascript:square(5);">find the square of 5</A>
```

액션스크립트에서도 다음과 같이 <A> 태그 안에서 자바스크립트 선언문을 실행시킬 수 있다.

```
myTextField = "<A HREF='javascript:alert(5);'>display the number 5</A>";
```

하지만 자바스크립트 선언문 안에 문자열 값을 집어넣으려면 다음과 같이 따옴표 대신 "를 사용해야 한다.

```
myTextField = "<A HREF='javascript:alert(&quot;hello world&quot;);'>"
              + "display hello world</A>";
```

HTML 링크에서 액션스크립트 함수 호출하기

플래시의 <A> 태그에서 액션스크립트 코드의 선언문을 마음대로 실행할 수는 없지만 액션스크립트 함수는 호출할 수 있다. 다음과 같이 하면 앵커 태그 안에서 액션스크립트 함수를 호출할 수 있다.

```
<A HREF="asfunction:myFunctionName">invoke the function</A>
```

함수 호출 연산자인 ()는 사용할 수 없기 때문에 앵커 태그 안에서 액션스크립트 함수를 호출할 때에는 함수 이름 뒤에 괄호를 붙일 수 없다. 앵커 태그에서 액션스크립트 함수를 호출하는 것 외에도 다음과 같이 함수에 한 개의 매개변수를 전달할 수 있다.

```
<A HREF="asfunction:myFunctionName,myParameter">invoke the function</A>
```

여기서 myParameter는 전달할 매개변수 값이다. 호출된 함수 안에서는 myParameter를 문자열로 인식한다. 앵커 태그에서 하나 이상의 정보를 함수의 매개변수로 전달하려면 myParameter 값에 구분자를 이용하여 여러 매개변수를 넣고 함수 안에서 그 문자열에 있는 여러 정보를 파싱하는 방법을 써야 한다. 예를 들어 아래의 앵커 태그에서는 두 개의 데이터를 | 문자로 구분하여 roleCall()이라는 함수에 전달한다.

```
<A HREF="asfunction:roleCall,megan|murray">invoke the function</A>
```

roleCall() 함수는 다음과 같이 만들면 된다. 이 함수에서는 split() 메소드를 이용하여 여러 개의 데이터를 분리한다.

```
function roleCall (name) {
  var bothNames = name.split("|");
  trace("first name: " + bothNames[0]);
  trace("last name: " + bothNames[1]);
}
```

텍스트 필드 선택 영역

사용자가 동적 텍스트 필드나 사용자 입력 텍스트 필드의 내용을 선택하면, 선택된 문자들의 위치가 Selection이라는 내장 객체에 저장된다. Selection 객체를 이용하면 사용자가 텍스트 필드에서 어떤 부분을 선택했는지 알아낼 수도 있고, 프로그래밍을 통해 텍스트 필드의 특정 부분을 자동으로 선택할 수도 있다. 또한 Selection 객체에서는 여러 텍스트 필드 중에 사용자가 어떤 텍스트 필드를 선택했는지도 알아낼 수 있다. Selection 객체를 이용하면 특정 텍스트 필드로 키보드 포커스를 옮겨서 사용자가 입력해야 할 부분으로 바로 커서를 옮길 수도 있다.

텍스트 필드 선택 영역의 사용법은 3부의 Selection 객체를 설명하는 부분에서 찾을 수 있다.

비어있는 텍스트 필드와 for-in 선언문

타임라인에 있는 모든 변수의 값을 조사할 때는 6장에서 나온 for-in 선언문을 이용하면 된다. 하지만 아직 정의되지 않은(즉 화면에 나와 있긴 하지만 그 값이 undefined인) 텍스트 필드는 for-in 선언문으로 직접 조사할 수 없다(비어있는 문자열("")이나 공백만 들어있는 문자열은 정의되지 않은 텍스트 필드로 간주되지 않기 때문에 for-in 선언문으로 확인할 수 있다).

오류를 체크하기 위한 스크립트에서 for-in 선언문을 사용하면 정의되지 않은 텍스트 필드를 건너뛰어 버린다는 문제점이 있다. 스크립트에서 for-in 루프를 사용하여 각 텍스트 필드를 조사할 때는 정의되지 않은 텍스트 필드도 고려해야 한다. 예를 들어 다음과 같은 코드를 이용하여 formClip이라는 무비 클립의 변수 중에 아무 내용도 없는 변수가 있는지 찾아보는 경우를 생각해 보자.

```
for (i in formClip) {
  if (formClip[i] == "") {
    trace(i + " is empty! don't submit the form!");
    break;
  }
}
```

하지만 위 코드를 그대로 실행시키면 비어있는 텍스트 필드는 조사할 수 없기 때문에, 우리가 원하는 대로 비어있는 변수를 모두 찾아낼 수는 없다. 정의되지 않은 텍스트 필드도 for-in 루프에서 확인하도록 하려면 비어있는 문자열을 텍스트 필드에 해당하는 타임라인 변수에 대입해야 한다. 다음과 같은 스크립트를 formClip의 프레임에 추가하면 앞 예제에서 비어있는 텍스트 필드도 모두 확인할 수 있다.

```
// 텍스트 필드에 비어있는 문자열을 대입하여
// for-in 루프에서 모든 텍스트 필드를 조사할 수 있도록 만든다.
formField1 = "";
formField2 = "";
```

앞으로 배울 내용

이제 이 책에서 가르쳐 주는 내용에서 벗어나 독자가 직접 프로젝트를 계획하고 새로운 아이디어를 구상해야 할 때가 다가왔다. 17장과 18장에서는 폼을 만드는 법, 스크린에 정보를 출력하는 법, 사용자로부터 데이터를 입력받는 법에 대해 배웠다. '3부. 레퍼런스'를 시작하기 전에 19장에서 코드에 있는 버그를 잡는 방법을 배워보자. 디버깅은 앞으로 프로그래밍을 하다 보면 반드시 거쳐야만 하는 작업이다.