7

조건문

지금까지 본 코드 예제는 매우 선형적(linear)이다. 즉 모든 선언문이 맨 앞에 있는 것부터 맨 뒤에 있는 것까지 순서대로 실행된다. 선형 코드를 사용하면 결과가 언제나 똑같다. 이와는 대조적으로 조건문은 주어진 조건이 성립될 때만 어떤 작업을 처리하는 유형의 선언문이다. 선형 코드에서는 인터프리터에서 선언문 A, 선언문 B, 선언문 C를 순서대로 실행한다. 조건문을 사용하면 인터프리터에서 선언문 A를 실행하고 나서 조건 X에 따라 선언문 B 또는 선언문 C 중 하나를 실행하도록 만들 수 있다.

조건문을 이용하면 두 개 이상의 결과가 나올 수 있는 상황을 만들고 제어할 수 있다. 예를 들어 비밀번호로 보호된 사이트를 만드는 경우를 생각해 보자. 사용자가 로그인했을 때 비밀번호가 맞는 경우에는 사용자를 사이트로 들여보내고, 그렇지 않으면 사용자에게 오류 메시지를 내보낸다. 이와 같이 두 가지 결과를 만들려면 무비의 스크립트에 서로 다른 코드 블록을 집어넣어야 한다. 한 블록에서는 플래시 플레이헤드를 사이트 시작 화면으로 보내야 하는 반면, 다른 한 블록에서는 플레이헤드를 오류 메시지를 출력하는 프레임으로 보내야 한다. 하지만 사용자가 로그인할 때는 둘 중 한 코드 블록만 실행된다. 조건문을 이용하면 상황에 따라 적절한 블록을 실행하고 나머지 블록은 건너뛰도록 만들 수 있다.

인터프리터에서는 실행할 코드 블록을 어떻게 결정할까? 조건문을 정의할 때는 첫째 코드 블록을 실행시키기 위해 필요한 조건을 지정한다. 이 조건이 만족되지 않으면 다른 코드 블록이 실행된다(또한 그 코드 블록에도 별도의 조건이 있을 수 있다). 조건문을 사용할 때는 결국 아래에 유사코드로 표시한 것과 비슷한 구조를 가진 플로우차트와 유사한 논리구조를 만들어야 한다.

```
if (첫 번째 조건) {
  // 이 코드를 실행한다.
} else if (두 번째 조건) {
  // 이 코드를 실행한다.
} ...그렇지 않으면 {
  // Execute this code
}
```

물론 각 조건은 인터프리터에서 이해할 수 있는 방법으로 기술해야 한다. 하지만 너무 어렵게 생각할 필요는 없다(지금부터 배우는 내용만 제대로 이해하면 된다). 개념적으로 볼 때 모든 조건문은 주어진 조건을 바탕으로 코드 블록을 실행할지 아니면 실행하지 않을지를 결정하는 역할을 한다.

if 선언문

if 선언문은 가장 일상적인 만능 조건문이다. if 선언문은 갈림길과 비슷하게 코드의 진행 방향을 두 갈래로 나누기 위해 사용한다. if 선언문에는 한 개 이상의 하위 선언문이 있어서 주어진 조건이 성립되면 그 선언문을 실행하도록 되어 있다. if 선언문의 사용법은 다음과 같다.

```
if (condition) {
  substatements
}
```

모든 if 선언문은 if라는 키워드로 시작한다. 괄호 안에 있는 주어진 조건이 만족되어야 하위선언문이 실행된다. substatements는 하나 이상의 액션스크립트 선언문이다. 각 하위 선언문은 서로 다른 줄에 있어야 하며 세미콜론으로 끝나야 한다. if 선언문 전체는 오른쪽 중괄호(})로 끝나며 그 뒤에는 세미콜론이 붙지 않는다.

if 선언문의 condition 부분에는 제대로 된 표현식이라면 어떤 표현식이라도 사용할 수 있다. if 선언문을 실행하면 인터프리터에서는 주어진 표현식(기준 표현식, test expression)의 값을 확인한다. 기준 표현식 값이 true이면 substatements가 실행되고 false이면 실행되지 않는다. 아래 예에서는 단순한 부울 값을 기준 표현식으로 사용한다.

```
if (true) {
  trace("The condition was met!");  // 이 선언문은 실행된다.
}
if (false) {
  trace("The condition was met!");  // 이 선언문은 절대 실행되지 않는다.
}
```

물론 부울 리터럴은 값이 고정되어 있기 때문에 실제로 기준 표현식에 부울 리터럴을 사용하는 경우는 없다. 대신 부울 값을 리턴하는 복합 표현식을 사용하게 된다. 예를 들어 비교 연산과 관련된 표현식은 부울 값을 리턴하므로 조건문의 기준 표현식으로 쓰기에 적합하다.

```
var pointerX = _xmouse;  // 마우스의 수평 위치

// pointerX > 300 이 true이면
if (pointerX > 300) {
  // 이 선언문을 실행한다.
  trace("The mouse is past the 300 pixel mark");
}
```

이제 조금 더 재미있는 부분을 살펴보자. 조건문의 기준 표현식이 반드시 부울 값이어야만 하는 것은 아니므로 아무 표현식이나 그 자리에 들어갈 수 있다. 다음과 같이 문자열이나 숫자를 조건문의 기준 표현식으로 사용할 수도 있다.

```
if ("hi") {
  trace("The condition was met!");
}
if (4) {
  trace("The condition was met!");
}
```

"hi"와 4는 부울형이 아닌데 어떻게 위와 같은 코드가 동작할 수 있을까? 그 해답은 〔표 3-3〕에 나온 데이터형 변환에서 찾을 수 있다. 조건문의 기준 표현식이 부울 값이 아닌 경우에는 인터프리터에서 그 표현식을 부울형으로 변환한다. 예를 들어 인터프리터에서는 "hi"를 false로 변환한다. 숫자가 아닌 문자열은 부울 값으로 따지면 false가 되기 때문이다. 따라서 이 조건은 성립되지 않기 때문에 위에 있는 첫째 trace() 선언문은 실행되지 않는다. 또한 인터프리터에서는 숫자 4를 true로 변환하므로(0이 아닌 모든 숫자는 true가 된다) 둘째 trace() 선언문은 실행된다.

드디어 데이터형 변환에 대해 열심히 배운 효과가 나타나고 있다. 아래 예는 이를 응용한 몇 가지 예제이다. 각 선언문의 실행 여부를 맞춰보자.

```
x = 3;
if (x) {
  trace("x is not zero");
}
```

아래 예제에서는 '5장. 연산자'에서 배운 OR 연산자를 이용한다.

```
lastName = "";
firstName = "";
if (firstName != "" || lastName != "") {
  trace("Welcome" + firstName + " " + lastName);
}
```

마지막으로 무비 클립 객체가 존재하는지 확인하는 예제이다.

```
if (myClip) {
  myClip._x = 0;  // myClip이 존재하면 그 클립을
                  // 스테이지의 왼쪽 끝에 붙인다.
}
```

else 선언문

if 선언문만 사용해도 조건에 따라 하나의 코드 블록을 실행시키는 것을 처리할 수 있다. 여기에 else 절을 사용하면 두 개의 코드 블록 중 어떤 것을 실행할지 결정할 수 있다. else 선언문은 다음과 같이 if 선언문을 확장시킨 형태로 쓰인다.

```
if (condition) {
  substatements1
} else {
  substatements2
}
```

제대로 된 표현식이라면 모두 condition 자리에 쓰일 수 있다. condition이 true이면 statements1이, false이면 statements2가 실행된다. 즉 else 선언문은 서로 배타적인 결정을 내려야 할 때 쓰인다. 둘 중 한 코드 블록만 실행되기 때문이다.

아래 코드를 살펴보자.

```
var lastName = "Grossman";
var gender = "male";
if (gender == "male") {
  trace("Good morning, Mr." + lastName + ".");
} else {
  trace("Good morning, Ms." + lastName + ".");
}
```

else 절은 종종 if 선언문을 뒷받침하기 위한 용도로 쓰인다. 비밀번호를 입력해야 들어갈 수 있는 웹사이트 예제를 다시 떠올려보자. 비밀번호가 맞으면 사용자가 사이트에 들어갈 수 있지만 그렇지 않으면 오류 메시지만 화면에 나타나게 된다. 패스워드를 조사하는 코드는 다음과 같은 형태로 쓸 수 있다(userName과 password는 사용자가 입력한 정보이고 validUser와 correctPassword는 사이트에 들어갈 수 있는 로그인 값이라고 가정하자).

```
if (userName == validUser && password == correctPassword) {
  gotoAndPlay("intro");
} else {
  gotoAndStop("loginError");
}
```

조건 연산자

두 부분으로 이루어지는 간단한 조건문은 조건 연산자(?:)를 이용하면 간편하게 표현할 수 있다. 조건 연산자에는 세 개의 피연산자가 필요하며, if-else 선언문의 각 요소와 거의 같다고 보면 된다.

```
condition ? expression1 : expression2;
```

조건 연산자에서는 condition이 true이면 expression1을 계산하여 그 값을 리턴한다. 그렇지 않으면 expression2를 계산하여 그 결과를 리턴한다. 조건 연산자는 다음과 같은 조건문을 간략하게 표현한 것이라고 볼 수 있다.

```
if (condition) {
  expression1
} else {
  expression2
}
```

조건문과 마찬가지로 조건 연산자를 이용하면 프로그램의 흐름을 제어할 수 있다.

```
// command가 "go"이면 무비를 재생하고 그렇지 않으면 멈춘다.
command == "go" ? play() : stop();
```

조건 연산자를 이용하면 변수에 값을 대입하는 것도 간단하게 표현할 수 있다.

```
var guess = "c";
var answer = "d";
var response = (guess == answer) ? "Right": "Wrong";
```

위 코드는 아래 코드와 똑같은 기능을 한다.

```
var guess = "c";
var answer = "d";
if  (guess == answer) {
  response = "Right";
} else {
  response = "Wrong";
}
```

else if 선언문

if와 else를 이용하면 조건에 따라 두 개의 코드 블록 중 하나를 실행할 수 있다. if와 else if를 사용하면 무한히 많은 코드 블록 중 하나를 실행시킬 수 있다(또는 그 중 아무것도 실행시키지 않을 수도 있다). else와 마찬가지로 else if는 다음과 같이 if 선언문을 확장한 형태로 쓰인다.

```
if (condition1) {
  substatements1
} else if (condition2) {
  substatements2
} else if (condition3) {
  substatements3
} else {
  substatements4  // 위 조건이 모두 맞지 않을 경우에 사용할 선언문
}
```

condition1, condition2, condition3 자리에는 임의의 유효한 표현식을 사용할 수 있다. condition1이 true이면 substatements1이 실행되고, condition1이 false이고 condition2가 true이면 substatements2가 실행된다. 그렇지 않으면 codition3을 확인한다. 이런 식으로 여러 개의 else if 문을 이용하여 무한히 많은 조건을 검사할 수 있다. 기준 표현식 중 어느 것도 true가 아니면 마지막에 있는 else 절에 있는 선언문이 실행된다. 예를 들어 로그인 검사 루틴을 다음과 같이 만들어서 다양한 오류 메시지를 출력할 수 있다.

```
if (userName != validUser) {
  message = "User not found. Please try again.";
  gotoAndStop("loginError");
} else if (password != correctPassword) {
  message = "Password incorrect. Please try again.";
  gotoAndStop("loginError");
} else {
  gotoAndPlay("intro");
}
```

else if 선언문은 사실 else를 여러 개의 if 선언문과 결합시킨 것에 지나지 않는다. 아래에 있는 두 코드는 결국 같은 기능을 하지만 첫째 코드가 훨씬 이해하기 쉽다.

```
// 일반적인 "else if" 구문
if (x > y) {
  trace("x is larger than y");
} else if (x < y) {
  trace("x is smaller than y");
} else {
  trace("x and y are equal");
}

// if/else를 풀어서 써 놓은 것
if (x > y) {
  trace("x is larger than y");
} else {
  if (x < y) {
    trace("x is smaller than y");
  } else {
    trace("x and y are equal");
  }
}
```

switch 선언문 흉내내기

액션스크립트에서는 switch 선언문(또는 case 선언문이라고도 부른다)을 지원하지 않지만 다른 방법으로 비슷한 효과를 낼 수는 있다. switch 선언문을 이용하면 하나의 기준 표현식 값을 기준으로 여러 개의 코드 블록 중 하나만을 실행할 수 있다. 예를 들면 아래에 나온 자바스크립트의 switch 선언문에서는 기준 표현식인 gender 값에 따라 각각 다른 인사말을 출력한다.

```
var surname = "Porter";
var gender = "male";

switch (gender) {
  case "femaleMarried":          // 기혼 여성의 경우
    alert("Hello Mrs. "+ surname);
    break;
  case "femaleGeneric":          // 기혼/미혼을 따지지 않는 경우
    alert("Hello Ms. "+ surname);
    break;
```

```
    case "male":                    // 남성인 경우
      alert("Hello Mr. "+ surname);
      break;
    default :
      alert("Hello "+ surname);
  }
```

위 자바스크립트 예제에서는 switch 선언문에서 gender 값을 각 case 표현식에 있는 "femealMarried", "femaleGeneric", "male"과 비교한다. 여기서는 gender가 "male"이기 때문에 alert("Hello Mr." + surname);이 실행된다. 기준 표현식이 어떤 조건과도 맞지 않는 경우에는 기본 선언문인 alert("Hello" + surname);이 실행된다.

액션스크립트에서 if-else if-else 선언문을 연결해서 사용하면 switch 선언문과 같은 효과를 연출할 수 있다.

```
var surname = "Porter";
var gender = "male";

if (gender == "femaleMarried") {
  trace("Hello Mrs. "+ surname);
} else if (gender == "femaleGeneric") {
  trace(" Hello Ms." + surname);
} else if (gender == "male") {
  trace("Hello Mr."+ surname);
} else {
  trace("Hello" + surname);
}
```

조금 더 고급 기법을 사용한다면 일반적인 객체 속성에 저장된 일련의 함수로 switch를 대신할 수 있다. 이러한 방법은 〔예제 7-1〕에 나와 있다. 주석을 주의 깊게 읽어보고 이 코드가 어떻게 작동하는지 이해하도록 하자. 그리고 조금 전에 배운 조건 연산자를 사용하는 방법도 자세히 살펴보자.

〔예제 7-1〕 switch 선언문 시뮬레이션

```
var surname = "Porter"; // 사용자의 이름
var gender = "male";    // 사용자의 성(기준 표현식으로 쓰임)

// switch 선언문을 대신할 객체를 만든다.
```

```
var mySwitch = new Object();

// mySwitch 객체에 "case expression" 속성을 만든다.
// 각 "case expression" 속성에는 함수가 들어있다.
mySwitch.femaleMarried = function() {
  trace("Hello Mrs." + surname);
};
mySwitch.femaleGeneric = function() {
  trace("Hello Ms." + surname);
};
mySwitch.male = function() {
  trace("Hello Mr." + surname);
};
mySwitch.default = function() {
  trace("Hello" + surname);
};

// gender의 값에 따라(이 예제에서는 "male") 적당한 함수를 실행한다.
// 그러한 이름을 가진 속성이 없으면 기본 함수를 실행한다.
mySwitch[gender] ? mySwitch[gender]() : mySwitch["default"]();
```

간결한 조건문

이 책에서는 선언문이 한 줄짜리인 경우에도 모든 조건문에서 선언문 블록(중괄호로 둘러싸인 선언문)을 사용했다.

```
if (x == y) {
  trace("x and y are equal");
}
```

액션스크립트에서는 조건문에 하위 선언문이 하나뿐인 경우에는 중괄호를 사용하지 않아도 된다. 하위 선언문이 하나뿐인 경우에는 if나 else if 바로 뒤에 중괄호 없이 선언문을 적어도 된다.

```
if (x == y) trace ("x and y are equal");
```

또는 다음과 같이 해도 된다.

```
if (x == y)
  trace ("x and y are equal");
```

프로그래머에 따라 이런 스타일을 사용하면 가독성이 떨어지고 프로그래밍 도중에 실수할 가능성이 많다고 하는 사람들도 있지만, 이렇게 하면 소스 코드의 크기를 줄이는 데 도움이 된다.

앞으로 배울 내용

조건문은 액션스크립트에서 매우 중요한 구성요소이다. 조건문을 이용하면 사용자가 코드의 실행 조건을 제어할 수 있다. 다음 장에서는 또 다른 중요한 실행 제어 선언문인 순환문에 대해 알아보도록 하자.