

이벤트

이벤트에 대한 개괄적인 내용은 자바스크립트의 이벤트 부분을 공부하며 배웠습니다. jQuery의 이벤트는 기존 자바스크립트의 이벤트가 모두 존재합니다. 하지만 jQuery를 사용하면 기존 자바스크립트의 이벤트를 연결할 때보다 훨씬 간편하게 이벤트를 연결할 수 있습니다.

이 장에서는 jQuery에서 이벤트를 어떻게 다루는지 공부합니다. 지금까지 jQuery를 이용할 때 무조건 다음 구문을 사용했습니다.

```
$(document).ready(function (event) {  
});
```

이 구문은 document 객체에 ready 이벤트를 연결하는 것입니다. 이 장을 진행하기 전에 그림 16-1처럼 보조 기능으로 어떠한 메서드가 이벤트와 관련되는지 예측해보세요.

그림 16-1 jQuery의 이벤트 메서드

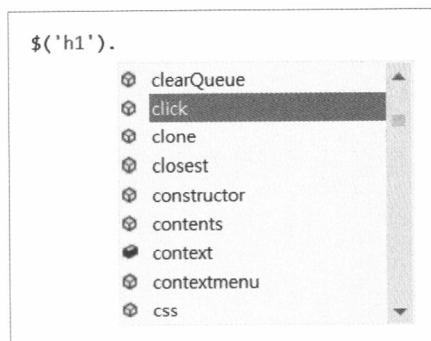


그림 16-1을 보면 blur(), change(), click() 메서드 등 이벤트 이름을 확인할 수 있죠?
직접 보고 공부하는 것과 그렇지 않은 것과는 상당한 차이가 있으므로 꼭 확인해보세요.

16.1 이벤트 연결 기본

jQuery로 이벤트를 연결하는 가장 기본적인 방법은 on() 메서드를 사용하는 것입니다.

표 16-1 이벤트 연결 메서드

메서드 이름	설명
on()	이벤트를 연결합니다.

on() 메서드는 다음 두 가지 형태로 사용할 수 있습니다.

- 1 \$(selector).on(eventName, function(event) { })
- 2 \$(selector).on(object)

예제를 작성하며 살펴보겠습니다. body 태그를 코드 16-1처럼 구성해주세요.

코드 16-1 body 태그 구성

```
<body>
  <h1>Header-0</h1>
  <h1>Header-1</h1>
  <h1>Header-2</h1>
</body>
```

on() 메서드의 1번 형태를 살펴봅시다. on() 메서드의 첫 번째 매개변수에 이벤트 이름을 입력하고 두 번째 매개변수에 이벤트 리스너를 입력합니다. 이벤트 리스너 안에서 this 키워드는 이벤트가 발생한 객체를 의미합니다.

코드 16-2는 h1 태그를 click 이벤트에 연결하고 이벤트 발생 시 이벤트 발생 객체에 '+' 글자를 추가합니다.

코드 16-2 on() 메서드(1)

```
<script>
$(document).ready(function () {
    // 이벤트를 연결합니다.
    $('h1').on('click', function () {
        $(this).html(function (index, html) {
            return html + '+';
        });
    });
}
</script>
```

2번 형태도 이용해봅시다. 코드 16-3처럼 on() 메서드의 매개변수에 객체를 넣어줍니다. 속성 이름과 속성 값에 이벤트 이름과 이벤트 리스너를 넣으면 이벤트를 쉽게 연결할 수 있습니다.

코드 16-3 on() 메서드(2)

```
<script>
$(document).ready(function () {
    // 이벤트를 연결합니다.
    $('h1').on('click', function () {
        $(this).html(function (index, html) {
            return html + '+';
        });
    });

    // 이벤트를 연결합니다.
    $('h1').on({
        mouseenter: function () { $(this).addClass('reverse'); },
        mouseleave: function () { $(this).removeClass('reverse'); }
    });
}
</script>
```

코드 16-3은 마우스가 진입하면 reverse 클래스 속성을 추가하고, 마우스가 빠져나가면 reverse 클래스 속성을 제거합니다.

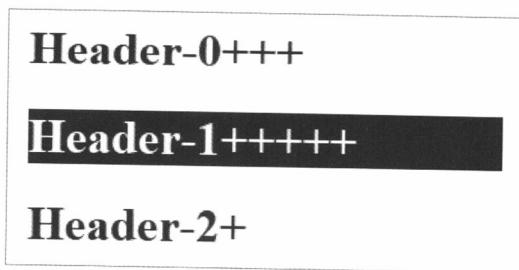
아직 클래스 속성을 생성하지 않았는데 사용했죠? 코드 16-4처럼 style 태그를 입력해주세요.

코드 16-4 style 태그 구성

```
<style>
.reverse {
    background: black;
    color: white;
}
</style>
```

코드를 실행하고 각각의 이벤트를 발생시키면 그림 16-2처럼 출력 결과를 얻을 수 있습니다.
jQuery를 사용하면 이벤트를 연결하는 방법이 엄청나게 간단합니다.

그림 16-2 이벤트 연결



16.2 간단한 이벤트 연결

jQuery는 많이 사용하는 이벤트를 간단한 방식으로 연결할 수 있게 했습니다. 지금까지 사용했던 ready() 메서드도 간단한 형태의 이벤트라고 할 수 있습니다. 간단한 방식으로 연결할 수 있는 이벤트는 표 16-2와 같습니다.

표 16-2 간단한 이벤트 연결

blur	focus	focusin	focusout	load
resize	scroll	unload	click	dblclick
mousedown	mouseup	mousemove	mouseover	mouseout
mouseenter	mouseleave	change	select	submit
keydown	keypress	keyup	error	ready

간단한 방식으로 이벤트를 연결할 때는 다음 방법을 사용합니다.

```
$(selector).method(function(event) { });
```

지금까지 여러 번 사용해봤으므로 별도로 설명하지 않겠습니다.

이렇게 간단한 형태의 이벤트 연결 방법에서 한 단계 더 나아가 jQuery는 표 16-3의 이벤트 연결 메서드도 제공해줍니다.

표 16-3 hover() 메서드

메서드 이름	설명
hover()	mouseenter 이벤트와 mouseleave 이벤트를 동시에 연결합니다.

표 16-3의 메서드는 다음 형태로 사용합니다.

```
$(selector).hover(function(event) { }, function(event) { });
```

HTML 페이지를 다음과 같이 구성해주세요.

코드 16-5 HTML 페이지 구성

```
<!DOCTYPE html>
<html>
<head>
<style>
.reverse {
    background: black;
    color: white;
}
</style>
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script>
$(document).ready(function () {

});

</script>
</head>
<body>
```

```
<h1>Header-0</h1>
<h1>Header-1</h1>
<h1>Header-2</h1>
</body>
</html>
```

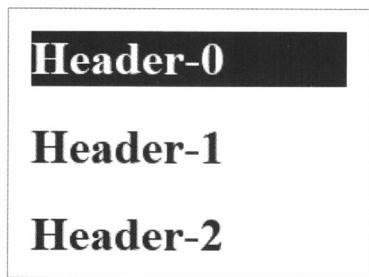
hover() 메서드는 코드 16-6처럼 두 개의 함수를 매개변수로 넣어줍니다. 첫 번째 매개변수에 넣는 함수는 mouseenter 이벤트 리스너이며 두 번째 매개변수에 넣는 함수는 mouseleave 이벤트 리스너입니다. 따라서 코드 16-6은 이전 절에서 살펴본 기능과 같은 기능을 수행합니다.

코드 16-6 hover() 메서드

```
<script>
$(document).ready(function () {
    // 이벤트를 연결합니다.
    $('h1').hover(function () {
        $(this).addClass('reverse');
    }, function () {
        $(this).removeClass('reverse');
    });
});
</script>
```

코드를 실행하고 h1 태그 위에 마우스를 올려보세요.

그림 16-3 hover() 메서드



사실 on() 메서드보다 간단한 이벤트 연결 방법을 많이 사용합니다. on() 메서드는 한 번에 여러 종류의 이벤트를 연결하거나, 간단한 이벤트 연결 방법으로 연결할 수 없을 때 사용합니다.

“간단한 이벤트 연결 방법으로 연결할 수 없는 이벤트도 있나요?”

개발자가 직접 만든 이벤트나 터치 이벤트는 간단한 이벤트 연결 방법으로는 연결할 수 없습니다. 이러한 경우에는 무조건 `on()` 메서드를 사용해야 합니다.

16.3 이벤트 연결 제거

이벤트를 제거할 때는 `off()` 메서드를 사용합니다. `off()` 메서드의 형태는 다음과 같습니다.

표 16-4 이벤트 연결 제거 메서드

메서드 이름	설명
<code>off()</code>	이벤트를 제거합니다.

`off()` 메서드는 다음 형태로 사용합니다.

- 1 `$(selector).off()`
- 2 `$(selector).off(eventName)`
- 3 `$(selector).off(eventName, function)`

1번 형태는 해당 문서 객체와 관련된 모든 이벤트를 제거합니다. 2번 형태는 해당 문서 객체의 특정 이벤트와 관련된 모든 이벤트를 제거하며, 3번 형태는 특정 이벤트 리스너를 제거합니다. 예제를 만들어봅시다. `on()` 메서드와 `off()` 메서드를 사용하면 이벤트를 한 번만 실행하는 예제를 쉽게 작성할 수 있습니다. 코드 16-7처럼 `body` 태그를 구성해주세요.

코드 16-7 `body` 태그 구성

```
<body>
  <h1>Header-0</h1>
  <h1>Header-1</h1>
  <h1>Header-2</h1>
</body>
```

코드 16-8처럼 `h1` 태그에 `click` 이벤트를 연결하고 이벤트 발생을 알립니다. 이후에 `off()` 메서드로 이벤트를 제거하면 이벤트를 한 번만 실행하겠죠?

코드 16-8 off() 메서드

```
<script>
$(document).ready(function () {
    // 이벤트를 연결합니다.
    $('h1').click(function () {
        // 출력합니다.
        $(this).html('CLICK');
        alert('이벤트가 발생했습니다.');

        // 이벤트를 제거합니다.
        $(this).off();
    });
});
</script>
```

코드를 실행하고 각 h1 태그를 선택해보세요. 이벤트를 한 번씩만 실행할 것입니다. 사실 이러한 기능을 제공하는 jQuery 메서드가 별도로 존재합니다. 표 16-5의 one() 메서드입니다.

표 16-5 one() 메서드

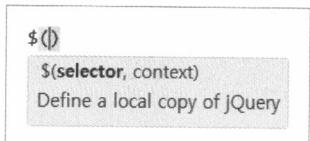
메서드 이름	설명
one()	이벤트를 한 번만 연결합니다.

one() 메서드의 매개변수는 on()과 같으므로 따로 언급하지 않겠습니다. one() 메서드를 직접 사용해보세요.

16.4 매개변수 context

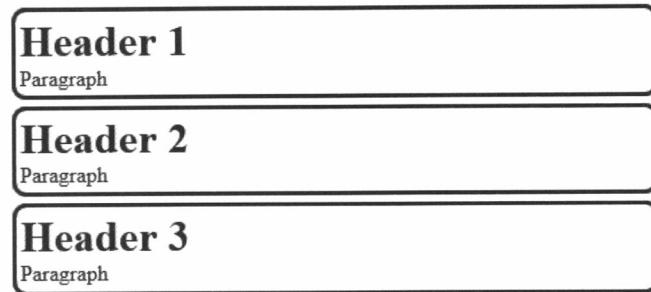
jQuery 메서드는 사실 매개변수를 두 개 입력할 수도 있습니다. 그림 16-4를 살펴보면 알 수 있지만, 지금까지 사용한 selector 매개변수 이외에 context 매개변수도 넣을 수 있습니다.

그림 16-4 매개변수 context



매개변수 context는 selector가 적용하는 범위를 한정합니다. 일반적으로 이벤트와 함께 사용합니다. 예를 들어 그림 16-5처럼 HTML 페이지가 있습니다. div 태그를 클릭했을 때, 클릭한 div 태그 내부의 h1 태그와 p 태그가 가지는 내용을 출력하려면 어떻게 해야 할까요?

그림 16-5 문제



지금까지 배운 방식으로는 이를 구현할 수 없습니다. \$('h1')과 같은 방식으로 선택했다가는, 페이지 전체의 h1 태그를 선택할 테니까요. 이렇게 특정 부분에 선택자를 적용하고 싶을 때 사용하는 것이 매개변수 context입니다.

예제를 만들면서 살펴봅시다. body 태그를 코드 16-9처럼 구성합니다.

코드 16-9 body 태그 구성

```
<body>
  <div>
    <h1>Header 1</h1>
    <p>Paragraph</p>
  </div>
  <div>
    <h1>Header 2</h1>
    <p>Paragraph</p>
  </div>
  <div>
    <h1>Header 3</h1>
    <p>Paragraph</p>
  </div>
</body>
```

style 태그를 코드 16-10처럼 구성합니다.

코드 16-10 style 태그 구성

```
<style>
* { margin: 0px; padding: 0px }

div
{
    margin: 5px; padding: 3px;
    border: 3px solid black;
    border-radius: 10px;
}
</style>
```

click 이벤트가 발생한 문서 객체 안에서 h1 태그와 p 태그를 선택하고 싶을 때는 코드 16-11처럼 this 키워드를 \$() 메서드의 두 번째 매개변수로 넣어줍니다. 범위를 이벤트 발생 객체로 한정하므로 쉽게 이벤트 발생 객체 안에만 선택자를 적용할 수 있습니다.

코드 16-11 context 객체

```
<script>
$(document).ready(function () {
    // 이벤트를 연결합니다.
    $('div').click(function () {
        // 변수를 선언합니다.
        var header = $('h1', this).text();
        var paragraph = $('p', this).text();

        // 출력합니다.
        alert(header + '\n' + paragraph);
    });
});
</script>
```

코드를 실행하고 각각의 div 태그를 클릭해보세요. 이벤트 발생 객체의 h1 태그와 p 태그에 있는 내용을 출력할 것입니다. jQuery 플러그인을 만들 때도 자주 사용하는 내용이므로 꼭 기억해주세요.

NOTE find() 메서드 활용

다음과 같이 find() 메서드를 사용해도 됩니다.

코드 16-12 find() 메서드 활용

```
<script>
$(document).ready(function () {
    // 이벤트를 연결합니다.
    $('div').click(function () {
        // 변수를 선언합니다.
        var header = $(this).find('h1').text();
        var paragraph = $(this).find('p').text();

        // 출력합니다.
        alert(header + '\n' + paragraph);
    });
});
</script>
```

16.5 이벤트 객체

모든 이벤트 리스너는 이벤트 객체가 있습니다. 1부에서 기존 자바스크립트의 이벤트를 다루면서 이벤트 객체를 살펴보았습니다. 그때의 이벤트 객체와 이 절에서 배울 이벤트 객체는 거의 비슷합니다. 주의할 점은 “같지는 않다”라는 점입니다.

기존 자바스크립트의 이벤트 객체는 브라우저마다 사용 방법은 물론 이벤트 객체에 있는 속성 조차 달랐습니다. 하지만, jQuery가 스스로 이벤트 객체를 정형화하므로 jQuery의 이벤트 객체는 모든 브라우저가 같은 방법으로 사용하고 같은 속성을 갖습니다. 표 16-6은 자주 사용하는 jQuery 이벤트 객체입니다. 추가 이벤트 객체의 속성은 jQuery 공식 홈페이지를 참고하세요.

표 16-6 이벤트 객체의 속성

이벤트 객체 속성	설명
event.pageX	브라우저의 화면을 기준으로 한 마우스의 X 좌표 위치
event.pageY	브라우저의 화면을 기준으로 한 마우스의 Y 좌표 위치
event.preventDefault()	기본 이벤트를 제거합니다.
event.stopPropagation()	이벤트 전달을 제거합니다.

이벤트 객체와 관련된 기본적인 내용은 1부에서 배웠으므로 이 절에서는 이벤트 객체를 활용하는 예제를 작성해봅시다. HTML5 Canvas를 사용하는 예제를 작성할 것인데요. 잡다한 메서드에 주목하기보다는 jQuery 이벤트 객체와 관련된 내용에 주목해주세요.

우선 body 태그를 코드 16-13처럼 구성합니다. canvas 태그에 id 속성을 입력하고 width, height, style 속성을 지정합니다.

코드 16-13 body 태그 구성

```
<body>
  <canvas id="canvas" width="700" height="400" style="border: 3px solid black">
    </canvas>
  </body>
```

canvas 태그는 HTML5부터 지원하는 태그이므로 인터넷 익스플로러 8에서는 지원하지 않습니다.

NOTE 인터넷 익스플로러 8 이하에서도 excanvas.js 파일을 추가하면 canvas 태그를 사용할 수 있습니다. 자세한 사항은 <http://excanvas.sourceforge.net/>을 참고하세요.

아직 jQuery가 canvas 태그를 정식으로 지원하지 못합니다. 따라서 코드 16-14처럼 document 객체의 getElementById() 메서드를 사용해야 하므로 canvas 태그에 id 속성을 부여한 것입니다. jQuery로 모든 것을 할 수 있는 것은 아니라는 사실을 기억해주세요. canvas 객체에서 context를 추출하고 이벤트를 연결합니다. canvas 객체와 관련된 내용은 넘어갑니다.

코드 16-14 2d context 객체 추출 및 이벤트 연결

```
<script>
$(document).ready(function () {
    // 변수를 선언합니다.
    var canvas = document.getElementById('canvas');
    var context = canvas.getContext('2d');

    // 이벤트를 연결합니다.
    $(canvas).on({
        mousedown: function (event) { },
        mouseup: function (event) { }
    });
});
</script>
```

선을 그리려면 선의 시작 위치와 끝 위치를 구해야 합니다. 코드 16-15는 문서 객체와 마우스 커서의 상대 위치를 구하는 방법입니다. 자주 사용하는 방법이므로 기억해주세요. 코드 16-15에서 주목할 점은 이벤트 객체에서 pageX 속성과 pageY 속성을 사용했다는 것입니다.

코드 16-15 이벤트 객체 활용

```
// 이벤트를 연결합니다.
$(canvas).on({
    mousedown: function (event) {
        // 위치를 얻어냅니다.
        var position = $(this).offset();
        var x = event.pageX - position.left;
        var y = event.pageY - position.top;
    },
    mouseup: function (event) {
        // 위치를 얻어냅니다.
        var position = $(this).offset();
        var x = event.pageX - position.left;
        var y = event.pageY - position.top;
    }
});
```

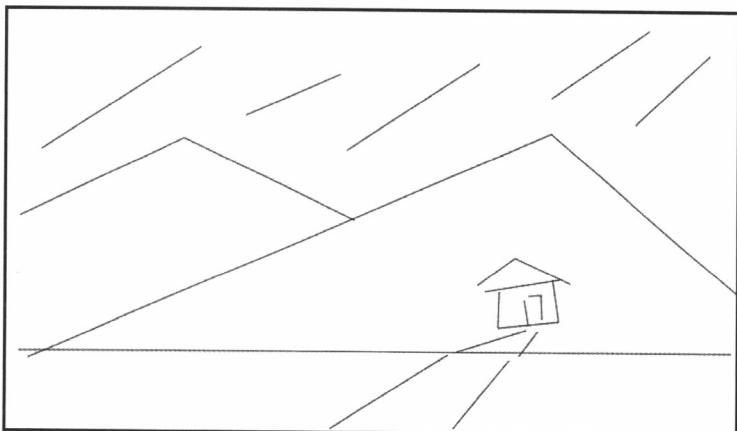
context 객체의 beginPath() 메서드와 moveTo(), lineTo(), stroke() 메서드를 사용하면 canvas 객체에 선을 그릴 수 있습니다.

코드 16-16 그림 그리기

```
// 이벤트를 연결합니다.  
$(canvas).on({  
    mousedown: function (event) {  
        // 위치를 얻어냅니다.  
        var position = $(this).offset();  
        var x = event.pageX - position.left;  
        var y = event.pageY - position.top;  
  
        // 선을 그립니다.  
        context.beginPath();  
        context.moveTo(x, y);  
    },  
    mouseup: function (event) {  
        // 위치를 얻어냅니다.  
        var position = $(this).offset();  
        var x = event.pageX - position.left;  
        var y = event.pageY - position.top;  
  
        // 선을 그립니다.  
        context.lineTo(x, y);  
        context.stroke();  
    }  
});
```

코드를 실행하면 canvas 태그 위에 그림 16-6처럼 마우스로 그림을 그릴 수 있습니다. 지금 까지 너무 간단하고 재미없는 예제만 살펴본 것 같아 잠시 색다른 예제를 작성해보았습니다.

그림 16-6 그림 그리기



NOTE HTML5 Canvas와 관련된 내용은 이 책에서 다루지 않습니다. 인터넷이나 별도의 책을 참고하세요.

16.6 이벤트 강제 발생

jQuery에서 이벤트를 강제로 발생시킬 때는 표 16-7의 메서드를 사용합니다.

표 16-7 이벤트 강제 발생 메서드

메서드 이름	설명
trigger()	이벤트를 강제로 발생시킵니다.

trigger() 메서드는 다음과 같은 형태로 사용할 수 있습니다. 2번 형태의 매개변수 data는 일반적으로 배열을 집어넣습니다.

- 1 \$(selector).trigger(eventName)
- 2 \$(selector).trigger(eventName, data)

1번 형태를 예제를 만들며 살펴봅시다. 코드 16-17처럼 body 태그를 구성해주세요.

코드 16-17 body 태그 구성

```
<body>
  <h1>Start: </h1>
  <h1>Start: </h1>
</body>
```

h1 태그에 click 이벤트를 연결합니다. 클릭하면 뒤에 별이 추가됩니다. 이어서 setInterval() 함수로 마지막에 위치한 h1 태그는 자동으로 1초마다 클릭 이벤트를 실행하게 했습니다.

코드 16-18 trigger() 메서드

```
<script>
$(document).ready(function () {
```

```
// 이벤트를 연결합니다.  
$('h1').click(function () {  
    $(this).html(function (index, html) {  
        return html + '★';  
    });  
});  
  
// 1초마다 함수를 실행합니다.  
setInterval(function () {  
    $('h1').last().trigger('click');  
, 1000);  
});  
</script>
```

코드를 실행하면 아래에 있는 h1 태그는 1초마다 별이 자동으로 생성됩니다. 물론 두 h1 태그 모두 직접 클릭해서 별을 추가할 수 있습니다.

그림 16-7 trigger() 메서드



trigger() 메서드로 이벤트를 강제로 발생시킬 수도 있지만, 간단한 이벤트 연결 방식을 사용할 수 있는 이벤트는 코드 16-19처럼 이벤트를 강제로 실행할 수 있습니다.

코드 16-19 간단한 이벤트 강제 실행

```
// 1초마다 함수를 실행합니다.  
setInterval(function () {  
    $('h1').last().click();  
, 1000);
```

간단한 이벤트 연결 방식을 사용할 수 없는 경우도 있으므로 trigger() 메서드를 꼭 기억해주세요. 또한 trigger() 메서드의 2번 형태를 사용하면 추가 기능을 수행할 수 있습니다. trigger() 메서드의 2번 형태는 자주 쓰이지 않지만 유용하게 사용할 수 있습니다. 코드 16-20처럼 코드를 입력합니다.

코드 16-20 trigger() 메서드를 사용한 매개변수 전달

```
<script>
$(document).ready(function () {
    // 이벤트를 연결합니다.
    $('h1').click(function (event, data1, data2) {
        alert(data1 + ' : ' + data2);
    });

    // 이벤트를 강제로 발생시킵니다.
    $('h1').eq(1).trigger('click', [273, 52]);
});
</script>
```

코드를 실행하면 문자열 '273 : 52'를 출력합니다. trigger() 메서드의 두 번째 매개변수에 배열을 입력하면 데이터가 자동으로 이벤트 리스너에 순서대로 추가 전송됩니다.

16.7 기본 이벤트와 이벤트 전달

1부의 자바스크립트 이벤트를 배우며 기본 이벤트와 이벤트 전달에 대한 내용을 모두 살펴보았습니다. 이 절에서는 jQuery에서 이를 어떻게 적용하는지 살펴봅니다. 기본 이벤트를 제거하고 이벤트 전달을 막을 때는 이벤트 객체에 있는 표 16-8의 메서드를 사용합니다.

표 16-8 이벤트 객체의 메서드

메서드 이름	설명
preventDefault()	기본 이벤트를 제거합니다.
stopPropagation()	이벤트 전달을 제거합니다.

간단한 예제를 작성하며 알아봅시다. 코드 16-21처럼 body 태그를 구성해주세요.

코드 16-21 body 태그 구성

```
<body>
<h1>
<a href="http://hanb.co.kr">Hanb Media</a>
```

```
</h1>  
</body>
```

간단하게 내용물의 범위를 구분하려고 코드 16-22처럼 style 태그를 입력했습니다.

코드 16-22 style 태그 구성

```
<style>  
* {  
    margin: 5px; padding: 5px;  
    border: 3px solid black;  
}  
</style>
```

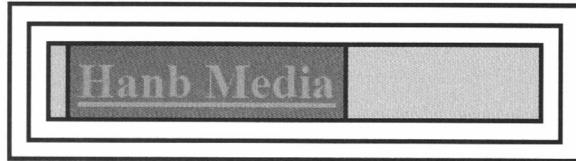
a 태그와 h1 태그에 이벤트를 연결합니다. a 태그를 클릭하면 무조건 기본 이벤트가 실행돼 폐이지가 이동하므로 h1 태그의 이벤트가 소용없겠죠? 코드 16-23처럼 preventDefault() 메서드로 a 태그의 기본 이벤트를 제거합니다.

코드 16-23 preventDefault() 메서드

```
<script>  
$(document).ready(function () {  
    $('a').click(function (event) {  
        $(this).css('background-color', 'blue');  
        event.preventDefault();  
    });  
  
    $('h1').click(function () {  
        $(this).css('background-color', 'red');  
    });  
});  
</script>
```

코드를 실행하고 a 태그를 클릭하면 그림 16-8처럼 a 태그 이외에도 h1 태그에 이벤트 전달이 일어납니다.

그림 16-8 a 태그를 클릭했을 경우의 실행 결과



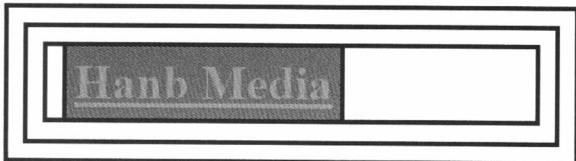
이벤트 전달을 막으려면 코드 16-24처럼 이벤트 객체의 `stopPropagation()` 메서드를 사용합니다.

코드 16-24 `stopPropagation()` 메서드

```
$('a').click(function (event) {  
    $(this).css('background-color', 'blue');  
    event.stopPropagation();  
    event.preventDefault();  
});
```

다시 코드를 실행해 a 태그를 클릭하면 이벤트 전달이 일어나지 않습니다.

그림 16-9 a 태그를 클릭했을 경우의 실행 결과



`stopPropagation()` 메서드와 `preventDefault()` 메서드를 함께 사용하는 경우가 많으므로, jQuery는 간단하게 코드 16-25처럼 `return false`를 사용하면 이 두 가지를 함께 적용하는 것으로 인식합니다.

코드 16-25 `return false`

```
$('a').click(function (event) {  
    $(this).css('background-color', 'blue');  
    return false;  
});
```

따라서 그림 16-8과 같은 실행 결과를 볼 수 있습니다. 기존 자바스크립트는 `return false`를 사용하면 기본 이벤트만 제거된다는 점을 주의해주세요. 두 가지를 헷갈리면 안됩니다.

16.8 이벤트 연결 범위 한정

이벤트를 연결할 때 `on()` 메서드를 사용합니다. 그런데 `on()` 메서드를 사용할 때 넣을 수 있는 매개변수를 살펴본 적 있나요?

그림 16-10 `on()` 메서드의 매개변수

```
$('h1').on()  
  on(types, selector, data, fn, one)
```

넣을 수 있는 매개변수가 지금까지 사용한 것보다 많습니다. `on()` 메서드는 `selector` 매개변수로 이벤트 범위를 한정할 수 있습니다. 이렇게 범위를 한정하는 이벤트 연결 방식을 `delegate` 방식이라고 합니다.

“왜 `delegate`라고 해요?”

이 절의 마지막에서 설명하겠습니다. 일단 코드를 입력하며 살펴보겠습니다. 다음과 같이 `body` 태그를 입력해주세요.

코드 16-26 `body` 태그 구성

```
<body>  
  <div id="wrap">  
    <h1>Header</h1>  
  </div>  
</body>
```

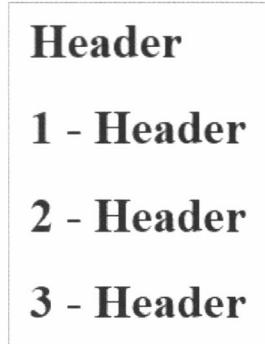
지금까지 배운대로 `on()` 메서드를 사용해보겠습니다.

코드 16-27 기본 이벤트 연결

```
<script>
$(document).ready(function () {
    $('h1').on('click', function () {
        var length = $('h1').length;
        var targetHTML = $(this).html();
        $('#wrap').append('<h1>' + length + ' - ' + targetHTML + '</h1>');
    });
});
</script>
```

코드를 실행하고 h1 태그를 누르면 그림 16-11처럼 h1 태그가 추가됩니다. 하지만, 새로 생긴 h1 태그를 클릭하면 아무 일도 일어나지 않습니다.

그림 16-11 맨 위의 h1 태그를 클릭하면 요소가 추가됨



on() 메서드는 현재 존재하는 태그에만 이벤트를 연결하기 때문입니다. 따라서 이러한 경우에는 상위 태그에 이벤트를 연결하고 “h1 태그를 클릭했을 때”를 검출해야 합니다.

다음 방식으로 이를 구현합니다.

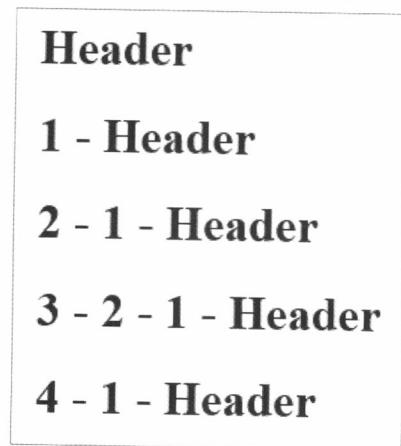
코드 16-28 delegate 방식을 사용하는 on() 메서드

```
<script>
$(document).ready(function () {
    $('#wrap').on('click', 'h1', function () {
        var length = $('h1').length;
        var targetHTML = $(this).html();
        $('#wrap').append('<h1>' + length + ' - ' + targetHTML + '</h1>');
    });
});
</script>
```

```
    });
});
</script>
```

이렇게 하면 새로 생성되는 태그도 이벤트가 적용됩니다.

그림 16-12 어떠한 h1 태그를 선택해도 요소가 추가됨



참고로 이벤트 리스너에서 this 키워드가 #wrap 태그가 아니라 h1 태그라는 것을 주의하세요.
이렇게 연결한 이벤트는 다음과 같은 방식으로 삭제합니다.

코드 16-29 delegate 방식으로 연결한 on() 메서드의 이벤트 리스너 삭제

```
<script>
$(document).ready(function () {
    // 이벤트를 연결합니다.
    $('#wrap').on('click', 'h1', function () {
        var length = $('h1').length;
        var targetHTML = $(this).html();
        $('#wrap').append('<h1>' + length + ' - ' + targetHTML + '</h1>');

        // 이벤트를 제거합니다.
        $('#wrap').off('click', 'h1');
    });
});
</script>
```

NOTE_ document 객체에 연결

상위 개념이 애매한 태그는 다음과 같이 document 객체에 이벤트를 연결하세요.

코드 16-30 document 객체에 연결

```
<script>
$(document).ready(function () {
    // 이벤트를 연결합니다.
    $(document).on('click', 'h1', function () { });
});
</script>
```

NOTE_ 이벤트 연결 방식

jQuery 1.9 버전 이전에는 다음 3개의 이벤트 연결 방식이 있었습니다. 1.9 버전 업데이트와 함께 모든 이벤트 연결 방식을 on() 메서드로 통일했습니다.

표 16-9 과거의 이벤트 연결 방식

메서드 이름	설명
bind()	현재 존재하는 문서 객체에만 이벤트를 연결합니다.
delegate()	현재 또는 미래에 존재하는 문서 객체에 이벤트를 연결합니다.
live()	현재 또는 미래에 존재하는 문서 객체에 이벤트를 연결합니다.

너무 많은 애플리케이션이 표 16-9의 메서드를 사용하고 있어서 없어진 직후 잠시 혼란이 일었습니다. 그래서 bind() 메서드와 delegate() 메서드는 다시 사용할 수 있게 만들었습니다.

하지만 되도록 on() 메서드를 사용합시다. 표 16-10의 메서드는 on() 메서드로 다음과 같이 대체됩니다.

표 16-10 현재의 이벤트 연결 방식

기존 이벤트 연결 방식	on() 메서드를 사용한 연결 방식
\$(h1).bind('click', function () {})	\$(h1).on('click', function () {})
\$('#wrap').delegate('click', 'h1', function () {})	\$('#wrap').on('click', 'h1', function () {})
\$(h1).live('click', function () {})	\$(#wrap).on('click', 'h1', function () {})

이벤트 연결 범위를 한정하는 이벤트 연결 방식을 delegate 방식이라고 했죠? delegate() 메서드에서 이름을 따온 것처럼입니다. 앞으로 “delegate 방식으로 이벤트를 연결합니다!”라고 하면 이처럼 이벤트 연결 범위를 한정하는 방법이라는 것을 기억해주세요.

16.9 마우스 이벤트

마우스 이벤트는 표 16-11과 같습니다.

표 16-11 마우스 이벤트

이벤트 이름	설명
click	마우스를 클릭할 때 발생합니다.
dblclick	마우스를 더블 클릭할 때 발생합니다.
mousedown	마우스 버튼을 누를 때 발생합니다.
mouseup	마우스 버튼을 뗄 때 발생합니다.
mouseenter	마우스가 요소의 경계 외부에서 내부로 이동할 때 발생합니다.
mouseleave	마우스가 요소의 경계 내부에서 외부로 이동할 때 발생합니다.
mousemove	마우스를 움직일 때 발생합니다.
mouseout	마우스가 요소를 벗어날 때 발생합니다.
mouseover	마우스를 요소 안에 들어올 때 발생합니다.

마우스 이벤트와 관련된 예제는 여러 번 살펴보았으므로 mouseover 이벤트와 mouseenter 이벤트, mouseleave 이벤트와 mouseout 이벤트의 차이만 구분해보고 다음 절로 넘어갑니다.

코드 16-31은 div 태그를 두 개 중첩해서 만든 후 외부에 있는 div 태그에 mouseover 이벤트와 mouseenter 이벤트를 모두 연결합니다.

코드 16-31 mouseover 이벤트와 mouseenter 이벤트의 차이

```
<!DOCTYPE html>
<html>
<head>
<style>
.outer {
    width: 200px;
    height: 200px;
    background: orange;
    padding: 50px;
    margin: 10px;
}

.inner {
    width: 100%;
```

```

        height: 100%;
        background: red;
    }

```

```
</style>
```

```
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
```

```
<script>
```

```
$(document).ready(function () {
    $('.outer').mouseover(function () {
        $('body').append('<h1>MOUSEOVER</h1>');
    }).mouseenter(function () {
        $('body').append('<h1>MOUSEENTER</h1>');
    });
});
```

```
</script>
```

```
</head>
```

```
<body>
```

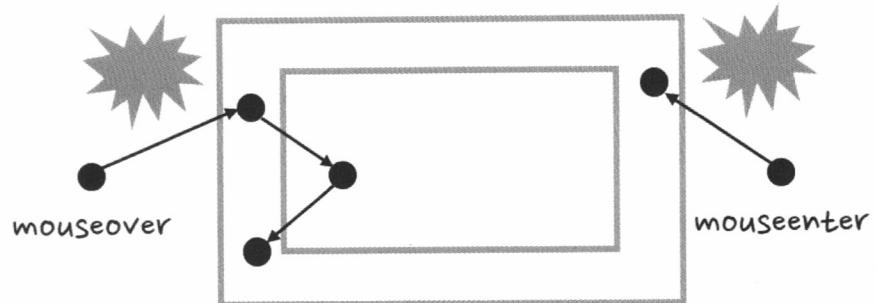
```
<div class="outer">
    <div class="inner"></div>
</div>
```

```
</body>
```

```
</html>
```

코드를 실행해 규칙을 찾으면 그림 16-13처럼 나타낼 수 있습니다. `mouseover` 이벤트는 이벤트 버블링을 적용합니다. 따라서 내부의 div 태그 안에 들어가도 이벤트를 발생시킵니다. 반면 `mouseenter` 이벤트는 문서 객체의 안에 있는지 외부에 있는지 따집니다.

그림 16-13 `mouseover` 이벤트와 `mouseenter` 이벤트



`mouseover` 이벤트는 사실상 현대에서 거의 사용하지 않고 대부분 `mouseenter` 이벤트만 사용하는 것이 추세입니다.

16.10 키보드 이벤트

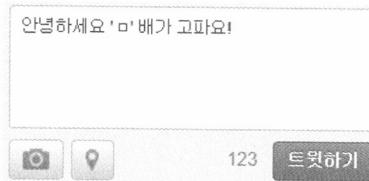
키보드 이벤트는 표 16-12와 같습니다.

표 16-12 키보드 이벤트

이벤트 이름	설명
keydown	키보드를 누를 때 발생합니다.
keypress	글자가 입력될 때 발생합니다.
keyup	키보드를 뗄 때 발생합니다.

이 절에서 실제로 사용되는 예제를 만들며 연습합시다. 그림 16-14는 트위터의 글쓰기 창입니다. 그림 16-14의 오른쪽 하단을 보면 입력 가능한 글자 수가 있습니다. 트위터에는 이렇게 동적으로 글자 수를 세어주는 부분이 있습니다.

그림 16-14 트위터 글 쓰기



NOTE 참고로 keypress 이벤트는 한글로 사용할 수 없습니다. 트위터도 그래서 한글을 입력할 때는 글자 개수를 세지 못합니다.

동적으로 글자 수를 세는 부분을 만들면서 키보드 이벤트와 키보드 이벤트의 주의할 점을 알아봅시다. 우선, 코드 16-32처럼 HTML 페이지를 구성합니다.

코드 16-32 HTML 페이지 구성

```
<!DOCTYPE html>
<html>
<head>
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script>
```

```
</script>
</head>
<body>
    <div>
        <p>지금 내 생각을</p>
        <h1>150</h1>
        <textarea cols="70" rows="5"></textarea>
    </div>
</body>
</html>
```

코드 16-33처럼 textarea 태그에 keyup 이벤트를 연결합니다. keyup 이벤트가 발생하면 글자의 개수를 받아 출력합니다.

코드 16-33 keyup 이벤트 연결

```
<script>
$(document).ready(function (event) {
    $('textarea').keyup(function () {
        // 남은 글자 수를 구합니다.
        var inputLength = $(this).val().length;
        var remain = 150 - inputLength;

        // 문서 객체에 입력합니다.
        $('h1').html(remain);
    });
});
</script>
```

정말 간단한 예제입니다. 하지만, 왜 keyup 이벤트를 사용할까요? 영어를 입력하면 keypress 이벤트를 사용하기 좋지만, 한글은 keypress 이벤트를 지원하지 않습니다. 따라서 keypress 이벤트는 배제합니다.

keydown 이벤트가 아니라 keyup 이벤트를 사용하는 이유는 약간 복잡합니다. 키보드 이벤트가 어떠한 순서로 발생하는지 알아야 합니다. 키보드 이벤트는 다음 순서로 진행됩니다.

- 1 사용자가 키보드를 누릅니다.
- 2 keydown 이벤트가 발생합니다.
- 3 글자가 입력됩니다.

4 keypress 이벤트가 발생합니다.

5 사용자가 키보드에서 손을 뗅니다.

6 keyup 이벤트가 발생합니다.

따라서 keydown 이벤트가 발생한 순간에는 글자가 입력돼 있지 않습니다. 입력한 글자 수를 표시해야 하므로 keyup 이벤트를 사용합니다.

“계속 누르고만 있으면 어떻게 해요?”

이 방법으로는 앞의 경우를 처리할 수 없습니다. 사실 keypress 이벤트를 보완하는 방법을 더 많이 사용합니다. textarea 태그에 입력을 시작하는 순간부터 50밀리 초마다 특정 함수를 발생시키는 것입니다. 하지만, 이는 복잡하므로 이 책에서 다루지 않겠습니다.

추가로 코드 16-34처럼 코드의 양수 음수에 따라 색상을 변경하게 만들었습니다.

코드 16-34 글자 개수에 따른 스타일 변경

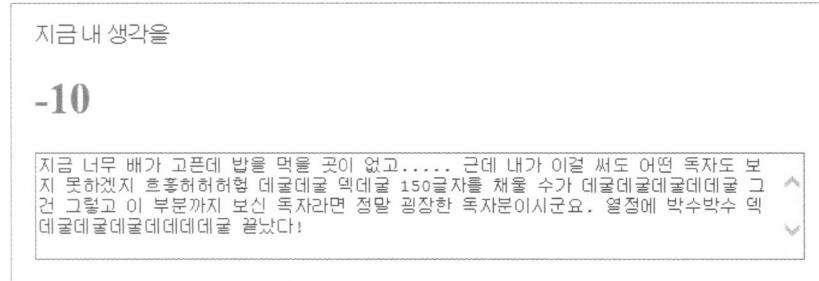
```
<script>
$(document).ready(function (event) {
    $('textarea').keyup(function () {
        // 남은 글자 수를 구합니다.
        var inputLength = $(this).val().length;
        var remain = 150 - inputLength;

        // 문서 객체에 입력합니다.
        $('h1').html(remain);

        // 문서 객체의 색상을 변경합니다.
        if (remain >= 0) {
            $('h1').css('color', 'black');
        } else {
            $('h1').css('color', 'red');
        }
    });
});
</script>
```

코드를 실행하고 글자를 입력하면 그림 16-15처럼 결과를 확인할 수 있습니다.

그림 16-15 지금 내 생각을....



16.11 윈도 이벤트

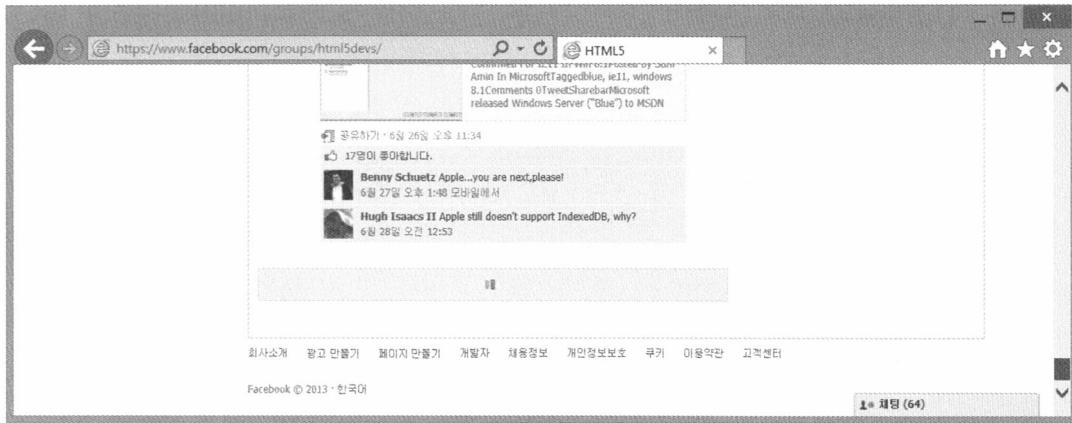
윈도 이벤트는 윈도 객체만 사용할 수 있는 이벤트가 아니라 window 객체와 document 객체 이외에 img 태그 등이 사용할 수 있는 이벤트입니다. 윈도 객체는 표 16-13과 같은 이벤트를 사용할 수 있습니다.

표 16-13 윈도 이벤트

이벤트	설명
ready	문서 객체가 준비 완료되면
load	윈도(문서 객체)를 불러들일 때 발생합니다.
unload	윈도(문서 객체)를 닫을 때 발생합니다.
resize	윈도의 크기를 변화시킬 때 발생합니다.
scroll	윈도를 스크롤할 때 발생합니다.
error	에러가 있을 때 발생합니다.

이 절에서는 윈도 이벤트로 무한 스크롤을 구현합니다. 무한 스크롤이란 최근 SNS와 같은 서비스에서 자주 볼 수 있는 효과입니다. 그림 16-16은 페이스북의 무한 스크롤입니다.

그림 16-16 페이스북의 무한 스크롤



이 사이트는 스크롤이 문서 끝까지 내려가면 자동으로 포스트를 추가해 스크롤을 늘립니다. 스크롤을 무한히 추가하므로 이를 무한 스크롤이라고 합니다. 참고로 아이폰이나 안드로이드폰 같은 스마트폰에서는 스크롤 오차가 있으므로 정확한 무한 스크롤을 구현할 수 없습니다. 이는 이 절의 마지막 부분에서 다룹니다.

이제 jQuery의 이벤트로 간단하게 무한 스크롤을 구현해봅시다. 코드 16-35처럼 기본적인 HTML 페이지를 구성합니다.

코드 16-35 HTML 페이지 구성

```
<!DOCTYPE html>
<html>
<head>
  <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
  <script>
    $(document).ready(function () {
      });
    </script>
  </head>
  <body>
  </body>
</html>
```

다음 방식으로 window 객체에 scroll 이벤트를 연결합니다. window 객체는 별도의 선택자를 사용하지 않는다는 점을 기억해주세요.

코드 16-36 scroll 이벤트 연결

```
<script>
$(document).ready(function () {
    $(window).scroll(function () {
        });
    });
</script>
```

사용자가 마우스 스크롤을 움직이면 무조건 scroll 이벤트가 발생합니다. 무한 스크롤을 만들려면 화면 끝까지 스크롤이 도달했다는 사실을 인식할 수 있어야 합니다. document 객체의 height 속성은 문서 전체의 높이를 의미합니다. 스크롤이 끝까지 내려가면 window 객체의 scrollTop 속성과 height 속성을 합한 값이 document 객체의 높이와 같아집니다. 이를 사용해 스크롤이 끝까지 도달했다는 것을 인식할 수 있습니다.

그림 16-17 무한 스크롤 이벤트의 구성



이 내용은 간단하게 다음 코드를 구성하면 됩니다. 스크롤이 끝까지 닿으면 자동으로 h1 태그 10개를 추가할 수 있게 만들었습니다.

코드 16-37 무한 스크롤

```
<script>
$(document).ready(function () {
    $(window).scroll(function () {
        var scrollHeight = $(window).scrollTop() + $(window).height();
        var documentHeight = $(document).height();
        if (scrollHeight == documentHeight - 200) {
            for (var i = 0; i < 10; i++) {
                $('<h1>Infinity Scroll</h1>').appendTo('body');
            }
        }
    });
});
</script>
```

documentHeight 속성에 -200을 하고 비교한 것은 웹 브라우저마다 미세한 오차가 있기 때문입니다.

최종적으로 완성한 코드는 다음과 같습니다. 간단한 테스트를 위해서 \$(document).ready() 를 두 번 사용했습니다. 첫 번째를 원하는 곳에 복사해 사용하기 쉽게 하려고 이렇게 구성했습니다. 무한 스크롤이 필요할 때는 위의 부분만 복사해서 사용하면 됩니다.

코드 16-38 무한 스크롤 전체 코드

```
<script>
// 무한 스크롤 부분
$(document).ready(function () {
    // 스크롤 이벤트 발생 시
    $(window).scroll(function () {
        // 필요한 변수를 구합니다.
        var scrollHeight = $(window).scrollTop() + $(window).height();
        var documentHeight = $(document).height();

        // 스크롤의 높이와 문서의 높이가 같을 때
        if (scrollHeight == documentHeight) {
            for (var i = 0; i < 10; i++) {
                $('<h1>Infinity Scroll</h1>').appendTo('body');
            }
        }
    });
});
</script>
```

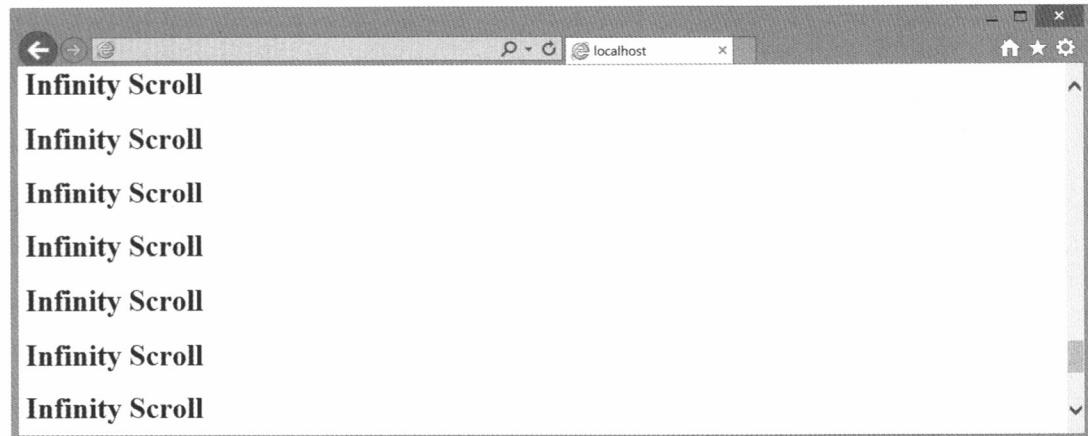
```
});

// 테스트를 위해 내부에 공간을 채워둡니다.
$(document).ready(function () {
    for (var i = 0; i < 20; i++) {
        $('<h1>Infinity Scroll</h1>').appendTo('body');
    }
});
</script>
```

두 번째 \$(document).ready() 이벤트에서 h1 태그를 20개 생성했는데요. 스크롤 이벤트를 발생시켜야 무엇을 해볼 수 있는데, 현재 문서에 아무 내용도 없으니 스크롤이 생기지 않습니다. 그래서 20개 정도 만들어 스크롤을 생성했습니다.

코드를 실행하고 스크롤을 내려보세요. 아무리 내려도 끝까지 닿을 수 없을 것입니다.

그림 16-18 무한 스크롤



16.12 입력 양식 이벤트

입력 양식과 관련된 이벤트는 표 16-14와 같습니다. 기존의 자바스크립트와 내용이 같으므로 활용 위주로 살펴보고 넘어갑니다.

표 16-14 입력 양식 이벤트

이벤트 이름	설명
change	입력 양식의 내용을 변경할 때 발생합니다.
focus	입력 양식에 초점을 맞추면 발생합니다.
focusin	입력 양식에 초점이 맞추어지기 바로 전에 발생합니다.
focusout	입력 양식에 초점이 사라지기 바로 전에 발생합니다.
blur	입력 양식에 초점이 사라지면 발생합니다.
select	입력 양식을 선택할 때 발생합니다(input[type="text"] 태그 및 textarea 태그 제외).
submit	submit 버튼을 누르면 발생합니다.
reset	reset 버튼을 누르면 발생합니다.

submit 이벤트부터 살펴보겠습니다. submit 이벤트를 살펴보려면 입력 양식이 있어야 합니다. 코드 16-39처럼 body 태그를 구성합니다.

코드 16-39 body 태그 구성

```
<body>
  <form id="my-form">
    <table>
      <tr>
        <td>이름: </td>
        <td><input type="text" name="name" id="name" /></td>
      </tr>
      <tr>
        <td>비밀번호: </td>
        <td><input type="password" name="password" id="password" /></td>
      </tr>
    </table>
    <input type="submit" value="제출" />
  </form>
</body>
```

submit 이벤트는 form 태그에서 발생하는 이벤트입니다. 따라서 form 객체에 submit() 메서드를 연결합니다. 입력 양식의 유효성 검사를 할 때는 기본 이벤트를 제거해야 합니다. submit 이벤트의 기본 이벤트는 코드 16-40과 같은 방식으로 제거합니다.

코드 16-40 submit 이벤트와 기본 이벤트 제거

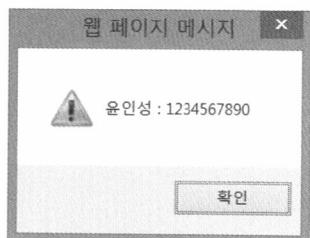
```
<script>
$(document).ready(function () {
    $('#my-form').submit(function (event) {
        // 입력 양식의 value를 가져옵니다.
        var name = $('#name').val();
        var password = $('#password').val();

        // 출력합니다.
        alert(name + ' : ' + password);

        // 기본 이벤트를 제거합니다.
        event.preventDefault();
    });
});
</script>
```

유효성 검사와 관련된 내용은 나중에 다루겠습니다. 코드를 실행하고 입력 양식을 입력한 후 제출 버튼을 누르면 그림 16-19처럼 경고창이 출력됩니다.

그림 16-19 실행 결과



type 속성이 checkbox와 radio인 input 태그의 상태를 변경하는 이벤트는 click 이벤트가 아니라 change 이벤트입니다.

코드 16-41 check 속성 변경

```
<script>
$(document).ready(function () {
    $('#all-check').change(function () {
        if (this.checked) {
            $('#check-item').children().prop('checked', true);
        } else {
```

```
$('#check-item').children().prop('checked', false);
}
});
});
});
</script>
<body>
<input type="checkbox" id="all-check" />
<label>All</label>
<div id="check-item">
<input type="checkbox" />
<label>A Option</label>
<input type="checkbox" />
<label>B Option</label>
<input type="checkbox" />
<label>C Option</label>
</div>
</body>
```

또한 체크 상태를 확인할 때는 입력 객체의 checked 속성을 확인하고, 체크 상태를 바꿀 때는 prop() 메서드를 사용합니다. prop() 메서드는 이 경우에만 사용하므로 간단하게 짧고 넓어 가세요!