

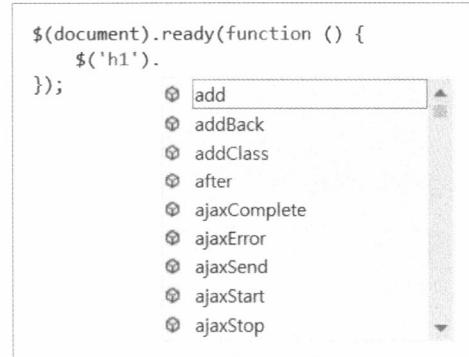
문서 객체 조작

자바스크립트를 공부하면서 문서 객체 모델을 살펴보았습니다. 기존의 자바스크립트만으로 문서 객체 모델을 다루려면 정말 복잡합니다. jQuery를 사용하면 이렇게 복잡한 문서 객체 모델을 쉽게 다룰 수 있습니다. 15장은 이 책에서 가장 중요합니다.

15장에서는 많은 메서드를 다룹니다. 모두 외우려고 하지 말고 자세한 이름이나 기능은 그림 15-1처럼 Visual Studio 2012 Express for Web의 보조 기능을 활용하세요.

“기능을 수행하는 메서드 이름이 대충 이것 같은데?” 정도의 생각을 하며 그림 15-1에서 원하는 메서드를 찾는 정도까지가 이 장에서 추구하는 가장 큰 목표입니다.

그림 15-1 jQuery 보조 기능



실무를 진행하면서 메서드를 여러 번 사용하면 자동으로 외워집니다. 15장에서는 모든 메서드를 예제와 함께 진행하며 살펴봅니다. 모두 직접 입력하면서 따라해보세요.

15.1 문서 객체의 클래스 속성 추가

가장 간단한 형태의 메서드부터 살펴봅시다. 표 15-1의 jQuery 메서드는 문서 객체에 클래스 속성을 추가할 때 사용하는 메서드입니다.

표 15-1 문서 객체의 클래스 속성 추가 메서드

메서드 이름	설명
addClass()	문서 객체의 클래스 속성을 추가합니다.

예제를 만들며 익혀봅시다. 코드 15-1처럼 body 태그를 구성해주세요.

코드 15-1 body 태그 구성

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

addClass() 메서드는 코드 15-2처럼 매개변수에 추가하려는 클래스 속성의 이름을 입력합니다.

코드 15-2 addClass() 메서드

```
<script>
$(document).ready(function () {
  $('h1').addClass('item');
});
</script>
```

코드를 실행하면 그림 15-2처럼 클래스 속성이 적용된 것을 확인할 수 있습니다.

그림 15-2 addClass() 메서드

```
▼<body>
  <h1 class="item">Header-0</h1>
  <h1 class="item">Header-1</h1>
  <h1 class="item">Header-2</h1>
</body>
```

15장에서 배울 jQuery 메서드는 대부분 이렇게 매개변수에 문자열을 입력하는 간단한 방법 외에 코드 15-3처럼 함수를 매개변수로 입력할 수도 있습니다.

코드 15-3 addClass() 메서드의 콜백 함수

```
<script>
$(document).ready(function () {
    $('h1').addClass(function (index) {
        return 'class' + index;
    });
});
</script>
```

addClass() 메서드의 매개변수로 입력하는 함수는 코드 15-3처럼 index 매개변수를 갖습니다. 선택자로 선택한 문서 객체에 차례대로 속성을 지정할 수 있습니다. 코드를 실행하면 그림 15-3처럼 출력합니다.

그림 15-3 클래스가 순서대로 추가

```
▼<body>
  <h1 class="class0">Header-0</h1>
  <h1 class="class1">Header-1</h1>
  <h1 class="class2">Header-2</h1>
</body>
```

간단한 내용이므로 쉽게 이해했을 것이라고 생각합니다.

15.2 문서 객체의 클래스 속성 제거

문서 객체의 클래스 속성을 제거할 때는 표 15-2의 메서드를 사용합니다.

표 15-2 문서 객체의 클래스 속성 제거 메서드

메서드 이름	설명
removeClass()	문서 객체의 클래스 속성을 제거합니다.

`removeClass()` 메서드의 매개변수에 아무것도 입력하지 않으면, 문서 객체의 모든 클래스를 제거합니다. 코드 15-4처럼 `body` 태그를 구성합니다.

코드 15-4 `body` 태그 구성

```
<body>
  <h1 class="item">Header-0</h1>
  <h1 class="item select">Header-1</h1>
  <h1 class="item">Header-2</h1>
</body>
```

`removeClass()` 메서드는 코드 15-5처럼 매개변수에 제거하려는 클래스 속성의 이름을 입력합니다.

코드 15-5 `removeClass()` 메서드

```
<script>
$(document).ready(function () {
  $('h1').removeClass('select');
});
</script>
```

코드를 실행하면 그림 15-4처럼 `select` 클래스 속성이 제거된 것을 확인할 수 있습니다.

그림 15-4 클래스 제거

```
▼<body>
  <h1 class="item">Header-0</h1>
  <h1 class="item">Header-1</h1>
  <h1 class="item">Header-2</h1>
</body>
```

NOTE `addClass()` 메서드와 `removeClass()` 메서드 이외에 `toggleClass()` 메서드도 존재합니다.

`toggleClass()` 메서드는 매개변수로 입력한 클래스 속성이 이미 지정돼 있으면 제거하고, 지정돼 있지 않으면 추가합니다.

15.3 문서 객체의 속성 검사

jQuery에서 문서 객체의 속성과 관련된 모든 기능은 표 15-3의 attr() 메서드가 처리합니다.

표 15-3 문서 객체의 속성 검사 메서드

메서드 이름	설명
attr()	속성과 관련된 모든 기능을 수행합니다.

이 절에서는 attr() 메서드로 문서 객체의 속성을 알아내는 기능을 배웁니다. body 태그를 코드 15-6처럼 구성합니다.

코드 15-6 body 태그 구성

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

attr() 메서드로 특정 속성의 값을 알아내고 싶을 때는 코드 15-7처럼 매개변수에 속성 이름을 입력합니다.

코드 15-7 attr() 메서드 – Getter

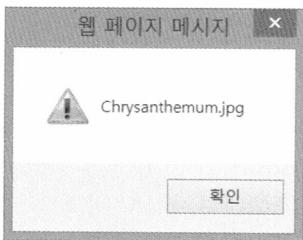
```
<script>
$(document).ready(function () {
  // 변수를 선언합니다.
  var src = $('img').attr('src');

  // 출력합니다.
  alert(src);
});
</script>
```

잠시 이상한 생각이 들 수 있습니다. 코드 15-7의 선택자를 살펴보면 img 태그를 선택하는데, img 태그는 세 개입니다. attr() 메서드로 속성 값을 출력하면 어떤 문서 객체의 속성을 출력할까요? 코드를 실행해보면 바로 알 수 있겠죠?

코드를 실행하면 그림 15-5처럼 첫 번째 문서 객체의 속성을 출력하는 것을 확인할 수 있습니다.

그림 15-5 attr() 메서드를 사용한 속성 검사



대부분의 값을 알아내는 jQuery 메서드는 여러 개의 문서 객체를 선택했을 때 첫 번째에 위치한 문서 객체의 값을 출력합니다. 앞으로도 여러 번 볼 것입니다. 기억하고 넘어갑시다.

15.4 문서 객체의 속성 추가

jQuery에서 문서 객체의 속성과 관련된 것은 모두 attr() 메서드가 처리한다고 했지요? 문서 객체에 속성을 추가할 때도 attr() 메서드를 사용합니다. attr() 메서드는 다음과 같은 세 가지 형태로 사용할 수 있습니다.

- 1 \$(selector).attr(name, value);
- 2 \$(selector).attr(name, function(index, attr) { });
- 3 \$(selector).attr(object);

body 태그를 코드 15-8처럼 구성하고 각각의 형태를 사용해봅시다.

코드 15-8 body 태그 구성

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

1번 형태입니다. 첫 번째 매개변수에는 속성 이름을 입력하고, 두 번째 매개변수에는 값을 넣어줍니다. 코드 15-9는 width 속성에 200을 입력합니다.

코드 15-9 attr() 메서드 – Setter(1)

```
<script>
$(document).ready(function () {
    $('img').attr('width', 200);
});
</script>
```

코드를 실행하면 그림 15-6처럼 모든 img 태그의 width 속성이 변경됩니다.

그림 15-6 attr() 메서드를 사용한 속성 부여(1)

```
▼ <body>
  
  
  
</body>
```

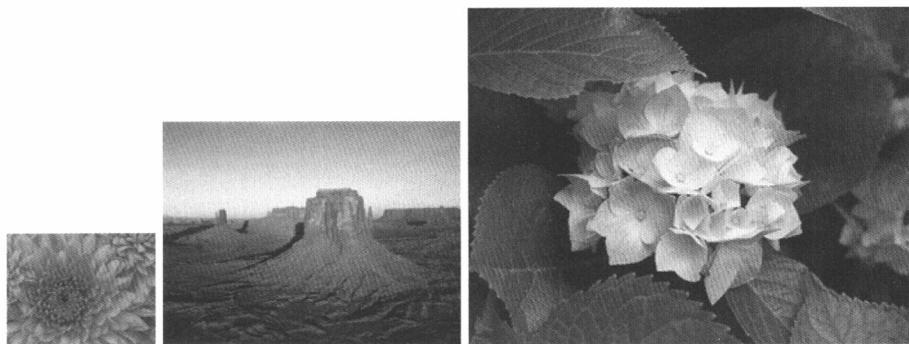
이제 attr() 메서드의 2번 형태를 살펴봅시다. 첫 번째 매개변수에는 속성의 이름을 입력하고, 두 번째 매개변수에는 index 매개변수를 갖는 함수를 입력합니다. 이전에 살펴본 addClass()와 비슷한 내용입니다. 코드 15-10은 각각의 img 태그에 단계적으로 크기를 적용합니다.

코드 15-10 attr() 메서드 – Setter(2)

```
<script>
$(document).ready(function () {
    $('img').attr('width', function (index) {
        return (index + 1) * 100;
    });
});
</script>
```

코드를 실행하면 그림 15-7처럼 단계적인 크기의 이미지를 볼 수 있습니다.

그림 15-7 attr() 메서드를 사용한 속성 부여(2)



attr() 메서드의 3번 형태는 코드 15-11처럼 매개변수에 객체를 입력합니다. 매개변수로 입력한 객체에는 키로 속성의 이름을 지정하고 값을 입력합니다. 값을 입력할 때는 1번 형태와 2번 형태를 모두 사용할 수 있습니다.

코드 15-11 attr() 메서드 – Setter(3)

```
<script>
$(document).ready(function () {
    $('img').attr({
        width: function (index) {
            return (index + 1) * 100;
        },
        height: 100
    });
})</script>
```

코드를 실행하면 그림 15-8처럼 속성을 출력하는 것을 확인할 수 있습니다.

그림 15-8 attr() 메서드를 사용한 속성 부여(3)

```
▼<body>



</body>
```

15장에서 대부분의 메서드는 attr() 메서드처럼 검사하는 형태는 1가지이고 추가하는 형태는 3가지입니다.

15.5 문서 객체의 속성 제거

문서 객체의 속성을 제거할 때는 표 15-4의 메서드를 사용합니다.

표 15-4 문서 객체의 속성 제거 메서드

메서드 이름	설명
removeAttr(name)	문서 객체의 속성을 제거합니다.

간단한 메서드이므로 간단하게 살펴봅시다. 코드 15-12처럼 body 태그를 구성합니다.

코드 15-12 body 태그 구성

```
<body>
  <h1 data-index="0">Header-0</h1>
  <h1 data-index="1">Header-1</h1>
  <h1 data-index="2">Header-2</h1>
</body>
```

removeAttr() 메서드는 첫 번째 매개변수에 삭제하려는 속성의 이름을 입력합니다. 코드 15-13은 h1 태그에 있는 data-index 속성을 제거합니다.

코드 15-13 removeAttr() 메서드

```
<script>
$(document).ready(function () {
  $('h1').removeAttr('data-index');
});
</script>
```

코드를 실행하면 그림 15-9처럼 data-index 속성이 제거됩니다.

그림 15-9 removeAttr() 메서드를 사용한 속성 제거

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

NOTE `removeClass()` 메서드와 `removeAttr()` 메서드 모두 클래스 속성을 제거할 수 있습니다. 하지만, `removeAttr()` 메서드를 사용하면 모든 클래스 속성이 한 번에 제거되는 것과 달리 `removeClass()` 메서드를 사용하면 여러 개의 클래스 속성 중 선택적으로 제거할 수 있습니다.

15.6 문서 객체의 스타일 검사

jQuery에서 문서 객체의 스타일 속성과 관련된 작업은 모두 표 15-5의 `css()` 메서드가 처리합니다.

표 15-5 문서 객체의 스타일 검사 메서드

메서드 이름	설명
<code>css()</code>	스타일과 관련된 모든 기능을 수행합니다.

지금까지 몇 번 사용해왔으므로 부담 없을 것이라고 생각합니다. 이 절에서는 `css()` 메서드로 스타일 속성을 알아내는 방법부터 살펴봅시다. 코드 15-14처럼 HTML 페이지를 구성해주세요. `style` 태그를 입력해서 각 태그에 스타일 속성을 지정했습니다.

코드 15-14 HTML 페이지 구성

```
<!DOCTYPE html>
<html>
<head>
<style>
    .first { color: red; }
    .second { color: pink; }
    .third { color: orange; }
</style>
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script>
$(document).ready(function () {
    });
</script>
</head>
```

```
<body>
  <h1 class="first">Header-0</h1>
  <h1 class="second">Header-1</h1>
  <h1 class="third">Header-2</h1>
</body>
</html>
```

css() 메서드로 특정 문서 객체의 스타일 속성을 알고 싶을 때는 코드 15-15처럼 css() 메서드의 매개변수에 스타일 속성 이름을 입력합니다.

코드 15-15 css() 메서드 – Getter

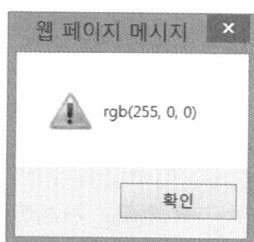
```
<script>
$(document).ready(function () {
  // 변수를 선언합니다.
  var color = $('h1').css('color');

  // 출력합니다.
  alert(color);
});
</script>
```

이때 입력하는 속성을 살펴보면 CSS의 background-color 스타일 속성의 경우 문자열 그대로 'background-color'를 넣어도 되고 'backgroundColor'의 형태로 넣어도 됩니다.

코드를 실행하면 첫 번째 문서 객체의 스타일 속성을 가져옵니다.

그림 15-10 css() 메서드를 사용한 속성 검사



15.7 문서 객체의 스타일 추가

문서 객체에 스타일을 추가하는 것은 지금까지 여러 번 사용했습니다. 지금까지 사용한 형태 외에도 다음 형태로 `css()` 메서드를 사용할 수 있습니다.

- 1 `$(selector).css(name, value);`
- 2 `$(selector).css(name, function(index, style) { });`
- 3 `$(selector).css(object);`

`attr()` 메서드와 같으므로 이해하기 쉬울 것입니다. `body` 태그를 다음과 같이 구성합니다.

코드 15-16 `body` 태그 구성

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

1번 형태는 **코드 15-17**처럼 첫 번째 매개변수에 스타일 속성의 이름을 입력하고 두 번째 매개변수에 스타일 속성의 값을 입력합니다. **코드 15-17**에서는 `color` 스타일 속성에 `red`를 입력합니다.

코드 15-17 `css()` 메서드 – Setter(1)

```
<script>
  $(document).ready(function () {
    $('h1').css('color', 'red');
  });
</script>
```

2번 형태는 1번 형태와 비슷하지만, 두 번째 매개변수에 함수를 입력합니다. 선택자로 선택한 문서 객체에 개별적으로 스타일 속성을 적용하고 싶을 때 사용하는 방법입니다. **코드 15-18**은 배열을 선언하고 배열의 내용물을 차례대로 문서 객체의 `color` 스타일 속성에 지정합니다.

코드 15-18 css() 메서드 – Setter(2)

```
<script>
$(document).ready(function () {
    // 변수를 선언합니다.
    var color = ['red', 'white', 'purple'];

    // 문서 객체의 스타일을 변경합니다.
    $('h1').css('color', function (index) {
        return color[index];
    });
});
</script>
```

코드를 실행하면 그림 15-11처럼 순서대로 스타일 속성이 적용된 것을 확인할 수 있습니다.

그림 15-11 css() 메서드를 사용한 스타일 속성 부여(1)

```
▼<body>
<h1 style="color: red; ">Header-0</h1>
<h1 style="color: white; ">Header-1</h1>
<h1 style="color: purple; ">Header-2</h1>
</body>
```

3번 형태도 attr() 메서드와 마찬가지이므로 금방 이해할 수 있을 것입니다. 3번 형태는 css() 메서드의 매개변수에 객체를 입력하는데요. 객체의 키는 코드 15-19처럼 backgroundColor 식별자를 넣을 수도 있고, background-color나 backgroundColor와 같은 문자열을 입력할 수도 있습니다.

코드 15-19 css() 메서드 – Setter(3)

```
<script>
$(document).ready(function () {
    // 변수를 선언합니다.
    var color = ['red', 'white', 'purple'];

    // 문서 객체의 스타일을 변경합니다.
    $('h1').css({
        color: function (index) {
            return color[index];
        },
    });
});
</script>
```

```
        backgroundColor: 'black'
    });
});
</script>
```

코드 15-19의 결과는 스스로 확인해보세요!

15.8 문서 객체의 내부 검사

기존의 자바스크립트에서 문서 객체의 `innerHTML`, `textContent` 속성과 관련된 jQuery 메서드가 표 15-6의 메서드입니다.

표 15-6 문서 객체의 내부 검사 메서드

메서드 이름	설명
<code>html()</code>	문서 객체 내부의 글자와 관련된 모든 기능을 수행합니다(HTML 태그 인식).
<code>text()</code>	문서 객체 내부의 글자와 관련된 모든 기능을 수행합니다.

두 메서드의 차이는 `html()` 메서드는 HTML 태그를 인식한다는 것이고, `text()` 메서드는 HTML 태그를 인식하지 않는다는 것입니다. 하지만, 문서 객체의 내용물을 검사할 때는 이외에도 한 가지 더 차이가 있습니다. `body` 태그를 코드 15-20처럼 구성하고 예제를 작성하며 알아봅시다.

코드 15-20 `body` 태그 구성

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

`html()` 메서드와 `text()` 메서드에 아무것도 입력하지 않으면 문서 객체 내부의 내용물을 검사할 수 있습니다. 코드 15-21은 `html()` 메서드로 문서 객체의 내용물을 출력합니다. 어떠한 출력 결과를 낼지 미리 예측해보세요.

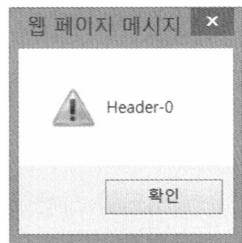
코드 15-21 html() 메서드 – Getter

```
<script>
$(document).ready(function () {
    // 변수를 선언합니다.
    var html = $('h1').html();

    // 출력합니다.
    alert(html);
});
</script>
```

코드를 실행하면 그림 15-12처럼 첫 번째 문서 객체의 내용물을 출력합니다. 지금까지 배운 다른 메서드와 비슷하므로 무리 없이 이해할 수 있을 것입니다.

그림 15-12 html() 메서드를 사용한 객체 내부 문자 검사



하지만, `text()` 메서드는 지금까지 살펴본 메서드와 약간 특성이 다릅니다. 코드 15-22는 코드 15-21과 거의 유사한 코드입니다. 차이점이 있다면 `html()` 메서드를 `text()` 메서드로 바꾸었다는 것입니다.

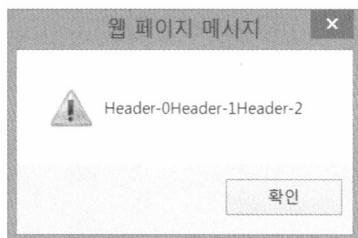
코드 15-22 text() 메서드 – Getter

```
<script>
$(document).ready(function () {
    // 변수를 선언합니다.
    var text = $('h1').text();

    // 출력합니다.
    alert(text);
});
</script>
```

코드 15-22를 실행하면 그림 15-13처럼 출력합니다. `text()` 메서드는 첫 번째 문서 객체의 글자를 가져오는 것이 아닙니다. 선택자로 선택한 모든 문서 객체의 글자를 이어서 출력합니다.

그림 15-13 `text()` 메서드를 사용한 객체 내부 문자 검사



자주 실수할 수 있는 부분이므로 기억해주세요. 지금까지 배운 모든 메서드 중 검사하는 경우는 하나의 특정한 문서 객체를 선택해서 사용하는 것을 추천합니다.

15.9 문서 객체의 내부 추가

문서 객체의 내부에 내용물을 추가하고 싶을 때도 `html()` 메서드와 `text()` 메서드를 사용합니다. 두 메서드 모두 다음과 같은 형태로 사용합니다.

```
1 $(selector).html(value);
  $(selector).text(value);

2 $(selector).html(function(index, html) { });
  $(selector).text(function(index, text) { });
```

예제를 만들면서 살펴봅니다. `body` 태그를 코드 15-23처럼 구성해주세요.

코드 15-23 `body` 태그 구성

```
<body>
  <div></div>
  <div></div>
  <div></div>
</body>
```

이어서 `html()` 메서드를 사용해봅시다. `html()` 메서드는 매개변수로 입력하고자 하는 HTML 문자열을 입력합니다. `html()` 메서드는 HTML 태그를 인식하여 문자열을 삽입합니다.

코드 15-24 `html()` 메서드 – Setter(1)

```
<script>
$(document).ready(function () {
    $('div').html('<h1>$().html() Method</h1>');
});
</script>
```

따라서 그림 15-14처럼 HTML 태그가 삽입된 것을 확인할 수 있습니다.

그림 15-14 `html()` 메서드를 사용한 문서 객체 내부 문자 추가(1)

```
▼<body>
  ▼<div>
    <h1>$().html() Method</h1>
  </div>
  ▼<div>
    <h1>$().html() Method</h1>
  </div>
  ▼<div>
    <h1>$().html() Method</h1>
  </div>
</body>
```

반면 `text()` 메서드는 HTML 태그를 인식하지 않습니다.

코드 15-25 `text()` 메서드 – Setter

```
<script>
$(document).ready(function () {
    $('div').text('<h1>$().html() Method</h1>');
});
</script>
```

문자열을 문자열 그대로 입력하므로 그림 15-15처럼 출력합니다.

그림 15-15 text() 메서드를 사용한 문서 객체 내부 문자 추가

```
▼<body>
  <div>&lt;h1&gt;$().text() Method</h1&gt;</div>
  <div>&lt;h1&gt;$().text() Method</h1&gt;</div>
  <div>&lt;h1&gt;$().text() Method</h1&gt;</div>
</body>
```

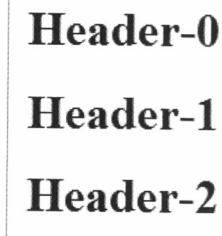
함수를 매개변수로 입력할 때는 index를 매개변수로 넣습니다.

코드 15-26 html() 메서드 – Setter(2)

```
<script>
$(document).ready(function () {
  $('div').html(function (index) {
    return '<h1>Header-' + index + '</h1>';
  });
});
</script>
```

코드를 실행하면 그림 15-16처럼 출력합니다.

그림 15-16 html() 메서드를 사용한 문서 객체 내부 문자 추가(2)



html() 메서드의 매개변수로 입력하는 함수는 두 번째 매개변수로 원래 있던 HTML 내용물을 얻을 수 있습니다. 물론 지금까지 살펴본 메서드에도 이러한 기능이 있지만, 잘 사용하지 않으므로 넘어갔습니다. html() 메서드의 매개변수로 입력한 함수의 두 번째 매개변수는 자주 사용하므로 살펴봅시다. body 태그를 코드 15-27처럼 구성합니다.

코드 15-27 body 태그 구성

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

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

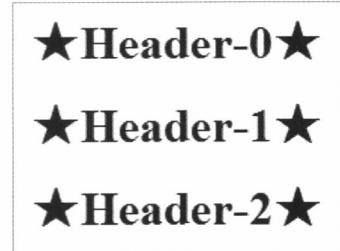
코드 15-28처럼 `html()` 메서드의 매개변수로 입력한 함수의 매개변수를 두 개 사용하면 두 번째 매개변수에 원래 있던 HTML의 내용을 의미합니다.

코드 15-28 `html()` 메서드 – Setter(3)

```
<script>
$(document).ready(function () {
    $('h1').html(function (index, html) {
        return '★' + html + '★';
    });
});
</script>
```

기존 HTML 내용의 양 옆에 별 모양을 추가했으므로 그림 15-17처럼 출력됩니다. `text()` 메서드도 마찬가지 방법으로 사용합니다.

그림 15-17 `html()` 메서드를 사용한 문서 객체 내부 문자 추가(3)



지금까지 살펴본 `attr()`, `css()`, `html()`, `text()` 메서드 모두 사용 방법이 비슷하므로 함께 기억해주세요.

15.10 문서 객체 제거

문서 객체를 제거할 때는 표 15-7의 메서드를 사용합니다.

표 15-7 문서 객체의 제거 메서드

메서드 이름	설명
remove()	문서 객체를 제거합니다.
empty()	문서 객체 내부를 비웁니다.

표 15-7의 empty() 메서드는 선택한 문서 객체의 내부에 들어 있는 모든 문서 객체를 제거합니다. body 태그를 코드 15-29처럼 구성합니다.

코드 15-29 body 태그 구성

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

remove() 메서드는 코드 15-30처럼 사용합니다. 코드 15-30은 선택한 h1 태그 중 첫 번째 문서 객체를 제거합니다.

코드 15-30 remove() 메서드

```
<script>
$(document).ready(function () {
  $('h1').first().remove();
});
</script>
```

코드를 실행하면 그림 15-18처럼 첫 번째 h1 태그가 제거된 것을 확인할 수 있습니다.

그림 15-18 remove() 메서드를 사용한 문서 객체 제거

Header-1

empty() 메서드는 코드 15-31처럼 사용합니다.

코드 15-31 empty() 메서드

```
<script>
$(document).ready(function () {
    $('div').empty();
});
</script>
```

div 태그에 empty() 메서드를 적용했으므로 body 태그의 모든 h1 태그가 제거됩니다.

15.11 문서 객체 생성(1)

문서 객체는 크게 텍스트 노드를 갖는 문서 객체와 텍스트 노드를 갖지 않는 문서 객체로 나눌 수 있습니다. 1부에서는 이 두 종류의 문서 객체를 생성하는 것이 조금 복잡했는데요. jQuery를 사용하면 정말 간단하게 문서 객체를 생성할 수 있습니다. 이 절에서는 텍스트 노드를 갖는 문서 객체를 생성하는 방법부터 알아봅시다.

문서 객체를 생성할 때는 표 15-8의 메서드를 사용합니다. jQuery() 메서드는 선택자로 문서 객체를 선택하는 기능 이외에도 문서 객체를 생성하는 기능이 있습니다.

표 15-8 jQuery 메서드

메서드 이름	설명
\$()	문서 객체를 생성합니다.

문서 객체를 새로 생성해서 body 태그에 추가하는 예제를 작성할 것이므로 body 태그는 코드 15-32처럼 비워줍니다.

코드 15-32 body 태그 구성

```
<body>
```

```
</body>
```

`$()` 메서드로 문서 객체를 생성할 때는 코드 15-33처럼 합니다. `$()` 메서드의 매개변수에 HTML 태그를 문자열로 넣기만 하면 문서 객체가 생성됩니다.

코드 15-33 문서 객체 생성

```
<script>
$(document).ready(function () {
    $('<h1></h1>');
});
</script>
```

생성한 문서 객체는 지금까지 배운 문서 객체 모델 조작 메서드로 코드 15-34처럼 내용을 추가할 수 있습니다. 코드 15-34에서는 `html()` 메서드만 사용했지만 `css()` 메서드나 `메서드`를 사용할 수도 있습니다.

코드 15-34 문서 객체 생성 및 텍스트 노드 추가

```
<script>
$(document).ready(function () {
    $('<h1></h1>').html('Hello World .. !');
});
</script>
```

생성한 문서 객체를 `body` 태그에 추가하려면 코드 15-35처럼 `appendTo()` 메서드를 사용합니다. 이 메서드는 곧 살펴보겠습니다. 우선 지금은 사용만 합시다.

코드 15-35 문서 객체 연결

```
<script>
$(document).ready(function () {
    $('<h1></h1>').html('Hello World .. !').appendTo('body');
});
</script>
```

코드를 실행하면 그림 15-19처럼 출력합니다. jQuery를 사용하면 문서 객체를 동적으로 생성하는 방법이 굉장히 쉽습니다.

그림 15-19 텍스트 노드를 갖는 문서 객체 생성

Hello World .. !

`$()` 메서드의 내부에는 HTML 태그를 넣는다고 했으므로 코드 15-36처럼 내용을 바로 넣을 수도 있습니다. 두 방법 모두 많이 사용되므로 기억해주세요.

코드 15-36 간단한 문서 객체 생성

```
<script>
    $(document).ready(function () {
        $('<h1>Hello World .. !</h1>').appendTo('body');
    });
</script>
```

다음 절로 넘어가기 전에 img 태그를 동적으로 생성해 body 태그에 추가해보세요. 무리 없이 해낼 수 있을 것입니다.

15.12 문서 객체 생성(2)

이번에는 텍스트 노드를 갖지 않는 문서 객체를 생성하는 방법을 알아봅니다. 물론 앞 절의 내용만 봐도 쉽게 해낼 수 있을 것입니다. img 태그를 생성할 때는 코드 15-37처럼 `$()` 메서드로 문서 객체를 생성하고 `attr()` 메서드로 속성을 입력하면 됩니다.

코드 15-37 문서 객체 생성과 속성 지정

```
<script>
    $(document).ready(function () {
        $('<img />').attr('src', 'Chrysanthemum.jpg').appendTo('body');
    });
</script>
```

`attr()` 메서드를 사용하지 않아도 코드 15-38처럼 간단한 방법으로 문서 객체를 생성합니다. `$()` 메서드의 두 번째 매개변수에 객체를 넣고 원하는 속성을 입력하면 됩니다.

코드 15-38 문서 객체 속성 지정

```
<script>
$(document).ready(function () {
    $('<img />', {
        src: 'Chrysanthemum.jpg',
        width: 350,
        height: 250
    }).appendTo('body');
});
</script>
```

코드를 실행하면 그림 15-20처럼 문서 객체가 생성됩니다.

그림 15-20 속성을 갖는 문서 객체 생성

```
▼<body>

</body>
```

그림 15-20의 결과를 보면 width 속성과 height 속성을 입력했는데 스타일 속성으로 입력됐습니다. jQuery의 개발자는 img 태그의 크기는 일반 속성으로 적용하는 것보다 스타일 속성으로 적용하는 것이 더 합리적이라고 생각해서 이렇게 만들었습니다.

15.13 문서 객체 삽입(1)

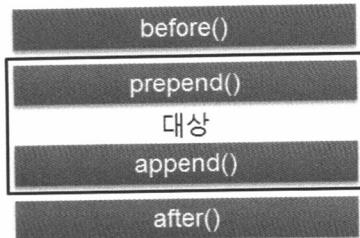
jQuery에는 문서 객체에 문서 객체를 추가하는 메서드가 여덟 개 있습니다. 여덟 개의 메서드는 크게 두 가지 형태로 나눌 수 있습니다. 우선 지금까지 몇 번 사용해본 appendTo() 메서드 같은 형태의 메서드를 표 15-9에서 볼 수 있습니다.

표 15-9 문서 객체 삽입 메서드(1)

메서드 이름	설명
\$(A).appendTo(B)	A를 B의 뒷부분에 추가합니다.
\$(A).prependTo(B)	A를 B의 앞부분에 추가합니다.
\$(A).insertAfter(B)	A를 B의 뒤에 추가합니다.
\$(A).insertBefore(B)	A를 B의 앞에 추가합니다.

append, prepend, after, before는 그림 15-21의 의미가 있습니다. 앞 절에서 사용한 appendTo() 메서드는 대상의 뒷부분에 문서 객체를 삽입한 것입니다.

그림 15-21 삽입 메서드 정리



15.14 문서 객체 삽입(2)

문서 객체를 추가하는 메서드는 두 가지 형태가 있다고 했죠? 이전 절에서 설명한 메서드와 반대의 순서로 문서 객체를 추가하는 표 15-10의 메서드도 있습니다.

표 15-10 문서 객체 삽입 메서드(2)

메서드 이름	설명
\$(A).append(B)	B를 A의 뒷부분에 추가합니다.
\$(A).prepend(B)	B를 A의 앞부분에 추가합니다.
\$(A).after(B)	B를 A의 뒤에 추가합니다.
\$(A).before(B)	B를 A의 앞에 추가합니다.

모든 메서드의 사용 방법이 같으므로 이 절에서는 append() 메서드를 중심으로 알아봅니다. 이 메서드는 다음 형태로 사용합니다.

- 1 \$(selector).append(content, content, , content)
- 2 \$(selector).append(function (index) { });

1번 형태는 content 위치에 문자열이 들어갈 수도 있고 jQuery 문서 객체도 들어갈 수 있습니다. 문서 객체가 들어가는 경우는 다음 절에 살펴봅니다. 예제를 만들어 살펴봅시다. body 태그는 코드 15-39처럼 아무것도 없는 상태로 구성해주세요.

코드 15-39 body 태그 구성

```
<body>
```

```
</body>
```

첫 번째 형태는 코드 15-40처럼 여러 개의 문서 객체를 한꺼번에 입력할 수 있습니다.

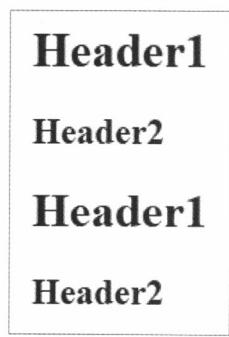
코드 15-40 append() 메서드

```
<script>
$(document).ready(function () {
    // 변수를 선언합니다.
    var h1 = '<h1>Header1</h1>';
    var h2 = '<h2>Header2</h2>';

    // 문서 객체를 추가합니다.
    $('body').append(h1, h2, h1, h2);
});
</script>
```

입력한 문자열 모두 문서 객체를 형성해 그림 15-22처럼 출력합니다.

그림 15-22 append() 메서드를 사용한 문서 객체 추가



두 번째 형태는 선택한 문서 객체에 서로 다른 내용을 추가할 수 있습니다. body 태그를 코드 15-41처럼 구성합니다. 각각의 div 태그에 서로 다른 내용물을 넣어볼 것입니다.

코드 15-41 body 태그 구성

```
<body>
  <div></div>
  <div></div>
  <div></div>
</body>
```

2번 형태를 사용해봅시다. 코드 15-42처럼 append() 메서드의 매개변수에 index 매개변수를 갖는 함수를 넣어줍니다. 각각의 div 태그에 다른 내용을 쉽게 입력하려고 배열을 선언했습니다.

코드 15-42 append() 메서드

```
<script>
$(document).ready(function () {
  // 변수를 선언합니다.
  var content = [
    { name: '윤인성', region: '서울특별시 강서구' },
    { name: '윤하린', region: '서울특별시 광진구' },
    { name: '윤인아', region: '미국 메사추세츠' }
  ];

  // 문서 객체를 추가합니다.
  $('div').append(function (index) {
    });

  });
</script>
```

append() 메서드 부분에 코드 15-43처럼 문서 객체를 만들어 리턴하면 각 div 태그에 서로 다른 문서 객체가 추가됩니다.

코드 15-43 배열을 사용한 문서 객체 생성과 추가

```
// 문서 객체를 추가합니다.
$('div').append(function (index) {
  // 변수를 선언합니다.
  var item = content[index];
  var output = '';
```

```
        output += '<h1>' + item.name + '</h1>';
        output += '<h2>' + item.region + '</h2>';

    return output;
});
```

코드를 실행하면 그림 15-23처럼 각 div 태그에 서로 다른 문서 객체가 들어간 것을 확인할 수 있습니다. 물론 이번 예제는 html() 메서드로도 만들 수 있습니다.

그림 15-23 배열과 append() 메서드를 사용한 문서 객체 추가

```
▼<body>
  ▼<div>
    <h1>윤민성</h1>
    <h2>서울특별시 강서구</h2>
  </div>
  ▼<div>
    <h1>윤하린</h1>
    <h2>서울특별시 광진구</h2>
  </div>
  ▼<div>
    <h1>윤민아</h1>
    <h2>미국 메사추세츠</h2>
  </div>
</body>
```

15.15 문서 객체 이동

기존 문서 객체를 선택하고 문서 객체 삽입 메서드를 사용하면 문서 객체를 쉽게 다른 곳으로 이동시킬 수 있습니다. 시간에 따라 이미지의 순서를 지속적으로 변경하는 간단한 예제를 만들며 익혀봅시다. 우선 body 태그를 구성해주세요.

코드 15-44 body 태그 구성

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

코드 15-45처럼 기존의 문서 객체를 선택하고 문서 객체의 삽입 메서드를 사용합니다. 이렇게 하면 img 태그 중 첫 번째 문서 객체를 body 태그의 뒷부분으로 이동시킵니다.

코드 15-45 appendTo() 메서드를 사용한 문서 객체의 이동

```
<script>
$(document).ready(function () {
    $('img').first().appendTo('body');
});
</script>
```

이를 응용하면 코드 15-46처럼 setInterval() 함수를 사용하면 2초마다 문서 객체의 순서를 변경하는 예제를 만들 수 있습니다.

코드 15-46 지속적인 이미지 순서 변경

```
<script>
$(document).ready(function () {
    // .image의 크기를 조정합니다.
    $('img').css('width', 250);

    // 함수를 2초마다 실행합니다.
    setInterval(function () {
        // 첫 번째 이미지를 마지막으로 보냅니다.
        $('img').first().appendTo('body');
    }, 2000);
});
</script>
```

코드를 실행하면 첫 번째 이미지가 계속 body 태그의 뒷부분으로 옮겨지므로 이미지의 순서가 계속 변경됩니다.

그림 15-24 첫 번째 이미지가 2초마다 맨 뒤로 이동



15.16 문서 객체 복제

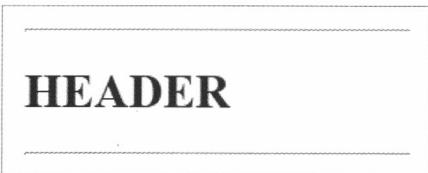
문서 객체를 생성하지 않고 기존의 문서 객체를 선택하고 문서 객체 삽입 메서드를 사용하면 문서 객체가 이동합니다. 코드 15-47의 실행 결과를 예측해봅시다.

코드 15-47 append() 메서드를 사용한 문서 객체의 이동

```
<!DOCTYPE html>
<html>
<head>
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script>
$(document).ready(function () {
    $('div').append($('h1'));
});
</script>
</head>
<body>
<h1>HEADER</h1>
<hr/><div></div><hr/>
</body>
</html>
```

기존 문서 객체를 선택하고 문서 객체 삽입 메서드를 사용했으므로 그림 15-25처럼 문서 객체가 이동하는 것을 확인할 수 있습니다.

그림 15-25 문서 객체가 이동



문서 객체를 복제해서 추가하고 싶을 때는 표 15-11의 메서드를 사용해야 합니다.

표 15-11 문서 객체 복제 메서드

메서드 이름	설명
clone()	문서 객체를 복제합니다.

jQuery의 `clone()` 메서드를 사용하면 쉽게 문서 객체를 복제할 수 있습니다. `clone()` 메서드는 다음 형태로 사용합니다.

```
1 $(selector).clone()
2 $(selector).clone(Boolean dataAndEvents)
3 $(selector).clone(Boolean dataAndEvents, Boolean deepDataAndEvents)
```

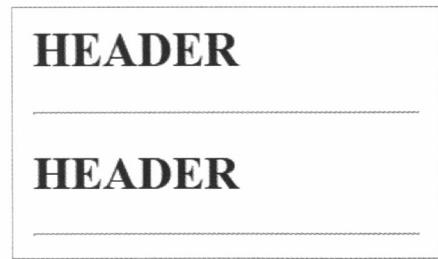
아직 이벤트를 배우지 않았으므로 모든 `clone()` 메서드의 형태를 살펴볼 수 없습니다. 지금은 가장 간단한 1번 형태의 `clone()` 메서드만 살펴봅니다. 코드 15-47의 `script` 태그를 코드 15-48처럼 수정해봅시다. 이번에는 `h1` 태그를 선택해 복제하여 삽입했습니다.

코드 15-48 `clone()` 메서드

```
<script>
$(document).ready(function () {
    $('div').append($('h1').clone());
});
</script>
```

코드를 실행하면 그림 15-26처럼 문서 객체가 복제된 것을 확인할 수 있습니다.

그림 15-26 문서 객체를 복제하여 삽입



15장에서는 jQuery의 문서 객체와 관련된 내용을 조금 자세히 살펴보았습니다. jQuery 자체가 기존의 자바스크립트보다 굉장히 쉬워 무리 없이 진행했으리라 생각합니다.