

문서 객체 선택과 탐색

14장에서는 선택자와 관련된 추가 메서드를 살펴봅니다. 사실 이 책의 이후 내용을 살펴봐도 코드가 워낙 간단하므로 이 장에서 살펴보는 수준의 선택자는 거의 사용할 필요가 없습니다. 하지만, 큰 규모의 웹 페이지에 jQuery를 적용할 때는 이 장의 내용을 어느 정도 확실하게 알고 있어야 합니다.

CSS를 이전 장에서 처음 살펴본 독자라면 이 장의 내용이 어려울 수 있습니다. 이 장의 내용을 진행하다가 너무 어렵다는 생각이 든다면 일단 다음 장으로 바로 넘어가세요. 2부를 모두 마친 후에 돌아와도 문제 없는 부분입니다.

14.1 기본 필터 메서드

jQuery의 선택자를 사용하면 원하는 문서 객체를 대부분 선택할 수 있습니다. 기본적으로 지원하지 않는 필터로 문서 객체를 선택해야 한다면 표 14-1의 메서드를 사용해야 합니다. `filter()` 메서드를 사용해야 하는 이유는 다음 절에서 살펴봅니다.

표 14-1 기본 필터 메서드

메서드 이름	설명
<code>filter()</code>	문서 객체를 필터링합니다.

`filter()` 메서드는 다음 두 가지 형태로 사용합니다.

- 1 `$(selector).filter(selector);`
- 2 `$(selector).filter(function () { });`

`filter()` 메서드를 사용할 수 있는 간단한 예제를 만들어봅시다. `body` 태그를 코드 14-1처럼 구성합니다.

코드 14-1 `body` 태그 구성

```
<body>
  <h3>Header-0</h3>
  <h3>Header-1</h3>
  <h3>Header-2</h3>
  <h3>Header-3</h3>
  <h3>Header-4</h3>
  <h3>Header-5</h3>
</body>
```

현재 `h3` 태그에서 홀수 번째에 위치하는 문서 객체를 선택하려면 코드 14-2처럼 선택자를 사용합니다. 코드 14-2는 홀수 번째에 위치하는 `h3` 태그의 스타일 속성을 변경합니다.

코드 14-2 필터 선택자

```
<script>
$(document).ready(function () {
  $('h3:even').css({
    backgroundColor: 'black',
    color: 'white'
  });
});
</script>
```

이러한 기능을 1번 형태의 `filter()` 메서드로도 수행할 수 있습니다. 코드 14-3처럼 `filter()` 메서드의 매개변수에 선택자를 입력합니다. 코드 14-3처럼 ':even'을 입력할 수도 있습니다. 홀수 번째에 위치하는 `h3` 태그를 선택해 스타일을 적용합니다.

코드 14-3 filter() 메서드

```
<script>
$(document).ready(function () {
    $('h3').filter(':even').css({
        backgroundColor: 'black',
        color: 'white'
    });
});
</script>
```

코드 14-3을 실행하면 코드 14-2와 마찬가지로 홀수 번째에 위치하는 h3 태그를 선택해 스타일을 변경합니다.

“이럴 거면 왜 써요?”

왜 쓰는지는 다음 절에서 알아봅시다. 이어서 2번 형태로 filter() 메서드를 사용해봅시다. filter() 메서드의 매개변수에 함수를 넣습니다. 이때 입력하는 함수는 매개변수로 index를 갖습니다. 함수에서 리턴하는 값에 따라 문서 객체를 선택합니다.

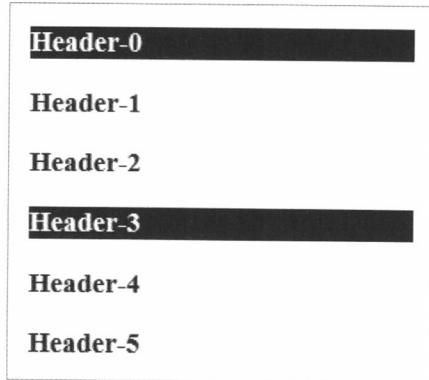
코드 14-4는 filter() 메서드의 2번 형태로 매개변수 index가 3의 배수인 h3 태그를 선택합니다.

코드 14-4 filter() 메서드의 매개변수로 함수를 넣을 경우

```
<script>
$(document).ready(function () {
    $('h3').filter(function (index) {
        return index % 3 == 0;
    }).css({
        backgroundColor: 'black',
        color: 'white'
    });
});
</script>
```

코드를 실행하면 그림 14-1처럼 출력됩니다.

그림 14-1 filter() 메서드



14.2 문서 객체 탐색 종료

앞 절에서 간단하게 filter() 메서드를 살펴보았습니다. 2번 형태는 왜 쓰는지 이해할 수 있지만, 1번 형태는 도대체 왜 쓰는지 짐작하기 힘듭니다. 코드 14-5의 실행 결과를 예측해봅시다.

코드 14-5 HTML 페이지 구성

```
<!DOCTYPE html>
<html>
<head>
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script>
$(document).ready(function () {
    $('h1').css('background', 'orange').filter(':even').css('color', 'red');
});
</script>
</head>
<body>
<h1>Header-0</h1>
<h1>Header-1</h1>
<h1>Header-2</h1>
</body>
</html>
```

체이닝을 사용해 한 줄로 서로 다른 문서 객체에 스타일을 적용했습니다. 이렇게 사용하면 코드가 간결해지므로 filter() 메서드를 사용합니다. 하지만, 이 방법은 점점 문서 객체의 범위를 좁게만 선택할 수 있습니다. 예를 들어 코드 14-6을 체이닝하려면 어떻게 해야 할까요?

코드 14-6 연습

```
<script>
    $(document).ready(function () {
        $('h1').css('background', 'orange');
        $('h1:even').css('color', 'white');
        $('h1:odd').css('color', 'red');
    });
</script>
```

지금까지 배운 방법으로는 체이닝을 사용하는 형태로 바꿀 수 없습니다. 체이닝을 사용할 때 추가한 filter() 메서드를 제거하려면 표 14-2의 메서드를 사용합니다.

표 14-2 문서 객체 탐색 종료 메서드

메서드 이름	설명
end()	문서 객체 선택을 한 단계 뒤로 돌립니다.

end() 메서드를 사용하면 코드 14-6을 코드 14-7의 형태로 사용할 수 있습니다.

코드 14-7 end() 메서드를 사용한 체이닝

```
$( 'h1' ).css( 'background', 'orange' ).filter( ':even' ).css( 'color', 'white' ).end().
filter( ':odd' ).css( 'color', 'red' );
```

end() 메서드는 filter() 메서드뿐만 아니라 지금부터 배울 메서드에 모두 적용할 수 있습니다. end() 메서드와 관련된 예제는 이후의 메서드를 배우면서 살펴보겠습니다.

14.3 특정 위치의 문서 객체 선택

필터 선택자를 이용하면 특정 위치에 존재하는 문서 객체를 선택할 수 있습니다. 하지만, 특정 위치에 존재하는 문서 객체를 선택하는 필터 선택자는 자주 사용하므로 표 14-3처럼 간단한 메서드로 제공해줍니다.

표 14-3 특정 위치의 문서 객체 선택 메서드

메서드 이름	설명
eq()	특정 위치에 존재하는 문서 객체를 선택합니다.
first()	첫 번째에 위치하는 문서 객체를 선택합니다.
last()	마지막에 위치하는 문서 객체를 선택합니다.

이 메서드들을 간단하게 사용해봅시다. 코드 14-8처럼 body 태그를 구성해주세요.

코드 14-8 body 태그 구성

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

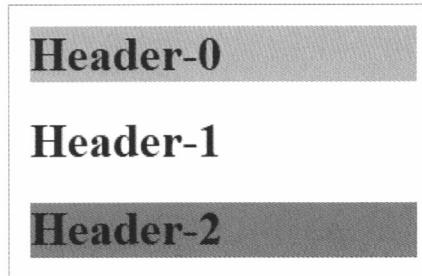
eq() 메서드는 매개변수에 숫자를 입력합니다. 이때 양수를 입력할 수도 있지만, 음수를 입력할 수도 있습니다. 음수를 입력하면 뒤쪽을 기준으로 선택합니다. 코드 14-9의 출력 결과를 예측해보세요.

코드 14-9 eq() 메서드

```
<script>
$(document).ready(function () {
  $('h1').eq(0).css('background', 'orange');
  $('h1').eq(-1).css('background', 'red');
});
</script>
```

코드를 실행하면 그림 14-2처럼 출력합니다.

그림 14-2 eq() 메서드



first() 메서드와 last() 메서드는 별도의 설명이 없어도 사용할 수 있을 겁니다.

14.4 문서 객체 추가 선택

지금까지 살펴본 메서드는 선택자의 범위를 점점 축소하는 메서드입니다. jQuery는 문서 객체의 체이닝을 더 유연하게 하려고 표 14-4의 메서드를 제공합니다. add() 메서드를 사용하면 현재 선택한 문서 객체의 범위를 확장할 수 있습니다.

표 14-4 문서 객체 추가 선택 메서드

메서드 이름	설명
add()	문서 객체를 추가적으로 선택합니다.

body 태그를 코드 14-10처럼 구성해주세요.

코드 14-10 body 태그 구성

```
<body>
  <h1>Header-0</h1>
  <h2>Header-1</h2>
  <h1>Header-2</h1>
  <h2>Header-3</h2>
  <h1>Header-4</h1>
</body>
```

`add()` 메서드는 코드 14-11처럼 사용할 수 있습니다. 코드 14-11은 `h1` 태그의 `background` 스타일 속성에 `Gray`를 입력하고, `h2` 태그를 추가로 선택해 `float` 스타일 속성에 `left`를 입력합니다.

코드 14-11 `add()` 메서드

```
<script>
$(document).ready(function () {
    $('h1').css('background', 'Gray').add('h2').css('float', 'left');
});
</script>
```

코드를 실행하면 그림 14-3처럼 스타일 속성이 적용된 것을 확인할 수 있습니다.

그림 14-3 `add()` 메서드



14.5 문서 객체의 특징 판별

문서 객체가 특징이 있는지 판단할 때 표 14-5의 메서드를 사용합니다.

표 14-5 문서 객체의 특징 판별 메서드

메서드 이름	설명
<code>is()</code>	문서 객체의 특징을 판별합니다.

아직 이벤트와 관련된 내용을 배우지 않았으므로 `is()` 메서드를 활용할 곳이 마땅히 없습니다. 간단하게 어떻게 사용하는 메서드인지만 살펴봅시다. 코드 14-12처럼 `body` 태그를 구성합니다.

코드 14-12 body 태그 구성

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

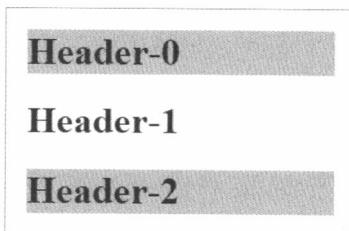
`is()` 메서드는 매개변수로 선택자를 입력합니다. 선택한 객체가 선택자와 일치하는지 판별해 불을 리턴합니다. 코드 14-13은 `h1` 태그 중 `select` 클래스 속성이 있는 문서 객체에 스타일을 적용합니다. 사실, 이렇게 쓸 필요는 당연히 없습니다. 연습용 예제이므로 살펴본 것입니다.

코드 14-13 `is()` 메서드

```
<script>
$(document).ready(function () {
  $('h1').each(function () {
    if ($(this).is('.select')) {
      $(this).css('background', 'orange');
    }
  });
});
</script>
```

코드를 실행하면 그림 14-4처럼 출력됩니다.

그림 14-4 `is()` 메서드



14.6 특정 태그 선택

문서 객체에서 특정 태그를 선택하는 방법을 알아봅시다. 물론 일반 선택자로 선택할 수도 있지만, 이 절에서 배울 메서드는 XML 문서에서 데이터를 추출하는 데 많이 사용하므로 꼭 기억해주세요. 특정 태그를 선택할 때는 표 14-6의 메서드를 사용합니다.

표 14-6 특정 태그 선택 메서드

메서드 이름	설명
find()	특정 태그를 선택합니다.

XML 문서에서 데이터를 추출하는 예제를 작성하며 find() 메서드를 배워봅시다. body 태그는 코드 14-14처럼 아무것도 입력하지 말아주세요.

코드 14-14 body 태그 구성

```
<body>
```

```
</body>
```

코드 14-15처럼 문자열에 XML 문서를 입력합니다. 이어서 \$.parseXML() 메서드로 문자열을 XML 문서 객체로 변경합니다. 대부분의 브라우저는 XML 문자열을 바로 사용할 수 있지만 인터넷 익스플로러에서는 문제가 발생합니다.

코드 14-15 XML 문자열 생성

```
<script>
    // 변수를 선언합니다.
    var xml = '';
    xml += '<friends>';
    xml += '    <friend>';
    xml += '        <name>연하진</name>';
    xml += '        <language>Ruby</language>';
    xml += '    </friend>';
    xml += '    <friend>';
    xml += '        <name>윤명월</name>';
    xml += '        <language>Basic</language>';
    xml += '    </friend>';

```

```
xml += '<friend>';
xml += '    <name>윤하린</name>';
xml += '    <language>C#</language>';
xml += '</friend>';
xml += '</friends>';

$(document).ready(function () {
    // 변수를 선언합니다.
    var xmlDoc = $.parseXML(xml);
});
</script>
```

지금은 XML 문서를 문자열로 직접 만들지만, 3부에서는 Ajax로 서버에서 XML 문서를 가져옵니다.

코드 14-16처럼 \$() 메서드의 매개변수에 생성한 XML 문서 객체를 입력합니다. \$() 메서드에는 문서 객체를 곧바로 넣을 수도 있습니다. 지금까지 사용했던 \$(document) 형태를 생각하면 이해하기 쉽습니다. 이는 15장에서 더 자세히 살펴봅니다. 코드 14-16처럼 find() 메서드로 friend 태그를 선택합니다. friend 태그는 총 세 개이므로 each() 메서드를 사용했습니다.

코드 14-16 \$.parseXML() 메서드와 each() 메서드

```
$(document).ready(function () {
    // 변수를 선언합니다.
    var xmlDoc = $.parseXML(xml);
    $(xmlDoc).find('friend').each(function (index) {

    });
});
```

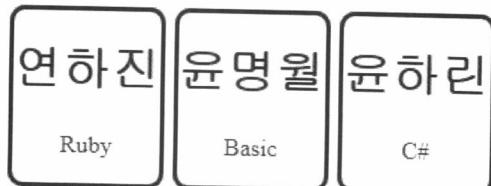
코드 14-17처럼 each() 메서드 안에서 find() 메서드를 한 번 더 사용해 각각의 데이터를 추출합니다. text() 메서드를 사용하면 태그 내부의 글자를 가져올 수 있습니다. 각각의 데이터로 HTML 태그를 만든 후에 innerHTML 속성에 추가합니다.

코드 14-17 XML 파싱

```
$(document).ready(function () {  
    // 변수를 선언합니다.  
    var xmlDoc = $.parseXML(xml);  
    $(xmlDoc).find('friend').each(function (index) {  
        // 변수를 선언합니다.  
        var output = '';  
        output += '<div>';  
        output += '    <h1>' + $(this).find('name').text() + '</h1>';  
        output += '    <p>' + $(this).find('language').text() + '</p>';  
        output += '</div>';  
  
        // 출력합니다.  
        document.body.innerHTML += output;  
    });  
});
```

아직 jQuery에서 `innerHTML` 속성과 관련된 메서드를 배우지 않았으므로 `innerHTML` 속성을 사용했습니다. 스타일 관련 부분을 약간 조작하고 코드를 실행하면 그림 14-5와 같은 출력 결과를 얻을 수 있습니다.

그림 14-5 XML 파싱과 문서 객체 생성



14장의 내용이 조금 깊이 와닿지 않을 것입니다. 모두 외울 필요 없이 “이런 것이 있구나” 정도로 기억하고 다음 장으로 넘어갑시다.