

Part III

jQuery

지금까지 자바스크립트와 관련된 내용을 살펴보았습니다. 쉬운 부분도 있었겠지만, 어려운 부분도 많았을 것입니다. jQuery는 자바스크립트를 더 쉽게 사용하고자 만들어진 자바스크립트 프레임워크입니다. 1부를 마친 독자라면 2부를 쉽게 진행할 수 있습니다.

2부에서는 jQuery로 문서 객체와 이벤트를 다루는 방법을 배웁니다. 또한 애니메이션 효과로 동적인 요소를 만드는 방법도 배웁니다.

기본

본격적으로 jQuery를 공부해봅시다. 이 책의 서문에서 언급했듯이 이 책의 독자는 HTML과 CSS의 기본적인 내용을 알고 있는 사람입니다. 하지만, 일부 CSS를 모르는 독자도 있을 것입니다. 13장에서는 가장 먼저 jQuery의 가장 중요한 구성 요소인 CSS 선택자를 정리합니다. 이어서 jQuery와 관련된 부가적인 내용을 살펴봅니다.

13.1 개요

jQuery는 모든 브라우저에서 동작하는 클라이언트 자바스크립트 라이브러리입니다. 2006년 1월, 존 레시[John Resig]@ BarCamp NYC에서 발표했으며, 무료로 사용 가능한 오픈 소스 라이브러리입니다. jQuery는 다음 기능을 위해 제작됐습니다.

- 문서 객체 모델과 관련된 처리를 쉽게 구현
- 일관된 이벤트 연결을 쉽게 구현
- 시각적 효과를 쉽게 구현
- Ajax 애플리케이션을 쉽게 개발

기능에 모두 ‘쉽게’라는 말이 들어 있습니다. 정말 쉽습니다. 1부에 100의 에너지를 쏟았다고 하면 2부는 30정도의 에너지만 쏟으면 됩니다.

빌트위드 Builtwith의 통계에 따르면 2013년 5월을 기준으로 전 세계 사람이 가장 많이 방문하는 웹 사이트 10,000개 중 약 67%의 웹 사이트가 jQuery를 사용합니다. 구글과 마이크로소프트는 현재 jQuery CDN 호스트를 제공하며, 마이크로소프트는 ASP.NET 프레임워크와 ASP.NET MVC 프레임워크에 jQuery를 채택해 사용합니다.

국내 사례로는 티켓몬스터, 쿠팡, 그루폰, 위메이크프라이스를 포함한 모든 소셜커머스 사이트에서 jQuery를 활용합니다. 기업 홈페이지로는 CJ제일제당, LG 등의 사이트에서 사용됩니다. 대형 포털 사이트는 대부분 자체 개발한 라이브러리를 사용하지만, 일부는 jQuery를 사용합니다.

jQuery를 사용하면 웹 표준만으로도 플래시와 실버라이트로 구현한 웹 사이트와 비슷한 수준의 시각적 효과를 구현할 수 있습니다. jQuery로 시각적 효과를 구현하면 플래시와 실버라이트를 작동하지 않는 아이폰, 아이패드 같은 장치에서도 작동합니다. 그럼 13-1은 CJ제일제당의 사이트입니다. CJ제일제당 사이트는 웹 표준만으로 구현했으므로 아이패드에서도 정상적으로 구동됩니다.

그림 13-1 CJ제일제당 사이트

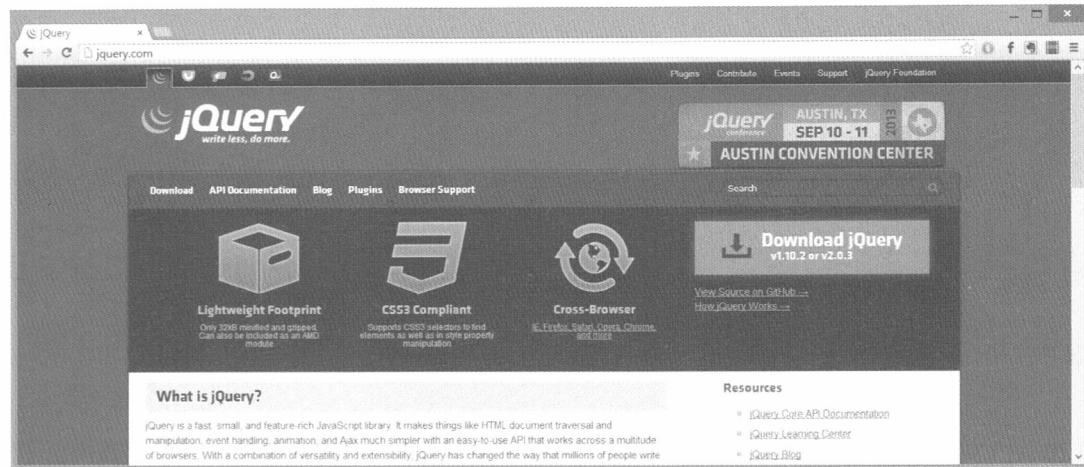


또한 브라우저의 차이로 발생하는 문제를 jQuery를 사용하면 쉽게 해결할 수 있습니다. 예를 들어 1부 이벤트 부분을 살펴보면서 브라우저마다 다른 이벤트 표준으로 인해 굉장히 복잡한 방법으로 이벤트를 연결했습니다. jQuery를 사용한다면 모든 브라우저에서 동작하는 이벤트에 쉽게 연결할 수 있습니다.

13.2 다운로드

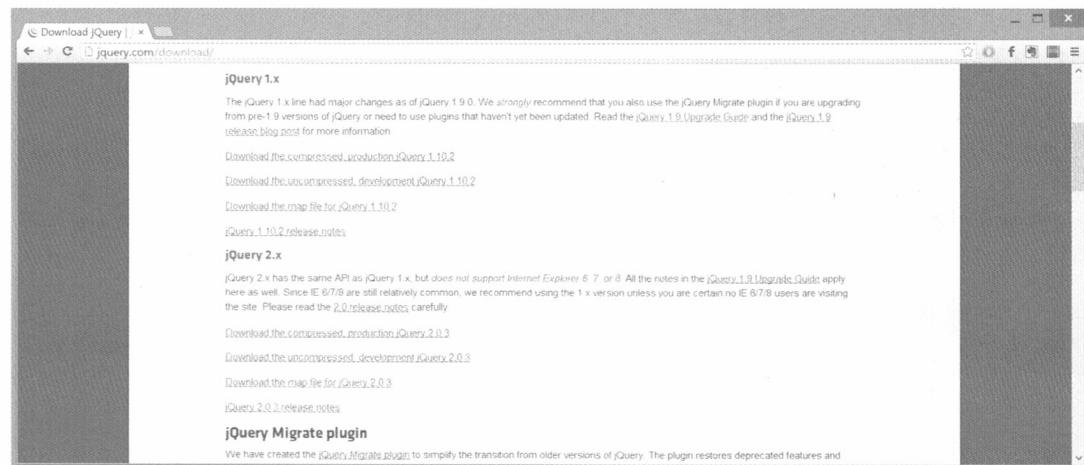
jQuery를 공부하려면 당연히 jQuery가 필요합니다. jQuery를 내려받으려면 <http://jquery.com>에 접속합니다. 메인 화면에서 다운로드 버튼을 누르면 다운로드 페이지로 이동합니다.

그림 13-2 <http://jquery.com/>



다운로드 페이지에 jQuery 1.X와 jQuery 2.X의 2가지 버전이 있지만 1.X 버전으로 내려받습니다.

그림 13-3 다운로드 페이지



NOTE_ jQuery 1.X 버전과 jQuery 2.X 버전

2013년에 jQuery 2.0 버전을 발표하면서 인터넷 익스플로러 8 버전 이하를 포기했습니다.

“그럼 어떻게 해요?”

jQuery 1.X 버전을 사용하면 됩니다. jQuery 재단은 인터넷 익스플로러 8 버전 이하가 거의 사라질 때 까지 jQuery 1.X 버전의 업그레이드 지원을 약속했습니다.

“차이는 뭔가요?”

jQuery 2.0 버전부터는 HTML5에서 추가된 새로운 기능을 지원합니다. 또한 jQuery 1.X 버전에서 사용하던 Sizzle 엔진 부분을 제거해 용량이 많이 줄었습니다. 그러므로 인터넷 익스플로러 8 버전 이하에서는 jQuery 2.0 버전에서 추가된 기능 자체를 사용할 수 없습니다.

jQuery 2.0 버전에서 추가된 기능은 부록에서 정리합니다. 본문을 진행할 때는 jQuery 1.X 버전을 사용해주세요.

jQuery는 두 가지 방법으로 사용할 수 있습니다. 첫 번째 방법은 CDN 호스트를 사용하는 방법이고, 두 번째는 직접 내려받아 사용하는 방법입니다. 두 가지 방법을 모두 사용해도 됩니다. 편리하게 사용하려면 CDN 호스트를 사용하고 처음 공부한다면 내려받는 것을 추천합니다.

우선 CDN 호스트를 사용하는 방법을 알아보겠습니다. 다운로드 페이지에 그림 13-4와 같은 부분이 있습니다.

그림 13-4 jQuery의 CDN 호스트 주소

Using jQuery with a CDN

CDNs can offer a performance benefit by hosting jQuery on servers spread across the globe. This also offers an advantage that if the visitor to your webpage has already downloaded a copy of jQuery from the same CDN, it won't have to be re-downloaded.

jQuery's CDN provided by MediaTemple

To use the jQuery CDN, just reference the file directly from <http://code.jquery.com> in the script tag:

```
1 | <script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
2 | <script src="http://code.jquery.com/jquery-migrate-1.2.1.min.js"></script>
```

구글과 마이크로소프트에서 CDN 호스트를 지원합니다. CDN^{Content Delivery Network}은 사용자에게 간편하게 콘텐츠를 제공하는 방식을 의미합니다. 구글, 마이크로소프트, jQuery측에서 사용자가 jQuery를 사용하기 편하게 콘텐츠를 제공하는 것입니다.

jQuery의 CDN 호스트를 사용하려면 다음과 같이 HTML 페이지를 구성합니다. script 태그의 src 속성에 제공되는 CDN 호스트를 입력합니다.

코드 13-1 CDN 호스트 사용

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

</script>
</head>
<body>

</body>
</html>
```

그림 13-4의 호스트 이외에 다음과 같은 CDN 호스트도 있습니다. 이는 jQuery 공식 홈페이지에서 확인할 수 있습니다.

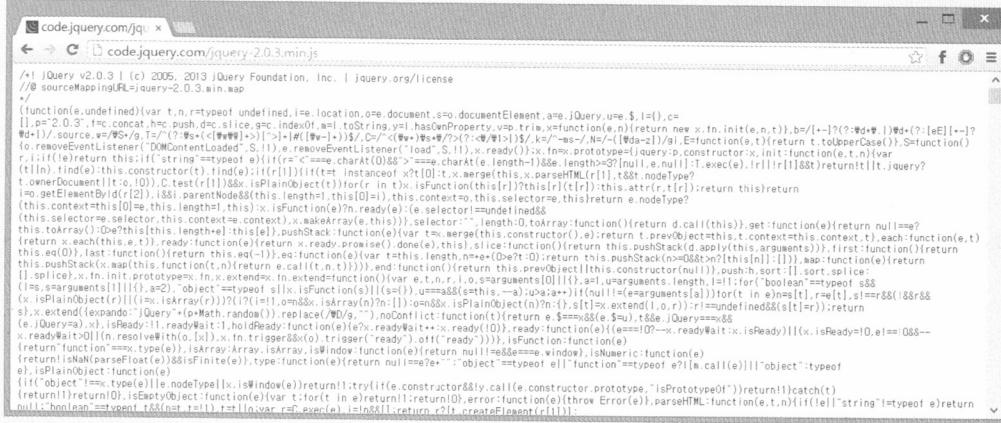
- <http://code.jquery.com/jquery-1.10.2.js>
- <http://code.jquery.com/jquery-1.10.2.min.js>
- <https://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.js>
- <https://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js>
- <http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.10.2.js>
- <http://ajax.aspnetcdn.com/ajax/jquery/jquery-1.10.2.min.js>

jQuery 파일명은 ○○.js 파일과 ○○.min.js 파일로 나뉩니다. ○○.js 파일은 Uncompressed 버전이라고 부르며, Minified 버전이라고 부릅니다. 두 파일은 용량이 5배 이상 차이납니다. Minified 버전은 파일의 용량을 최소화하려고 지핑zipping한 파일입니다. 파일을 직접 확인하면 차이를 쉽게 알 수 있습니다. 이 책에서는 Uncompressed 버전을 사용합니다.

NOTE_ 지핑

소스 코드를 확인하면 그림 13-5처럼 모든 코드가 들여쓰기 구분이 없는 것을 확인할 수 있습니다. 이렇게 소스 코드가 모두 붙어 클라이언트에게 제공할 웹 페이지 용량을 줄이는 것을 지핑(Zipping)이라고 합니다.

그림 13-5 지핑



하이브리드 애플리케이션 같은 오프라인 환경에서 jQuery를 사용한다면 반드시 내려받아 사용합니다. jQuery를 내려받아 파일 경로를 같은 방법으로 입력합니다.

코드 13-2 내려받아 사용

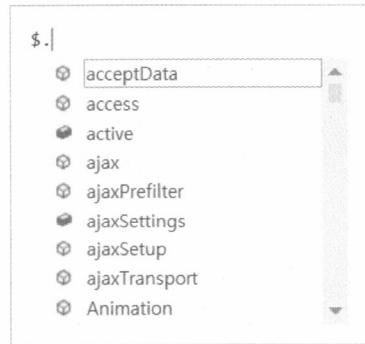
```
<!DOCTYPE html>
<html>
<head>
<script src="jquery-1.10.2.js"></script>
<script>

</script>
</head>
<body>

</body>
</html>
```

이전에 처음 공부할 때는 내려받아 사용하는 것이 좋다고 했죠? 내려받아 사용할 경우는 그림 13-6처럼 보조 기능을 활용할 수 있습니다.

그림 13-6 보조 기능 활용



NOTE_ jQuery 보조 기능

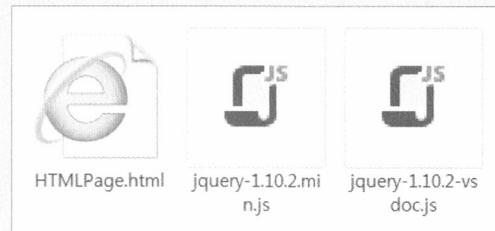
프로그래밍을 여러 번 접해본 독자에게는 이 절에서 설명하는 내용이 2부 전체의 내용이 될 수도 있습니다. 이 절에서 한 번만 사용하는 방법을 알려준 뒤 이후의 절에서는 사용하지 않습니다. 필요 없다면 사용하지 않아도 되지만 사용하는 것을 추천합니다.

마이크로소프트의 Visual Studio 제품군은 jQuery를 위한 보조 기능을 제공합니다. 우리가 지금 사용하는 Visual Studio 2012 Express for Web도 Visual Studio 제품군이므로 보조 기능을 사용할 수 있습니다.

<http://Ajax.aspnetcdn.com/ajax/jQuery/jquery-1.10.2-vsdoc.js>는 마이크로소프트에서 제공하는 jQuery 보조 파일입니다. 이 자바스크립트 파일과 관련된 내용은 <http://www.asp.net/ajaxlibrary/cdn.ashx>를 참고하세요.

보조 기능을 사용하려면 jQuery를 내려받아야 합니다. jQuery와 보조 파일을 내려받아 그림 13-7처럼 폴더를 구성합니다.

그림 13-7 폴더 구성



그리고 다음과 같이 HTML 페이지를 구성합니다.

코드 13-3 HTML 페이지 구성

```
<!DOCTYPE html>
<html>
<head>
  <script src="jquery-1.10.1.min.js"></script>
  <script src="jquery-1.10.2-vsdoc.js"></script>
  <script>

  </script>
</head>
<body>

</body>
</html>
```

이렇게 페이지를 구성하면 그림 13-8처럼 보조 설명을 이용할 수 있습니다.

그림 13-8 자세한 보조 기능

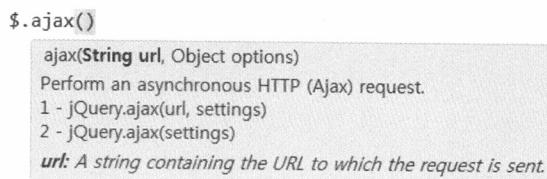


그림 13-8을 보면 `jQuery()` 메서드를 어떠한 방식으로 사용해야 하는지 친절하게 설명합니다. 어떠한 매개변수를 넣고, 어떠한 것을 리턴하는지 쉽게 알 수 있습니다. 이 책의 내용을 진행하면서 직접 `jQuery`의 문서와 이 보조 기능을 함께 사용하면서 공부해주세요.

13.3 \$(document).ready()

`jQuery`를 사용한 모든 웹 페이지는 다음 코드로 시작합니다.

코드 13-4 document 객체의 ready 이벤트 연결

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

`$(document).ready()`는 문서가 준비되면 매개변수로 넣은 콜백 함수를 실행하라는 의미입니다. jQuery 이벤트 메서드 중 하나입니다. 이 메서드는 다음과 비슷한 기능을 수행합니다.

코드 13-5 window 객체의 load 이벤트 연결

```
<script>
window.onload = function () {
    };
</script>
```

고전 이벤트 모델은 한 번에 하나의 이벤트만 연결할 수 있습니다. 반면에 jQuery의 이벤트 메서드는 표준 이벤트 모델이나 인터넷 익스플로러 이벤트 모델과 마찬가지로 이벤트로 여러 개의 함수를 연결할 수 있습니다. 코드 13-6을 실행하면 문서가 준비되는 순간 경고창 세 개가 연달아 표시됩니다.

코드 13-6 복수 개의 이벤트의 연결

```
<script>
$(document).ready(function () {
    alert('First READY');
});

$(document).ready(function () {
    alert('Second READY');
});

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

`$(document).ready()` 메서드는 굉장히 많이 사용되므로 jQuery에서는 간단하게 사용할 수 있는 형태를 제공합니다.

코드 13-7 간단한 형식의 `$(document).ready()`

```
<script>
$(function () {
});
```

이 책에서는 간단한 형태보다는 완전한 형태로 내용을 진행하겠습니다. 이제부터 jQuery에 대한 본격적인 내용을 하나씩 살펴봅시다.

NOTE_ jQuery와 \$

아마 이 절의 코드를 입력하면서 다음과 같은 생각을 했을 것입니다.

“대체 \$는 뭐죠?”

자바스크립트에서 식별자로 사용할 수 있는 특수 기호는 \$와 _입니다. 여기서 \$를 식별자로 사용했을 뿐입니다. jQuery의 코드를 보면 다음과 같은 부분이 있습니다.

```
window.jQuery = window.$ = jQuery;
```

jQuery 식별자를 \$로 대체했을 뿐입니다. 따라서 \$ 식별자가 아니라 jQuery 식별자를 사용해도 됩니다. 하지만 많이 사용하는 식별자이므로 쉽게 \$를 사용하는 것이 일반적입니다.

13.4 기본 선택자

jQuery 메서드의 가장 기본적인 형태는 그림 13-9와 같습니다. 이 형태는 문서 객체를 다룰 때 사용하는 형태이며 jQuery에서 가장 많이 사용하는 형태입니다.

그림 13-9 jQuery 메서드의 기본 형태

\$('hi').css('color', 'red');

선택자는 jQuery에서 가장 중요한 역할을 합니다. jQuery의 선택자는 이미 웹 개발에서 많이 사용되는 CSS 선택자와 유사하므로 CSS를 조금이라도 안다면 jQuery를 쉽게 배울 수 있습니다. 이 장에서는 CSS 선택자를 정리합니다.

13.4.1 전체 선택자

CSS의 가장 기본적인 선택자는 전체 선택자(Wildcard Selector)입니다. HTML 페이지에 있는 모든 문서 객체를 선택하는 선택자입니다. 코드 13-8에 사용한 *를 전체 선택자라고 부릅니다.

코드 13-8 전체 선택자

```
<!DOCTYPE html>
<html>
<head>
    <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
    <script>
        $(document).ready(function () {
            $('*').css('color', 'red');
        });
    </script>
</head>
<body>
    <h1>Lorem ipsum</h1>
</body>
</html>
```

`css()` 메서드는 jQuery의 가장 기본적인 메서드로 첫 번째 매개변수에 바꾸고자 하는 스타일 속성 이름을 입력하고 두 번째 매서드에 스타일 속성 값을 입력합니다. 코드 13-8은 HTML 페이지에 존재하는 모든 문서 객체의 `color` 스타일 속성에 `red`를 입력합니다.

따라서 코드를 실행하면 `h1` 태그가 붉은색으로 표시된 것이 보입니다. 유효성 검사를 사용하면

h1 태그뿐만 아니라 그림 13-10처럼 html, head, script, body 태그까지 스타일이 적용된 것을 알 수 있습니다.

그림 13-10 전체 선택자

```
▼<html style="color: red;">
  ▶<head style="color: red;">...</head>
    ▼<body style="color: red;">
      <h1 style="color: red;">Lorem ipsum</h1>
    </body>
  </html>
```

13.4.2 태그 선택자

태그 선택자는 특정한 태그를 선택하는 선택자입니다. 태그 선택자는 태그의 이름을 그냥 사용합니다. 태그 선택자를 사용하는 방법을 알아보려고 body 태그를 코드 13-9처럼 구성했습니다.

코드 13-9 body 태그 구성

```
<body>
  <h1>Lorem ipsum</h1>
  <p>Lorem ipsum dolor sit amet.</p>
  <h1>Lorem ipsum</h1>
  <p>consectetur adipiscing elit.</p>
</body>
```

태그 선택자는 코드 13-10처럼 사용합니다. 코드 13-10은 HTML 페이지 내의 모든 h1 태그에 스타일을 적용합니다.

코드 13-10 태그 선택자

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

코드를 실행하면 그림 13-11처럼 h1 태그에만 스타일이 적용됩니다.

그림 13-11 태그 선택자

```
▼<body>
  <h1 style="color: orange; ">Lorem ipsum</h1>
  <p>Lorem ipsum dolor sit amet.</p>
  <h1 style="color: orange; ">Lorem ipsum</h1>
  <p>consectetur adipiscing elit.</p>
</body>
```

NOTE_ 여러 개의 태그 선택

하나 이상의 태그 선택자를 동시에 사용하고 싶을 때는 코드 13-11처럼 콤마로 선택자를 구분합니다.
코드 13-11은 h1 태그와 p 태그의 스타일을 모두 변경합니다.

코드 13-11 여러 개의 태그 선택

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

13.4.3 아이디 선택자

아이디 선택자는 특정한 id 속성이 있는 문서 객체를 선택하는 선택자입니다. 코드 13-12처럼 body 태그를 구성해봅시다. 코드 13-12의 두 번째에 위치한 h1 태그가 id 속성으로 target을 가진다는 것을 봐주세요.

코드 13-12 body 태그 구성

```
<body>
  <h1>Header-0</h1>
  <h1 id="target">Header-1</h1>
  <h1>Header-2</h1>
</body>
```

id 선택자는 코드 13-13처럼 #을 이용해 사용합니다. 코드 13-13은 id 속성이 target인 문서 객체에 스타일을 적용합니다.

코드 13-13 아이디 선택자

```
<script>
$(document).ready(function () {
    $('#target').css('color', 'orange');
});
</script>
```

HTML 웹 표준에 따르면 id 속성은 HTML 페이지 내에서 유일한 값을 가져야 합니다. 즉 현재 예제에서 id 속성 target은 무조건 한 개만 존재해야 합니다.

NOTE 2개 이상 있어도 실제 실행에는 문제가 없습니다.

따라서 필요는 없지만 코드 13-14처럼 더 정확하게 h1 태그 중 id 속성이 target인 문서 객체를 선택할 수 있습니다.

코드 13-14 특정 id 속성을 가지는 태그 선택

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

13.4.4 클래스 선택자

클래스 선택자는 특정한 class 속성이 있는 문서 객체를 선택하는 선택자입니다. 코드 13-15를 보면 각각의 h1 태그에 class 속성을 입력했습니다.

코드 13-15 body 태그 구성

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

클래스 선택자 역시 아이디 선택자와 비슷하게 코드 13-16의 형태로 사용합니다.

코드 13-16 클래스 선택자와 특정 class 속성을 가지는 태그 선택

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

두 클래스 속성을 모두 갖는 문서 객체를 선택하고 싶을 때는 코드 13-17처럼 두 클래스 선택자를 붙여서 사용합니다. 코드 13-17은 클래스 속성으로 item과 select가 모두 있는 문서 객체를 선택합니다.

코드 13-17 두 가지 클래스를 모두 가지는 태그 선택

```
<script>
$(document).ready(function () {
    $('.item.select').css('color', 'orange');
});
</script>
```

현재 body 태그에서 두 클래스 속성이 모두 있는 문서 객체는 두 번째에 위치한 h1 태그이므로 그림 13-12처럼 두 번째에 위치한 h1 태그에만 스타일을 적용합니다.

그림 13-12 두 가지 클래스 선택

```
▼<body>
  <h1 class="item">Header-0</h1>
  <h1 class="item select" style="color: orange;">Header-1</h1>
  <h1 class="item">Header-2</h1>
</body>
```

지금 살펴본 네 개의 선택자가 가장 중요한 선택자입니다. 매우 중요한 것에 비해 어려운 내용이 아니므로 꼭 외워주세요. 이 책에서 외우라는 내용이 많지 않았죠? 이 내용은 꼭 외우세요.

13.5 자손 선택자와 후손 선택자

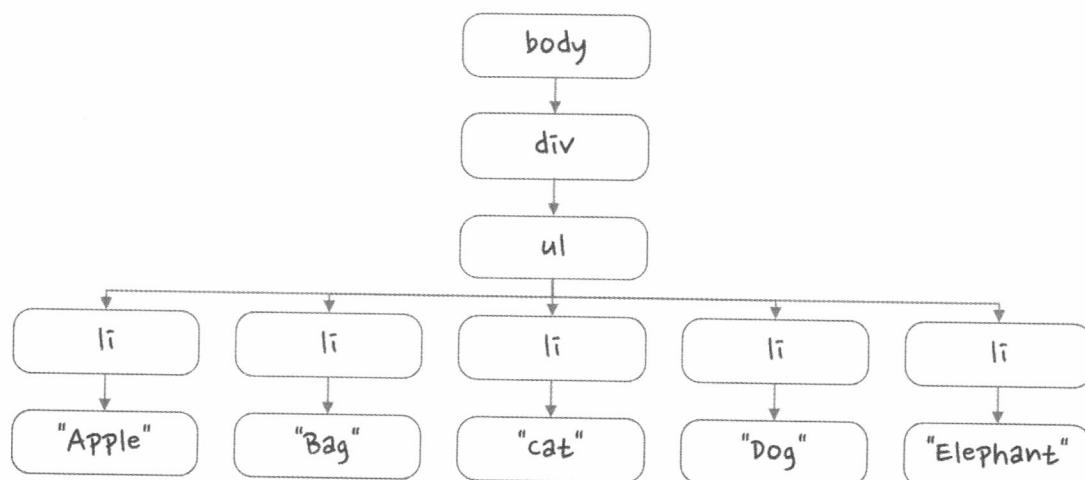
자손 선택자와 후손 선택자는 기본 선택자의 앞에 붙여 사용하며 기본 선택자의 범위를 제한합니다. 코드 13-18처럼 body 태그를 구성합니다.

코드 13-18 body 태그 구성

```
<body>
  <div>
    <ul>
      <li>Apple</li>
      <li>Bag</li>
      <li>Cat</li>
      <li>Dog</li>
      <li>Elephant</li>
    </ul>
  </div>
</body>
```

전체적인 문서 객체 모델을 나타내면 그림 13-13처럼 나타낼 수 있습니다. body 태그를 기준으로 할 때 body 태그 바로 아래 있는 div 태그를 자손이라고 부릅니다. 이름 그대로 body 태그의 아래에 있는 모든 문서 객체를 후손이라고 부릅니다.

그림 13-13 자손과 후손



이 책의 내용을 살펴보면 알겠지만 예제의 소스 코드가 1페이지만 넘어가도 모두 이 선택자를 사용합니다. 일반 웹 사이트는 당연히 1페이지는 넘어가므로 이 선택자를 굉장히 많이 사용합니다.

13.5.1 자손 선택자

자손 선택자는 자손을 선택하는 선택자이며 ‘요소A > 요소B’의 형태로 사용합니다. 코드 13-19처럼 사용하면 body 태그의 자식으로 범위를 한정해 전체가 선택됩니다.

코드 13-19 자손 선택자

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

따라서 코드를 실행하면 그림 13-14처럼 div 태그에만 스타일이 적용된 것을 확인할 수 있습니다.

그림 13-14 자손 선택자

```
▼<html>
  ▶<head>...</head>
  ▼<body>
    ▼<div style="color: red; ">
      ▼<ul>
        <li>Apple</li>
        <li>Bag</li>
        <li>Cat</li>
        <li>Dog</li>
        <li>Elephant</li>
      </ul>
    </div>
  </body>
</html>
```

13.5.2 후손 선택자

후손 선택자는 이름 그대로 후손을 선택하는 선택자입니다. ‘요소A 요소B’의 형태로 사용하며 요소A의 후손으로 범위를 한정합니다. 코드 13-20은 body 태그의 모든 후손을 선택합니다.

코드 13-20 후손 선택자

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

따라서 코드를 실행해보면 그림 13-15처럼 body 태그의 후손에 모두 스타일이 적용됐습니다.

그림 13-15 후손 선택자

```
▼<html>
  ▶<head>..</head>
  ▼<body>
    ▼<div style="color: red; ">
      ▼<ul style="color: red; ">
        <li style="color: red; ">Apple</li>
        <li style="color: red; ">Bag</li>
        <li style="color: red; ">Cat</li>
        <li style="color: red; ">Dog</li>
        <li style="color: red; ">Elephant</li>
      </ul>
    </div>
  </body>
</html>
```

이 절에서는 자손 선택자와 후손 선택자, 전체 선택자를 사용했는데요. 앞 절에서 배운 기본 선택자를 모두 활용할 수 있습니다.

13.6 속성 선택자

이 절에서 살펴볼 내용은 속성 선택자입니다. 속성 선택자는 기본 선택자 뒤에 붙여 사용합니다. jQuery는 표 13-1의 속성 선택자를 지원합니다.

표 13-1 jQuery 속성 선택자

선택자 형태	설명
요소[속성=값]	속성과 값이 같은 문서 객체를 선택합니다.
요소[속성!=값]	속성 안의 값이 특정 값과 같은 문서 객체를 선택합니다.
요소[속성~=값]	속성 안의 값이 특정 값을 단어로 시작하는 문서 객체를 선택합니다.
요소[속성^=값]	속성 안의 값이 특정 값으로 시작하는 문서 객체를 선택합니다.
요소[속성\$=값]	속성 안의 값이 특정 값으로 끝나는 문서 객체를 선택합니다.
요소[속성*=값]	속성 안의 값이 특정 값을 포함하는 문서 객체를 선택합니다.

속성 선택자는 입력 양식과 관련된 태그를 선택할 때 많이 사용합니다. 코드 13-21의 body 태그를 살펴봅시다. 코드 13-21의 input 태그들은 태그 선택자로 구분할 수 없습니다. 그렇다고 아이디 선택자나 클래스 선택자를 사용하겠다고, id 속성이나 class 속성을 입력하면 굉장히 복잡하겠죠?

코드 13-21 body 태그 구성

```
<body>
    <input type="text" />
    <input type="password" />
    <input type="radio" />
    <input type="checkbox" />
    <input type="file" />
</body>
```

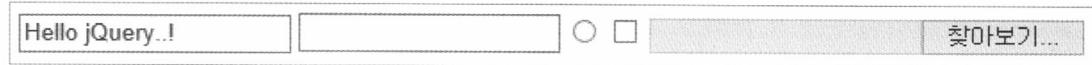
이러한 경우에 사용하는 것이 속성 선택자입니다. 속성 선택자는 코드 13-22처럼 사용합니다.

코드 13-22 속성 선택자

```
<script>
$(document).ready(function () {
    $('input[type="text"]').val('Hello jQuery..!');
});
</script>
```

코드 13-22에서 사용한 `val()` 메서드는 매개변수를 입력하면 `input` 태그의 `value` 속성을 지정하고, 매개변수를 입력하지 않으면 `value` 속성을 검사합니다. 코드에서는 매개변수를 입력했으므로 선택한 문서 객체에 그림 13-16처럼 `value` 속성을 입력합니다.

그림 13-16 속성 선택자



코드 13-21로 다른 태그를 선택하는 연습도 해보세요. 다른 속성 선택자는 활용도가 많이 떨어지므로 별도로 언급하지 않겠습니다. 이 절에서 꼭 외우고 지나가야 하는 내용은 예제로 살펴본 일치 속성 선택자입니다. 다음 두 절의 선택자가 너무 어렵다고 생각한다면, 속성 선택자만이라도 확실히 알고 넘어가세요.

13.7 필터 선택자

선택자 중에 : 기호를 포함하는 선택자를 필터 선택자라고 부릅니다. 이 절에서는 필터 선택자를 살펴보겠습니다.

13.7.1 입력 양식 필터 선택자

앞 절에서 살펴본 속성 선택자를 사용하면 `input` 태그의 `type` 속성에 따라 문서 객체를 선택할 수 있습니다. 조금 더 간단한 방법을 쓰고 싶다면 표 13-2의 필터 선택자를 사용합니다. 필터 선택자는 기본 선택자 뒤에 사용합니다.

표 13-2 jQuery의 입력 양식 필터 선택자(1)

선택자 형태	설명
<code>요소:button</code>	<code>input</code> 태그 중 <code>type</code> 속성이 <code>button</code> 인 문서 객체와 <code>button</code> 태그를 선택합니다.
<code>요소:checkbox</code>	<code>input</code> 태그 중 <code>type</code> 속성이 <code>checkbox</code> 인 문서 객체를 선택합니다.
<code>요소:file</code>	<code>input</code> 태그 중 <code>type</code> 속성이 <code>file</code> 인 문서 객체를 선택합니다.
<code>요소:image</code>	<code>input</code> 태그 중 <code>type</code> 속성이 <code>image</code> 인 문서 객체를 선택합니다.
<code>요소:password</code>	<code>input</code> 태그 중 <code>type</code> 속성이 <code>password</code> 인 문서 객체를 선택합니다.
<code>요소:radio</code>	<code>input</code> 태그 중 <code>type</code> 속성이 <code>radio</code> 인 문서 객체를 선택합니다.
<code>요소:reset</code>	<code>input</code> 태그 중 <code>type</code> 속성이 <code>reset</code> 인 문서 객체를 선택합니다.
<code>요소:submit</code>	<code>input</code> 태그 중 <code>type</code> 속성이 <code>submit</code> 인 문서 객체를 선택합니다.
<code>요소:text</code>	<code>input</code> 태그 중 <code>type</code> 속성이 <code>text</code> 인 문서 객체를 선택합니다.

표 13-2의 입력 양식 필터 선택자는 ‘input:button’의 형태로 선택하면 됩니다. 이 절에서 예제를 작성하며 알아볼 내용은 표 13-3의 필터 선택자입니다.

표 13-3 jQuery의 입력 양식 필터 선택자(2)

선택자 형태	설명
요소:checked	체크되어 있는 입력 양식을 선택합니다.
요소:disabled	비활성화된 입력 양식을 선택합니다.
요소:enabled	활성화된 입력 양식을 선택합니다.
요소:focus	초점이 맞추어져 있는 입력 양식을 선택합니다.
요소:input	모든 입력 양식을 선택합니다(input, textarea, select, button 태그).
요소:selected	option 객체 중 선택된 태그를 선택합니다.

아직 jQuery의 이벤트를 배우지 않았으므로 표 13-3의 속성 선택자를 모두 알아보는 것은 조금 무리입니다. 웹 페이지를 실행하자마자 메서드를 실행할 것인데 웹 페이지를 실행한 순간에 사용자가 입력 양식을 이미 선택했을 수는 없겠죠? 그래서 setTimeout() 함수로 웹 페이지를 실행하고 5초 정도 후에 선택자로 간단한 연습만 해봅시다.

body 태그를 코드 13-23처럼 구성합니다.

코드 13-23 body 태그 구성

```
<body>
  <select>
    <option>Apple</option>
    <option>Bag</option>
    <option>Cat</option>
    <option>Dog</option>
    <option>Elephant</option>
  </select>
</body>
```

코드 13-24처럼 setTimeout() 함수를 사용하고 내부에서 선택자를 사용합니다. 코드 13-24에서 사용한 선택자는 select 태그 아래에 있는 option 태그 중 사용자가 선택한 태그를 선택합니다. 이어서, 선택한 문서 객체의 val() 메서드를 매개변수 없이 사용하면 입력 양식의 value 속성을 알아낼 수 있습니다.

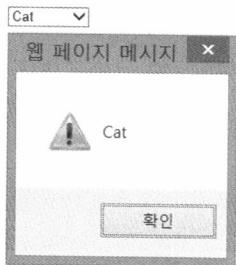
코드 13-24 입력 양식 필터 선택자

```
<script>
$(document).ready(function () {
    // 5초 후에 코드를 실행합니다.
    setTimeout(function () {
        // 변수를 선언합니다.
        var value = $('select > option:selected').val();

        // 출력합니다.
        alert(value);
    }, 5000);
});
</script>
```

코드 13-24를 실행하고 5초를 기다리면 select 태그로 선택한 글자가 그림 13-17처럼 출력됩니다.

그림 13-17 필터 선택자를 사용한 문서 객체 선택



이 절에서 사용한 :selected 필터는 굉장히 많이 사용됩니다. 내용을 따로 외워두는 것도 좋습니다. 이 절의 내용을 연습 삼아 다음 예제를 직접 만들어봅시다. 아직 이벤트를 안 배웠으므로 방금 사용한 setTimeout() 함수로 구현하세요.

- 사용자가 선택한 텍스트 박스에 색상 적용
- 체크한 체크 박스의 value 속성 출력

아직은 약간 어려울 수 있지만 최대한 만들어봅시다.

13.7.2 위치 필터 선택자

jQuery는 굉장히 많은 필터 선택자를 제공합니다. 위치와 관련된 필터 선택자를 살펴보겠습니다. 표 13-4는 위치와 관련된 필터 선택자입니다.

표 13-4 jQuery의 위치 필터 선택자

선택자 형태	설명
요소:odd	홀수 번째에 위치한 문서 객체를 선택합니다.
요소:even	짝수 번째에 위치한 문서 객체를 선택합니다.
요소:first	첫 번째에 위치한 문서 객체를 선택합니다.
요소:last	마지막에 위치한 문서 객체를 선택합니다.

선택자로 문서 객체를 선택하고 표 13-4의 위치를 지정한다는 점을 기억해주세요. body 태그를 코드 13-25처럼 구성해 줄무늬의 표를 만들어봅시다.

코드 13-25 body 태그 구성

```
<body>
  <table>
    <tr><th>이름</th><th>혈액형</th><th>지역</th></tr>
    <tr><td>강민수</td><td>AB형</td><td>서울특별시 송파구</td></tr>
    <tr><td>구지연</td><td>B형</td><td>미국 캘리포니아</td></tr>
    <tr><td>김미화</td><td>AB형</td><td>미국 메사추세츠</td></tr>
    <tr><td>김선화</td><td>O형</td><td>서울 강서구</td></tr>
    <tr><td>남기주</td><td>A형</td><td>서울 노량진구</td></tr>
    <tr><td>윤하린</td><td>B형</td><td>서울 용산구</td></tr>
  </table>
</body>
```

줄무늬가 되게 하려면 홀수 번째 색과 짝수 번째 색이 다르면 되겠죠? 코드 13-26처럼 홀수 번째 tr 태그와 짝수 번째 tr 태그에 다른 배경색을 입력합니다. 첫 번째 tr 태그에는 배경색을 검정색으로, 글자색을 흰색으로 적용합니다.

코드 13-26 위치 필터 선택자

```
<script>
$(document).ready(function () {
  $('tr:odd').css('background', '#F9F9F9');
```

```

        $('tr:even').css('background', '#9F9F9F');

        $('tr:first').css('background', '#000000').css('color', '#FFFFFF');
    });
</script>

```

첫 번째 tr 태그를 선택할 때 1부에서 살펴본 체이닝을 사용했습니다. 대부분의 jQuery 메서드가 이렇게 체이닝을 사용할 수 있다는 사실을 기억해두세요. 결과는 그림 13-18과 같습니다. 간단하지만 문서 객체를 예쁘게 꾸밀 수 있습니다.

그림 13-18 필터 선택자를 사용한 테이블 줄무늬 꾸미기(1)

이름	혈액형	지역
강민수	AB형	서울특별시 송파구
구지연	B형	미국 캘리포니아
김미화	AB형	미국 메사추세츠
김선화	O형	서울 강서구
남기주	A형	서울 노량진구
윤하린	B형	서울 용산구

13.7.3 함수 필터 선택자

jQuery는 함수 형태의 필터 선택자를 제공합니다. 표 13-5는 jQuery가 제공하는 함수 필터 선택자입니다.

표 13-5 jQuery의 함수 필터 선택자

선택자 형태	설명
요소:contains(문자열)	특정 문자열을 포함하는 문서 객체를 선택합니다.
요소:eq(n)	n번째에 위치하는 문서 객체를 선택합니다.
요소:gt(n)	n번째 초과에 위치하는 문서 객체를 선택합니다.
요소:has(h1)	h1 태그가 있는 문서 객체를 선택합니다.
요소:lt(n)	n번째 미만에 위치하는 문서 객체를 선택합니다.
요소:not(선택자)	선택자와 일치하지 않는 문서 객체를 선택합니다.
요소:nth-child(3n+1)	3n+1번째에 위치하는 문서 객체를 선택합니다(1, 4, 7,).

표 13-5의 선택자 중 :nth-child() 필터 선택자는 사용 방법이 조금 특이합니다. 코드 13-27처럼 함수의 괄호 안에 식별자 n으로 문서 객체의 순서를 나타냅니다. 코드 13-27의

13.7.2 위치 필터 선택자

jQuery는 굉장히 많은 필터 선택자를 제공합니다. 위치와 관련된 필터 선택자를 살펴보겠습니다. 표 13-4는 위치와 관련된 필터 선택자입니다.

표 13-4 jQuery의 위치 필터 선택자

선택자 형태	설명
요소:odd	홀수 번째에 위치한 문서 객체를 선택합니다.
요소:even	짝수 번째에 위치한 문서 객체를 선택합니다.
요소:first	첫 번째에 위치한 문서 객체를 선택합니다.
요소:last	마지막에 위치한 문서 객체를 선택합니다.

선택자로 문서 객체를 선택하고 표 13-4의 위치를 지정한다는 점을 기억해주세요. body 태그를 코드 13-25처럼 구성해 줄무늬의 표를 만들어봅시다.

코드 13-25 body 태그 구성

```
<body>
  <table>
    <tr><th>이름</th><th>혈액형</th><th>지역</th></tr>
    <tr><td>강민수</td><td>AB형</td><td>서울특별시 송파구</td></tr>
    <tr><td>구지연</td><td>B형</td><td>미국 캘리포니아</td></tr>
    <tr><td>김미화</td><td>AB형</td><td>미국 메사추세츠</td></tr>
    <tr><td>김선화</td><td>O형</td><td>서울 강서구</td></tr>
    <tr><td>남기주</td><td>A형</td><td>서울 노량진구</td></tr>
    <tr><td>윤하린</td><td>B형</td><td>서울 용산구</td></tr>
  </table>
</body>
```

줄무늬가 되게 하려면 홀수 번째 색과 짝수 번째 색이 다르면 되겠죠? 코드 13-26처럼 홀수 번째 tr 태그와 짝수 번째 tr 태그에 다른 배경색을 입력합니다. 첫 번째 tr 태그에는 배경색을 검정색으로, 글자색을 흰색으로 적용합니다.

코드 13-26 위치 필터 선택자

```
<script>
$(document).ready(function () {
  $('tr:odd').css('background', '#F9F9F9');
```

```

        $('tr:even').css('background', '#9F9F9F');

        $('tr:first').css('background', '#000000').css('color', '#FFFFFF');
    });
</script>

```

첫 번째 tr 태그를 선택할 때 1부에서 살펴본 체이닝을 사용했습니다. 대부분의 jQuery 메서드가 이렇게 체이닝을 사용할 수 있다는 사실을 기억해두세요. 결과는 그림 13-18과 같습니다. 간단하지만 문서 객체를 예쁘게 꾸밀 수 있습니다.

그림 13-18 필터 선택자를 사용한 테이블 줄무늬 꾸미기(1)

이름	혈액형	지역
강민수	AB형	서울특별시 송파구
구지연	B형	미국 캘리포니아
김미화	AB형	미국 메사추세츠
김선화	O형	서울 강서구
남기주	A형	서울 노량진구
윤하린	B형	서울 용산구

13.7.3 함수 필터 선택자

jQuery는 함수 형태의 필터 선택자를 제공합니다. 표 13-5는 jQuery가 제공하는 함수 필터 선택자입니다.

표 13-5 jQuery의 함수 필터 선택자

선택자 형태	설명
요소:contains(문자열)	특정 문자열을 포함하는 문서 객체를 선택합니다.
요소:eq(n)	n번째에 위치하는 문서 객체를 선택합니다.
요소:gt(n)	n번째 초과에 위치하는 문서 객체를 선택합니다.
요소:has(h1)	h1 태그가 있는 문서 객체를 선택합니다.
요소:lt(n)	n번째 미만에 위치하는 문서 객체를 선택합니다.
요소:not(선택자)	선택자와 일치하지 않는 문서 객체를 선택합니다.
요소:nth-child(3n+1)	3n+1번째에 위치하는 문서 객체를 선택합니다(1, 4, 7,).

표 13-5의 선택자 중 :nth-child() 필터 선택자는 사용 방법이 조금 특이합니다. 코드 13-27처럼 함수의 괄호 안에 식별자 n으로 문서 객체의 순서를 나타냅니다. 코드 13-27의

첫 번째 nth-child() 필터 선택자는 $3n+1$ 번째에 위치하는 선택자를 선택하므로 1, 4, 7번째에 위치하는 문서 객체를 선택합니다.

코드 13-27 함수 필터 선택자

```
<script>
$(document).ready(function () {
    $('tr:eq(0)').css('background', '#000000').css('color', 'white');
    $('td:nth-child(3n+1)').css('background', '#565656');
    $('td:nth-child(3n+2)').css('background', '#A9A9A9');
    $('td:nth-child(3n)').css('background', '#F9F9F9');
});
</script>
```

코드를 실행하면 그림 13-19처럼 3의 배수 단위로 열의 색상이 같은 표를 볼 수 있습니다.

그림 13-19 필터 선택자를 사용한 테이블 줄무늬 꾸미기(2)

이름	혈액형	지역
김민수	AB형	서울특별시 송파구
고지영	B형	미국 캘리포니아
김민희	AB형	미국 메사추세츠
김선희	O형	서울 강서구
남기주	A형	서울 노량진구
윤여린	B형	서울 용산구

필터 선택자는 기본 선택자에 비해 활용도가 굉장히 떨어집니다. 따라서 필터 선택자를 모두 외우기보다는 필요할 때 찾아 쓰는 쪽을 추천합니다.

13.8 배열 관리

jQuery로 배열을 관리할 때는 each() 메서드를 사용합니다. each() 메서드는 매개변수로 입력한 함수로 for in 반복문처럼 객체나 배열의 요소를 검사하는 메서드입니다. each() 메서드는 다음과 같이 두 가지 형태로 사용합니다.

- 1 \$.each(object, function(index, item) { })
- 2 \$(selector).each(function(index, item) { })

13.8.1 자바스크립트 배열 관리

자바스크립트 배열에 들어 있는 내용을 HTML 페이지에 표시하는 간단한 예제를 만들며 첫 번째 형태인 `$.each()` 메서드를 살펴보겠습니다.

우선 코드 13-28처럼 `script` 태그에 배열 `array`를 선언합니다. 배열 `array`는 4개의 객체가 있고, 각각의 객체는 `name` 속성과 `link` 속성이 있습니다.

코드 13-28 배열 선언

```
<script>
$(document).ready(function () {
    // 변수를 선언합니다.
    var array = [
        { name: 'Hanbit Media', link: 'http://hanb.co.kr' },
        { name: 'Naver', link: 'http://naver.com' },
        { name: 'Daum', link: 'http://daum.net' },
        { name: 'Paran', link: 'http://paran.com' }
    ];
});
</script>
```

`$.each()` 메서드의 첫 번째 매개변수에는 코드 13-29처럼 배열을 넣습니다. 두 번째 매개변수는 매개변수로 `index`와 `item`을 갖는 함수를 넣습니다.

코드 13-29 `$.each()` 메서드

```
<script>
$(document).ready(function () {
    // 변수를 선언합니다.
    var array = [
        { name: 'Hanbit Media', link: 'http://hanb.co.kr' },
        { name: 'Naver', link: 'http://naver.com' },
        { name: 'Daum', link: 'http://daum.net' },
        { name: 'Paran', link: 'http://paran.com' }
    ];

    // $.each() 메서드를 사용합니다.
    $.each(array, function (index, item) {
```

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

두 번째 매개변수로 넣은 함수의 매개변수 `index`는 배열의 인덱스 또는 객체의 키를 의미하며 매개변수 `item`은 해당 인덱스나 키가 가진 값을 의미합니다.

이를 이용해 코드 13-30처럼 `item` 객체 안에 들어 있는 `name` 속성과 `link` 속성으로 링크를 만들어 `body` 태그의 뒷부분에 넣습니다.

코드 13-30 `$.each()` 메서드의 콜백 함수

```
// $.each() 메서드를 사용합니다.
$.each(array, function (index, item) {
    // 변수를 선언합니다.
    var output = '';

    // 문자열을 만듭니다.
    output += '<a href="' + item.link + '">';
    output += '    <h1>' + item.name + '</h1>';
    output += '</a>';

    // 집어넣습니다.
    document.body.innerHTML += output;
});
```

코드를 실행하면 그림 13-20처럼 출력됩니다.

그림 13-20 배열로부터 동적으로 생성한 문서 객체



3부에서 Ajax를 진행할 때 `$.each()` 형태의 메서드를 굉장히 많이 사용하므로 꼭 기억해주세요.

NOTE `forEach()` 메서드와 차이점

ECMAScript5에서 추가된 `forEach()` 메서드와 매개변수의 순서가 다르므로 주의하세요!

<code>forEach()</code> 메서드	<code>each()</code> 메서드
<code>[].forEach(function (item, index) { });</code>	<code>\$.each(function (index, item) { });</code>

13.8.2 jQuery 배열 관리

이제 자바스크립트 배열이 아니라 jQuery 배열 객체를 다루는 방법을 살펴보겠습니다. jQuery의 배열 객체는 따로 만드는 것이 아니라, 선택자로 여러 개의 문서 객체를 선택할 때 생성됩니다.

코드 13-31처럼 HTML 페이지를 구성하고 시작합니다. 우선 `style` 태그에서 `high-light` 클래스에 `background` 속성을 지정합니다.

코드 13-31 HTML 페이지 구성

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

```
<h1>item - 0</h1>
<h1>item - 1</h1>
<h1>item - 2</h1>
<h1>item - 3</h1>
<h1>item - 4</h1>
</body>
</html>
```

each() 메서드를 살펴보기 전에 addClass() 메서드와 removeClass() 메서드부터 간단하게 살펴봅시다. addClass() 메서드는 문서 객체에 class 속성을 추가하는 메서드입니다. addClass() 메서드는 코드 13-32처럼 사용합니다. 반대로 removeClass() 메서드는 문서 객체의 class 속성을 제거하는 메서드입니다.

코드 13-32 addClass() 메서드

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

본격적으로 이 절의 주제인 each() 메서드를 살펴봅시다. \$() 메서드로 h1 객체를 선택하면 현재 body 태그 안에 h1 태그가 5개이므로 5개의 문서 객체를 가져옵니다. 각각의 객체에 다른 설정을 하고 싶을 때 코드 13-33처럼 each() 메서드를 사용합니다.

코드 13-33 \$(selector).each() 메서드

```
<script>
$(document).ready(function () {
    $('h1').each(function (index, item) {

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

each() 메서드의 매개변수로 입력하는 함수는 첫 번째 매개변수에 index를 가지며, 두 번째 매개변수에는 각각의 item을 가집니다. 이전 절의 \$.each() 메서드와 차이가 없으므로 쉽게 이해할 수 있을 것입니다.

각각의 문서 객체에 다른 클래스를 적용하는 간단한 예제를 만들어봅시다. style 태그에 코드 13-34처럼 다섯 개의 클래스를 입력합니다.

코드 13-34 style 태그 구성

```
<style>
    .high-light-0 { background: yellow; }
    .high-light-1 { background: orange; }
    .high-light-2 { background: blue; }
    .high-light-3 { background: green; }
    .high-light-4 { background: red; }
</style>
```

이어서 each() 메서드를 코드 13-35처럼 사용합니다. 이렇게 하면 각각의 객체에 서로 다른 클래스가 적용됩니다.

코드 13-35 each() 메서드를 사용한 서로 다른 클래스 적용

```
<script>
$(document).ready(function () {
    $('h1').each(function (index, item) {
        $(item).addClass('high-light-' + index);
    });
});
</script>
```

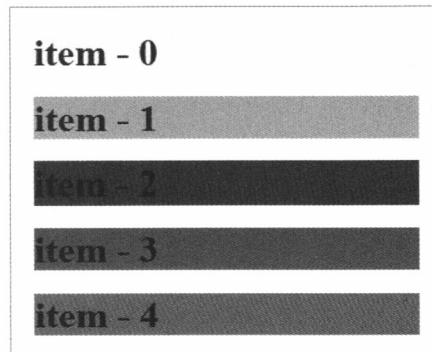
이렇게 매개변수 item을 사용할 수도 있지만, 일반적으로 코드 13-36처럼 this 키워드를 더 많이 사용합니다. each() 메서드의 매개변수로 입력한 함수 안에서 this 키워드와 item은 의미가 같습니다.

코드 13-36 each() 메서드에서의 this 키워드

```
<script>
$(document).ready(function () {
    $('h1').each(function (index, item) {
        $(this).addClass('high-light-' + index);
    });
});
</script>
```

코드를 실행하면 그림 13-21처럼 각각의 h1 태그에 서로 다른 클래스가 적용됩니다.

그림 13-21 각각의 클래스가 적용돼 색상이 나타납니다



참고로 addClass() 메서드의 매개변수에 코드 13-37처럼 함수를 입력할 수 있습니다.

코드 13-37 addClass() 메서드의 특수한 사용 방법

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

addClass() 메서드 이외에도 대부분의 jQuery 메서드들은 이와 같은 특성이 있습니다. 이는 이후의 내용을 살펴보며 차근차근 알아보겠습니다.

13.9 객체 확장

이 절에서 배울 메서드는 \$.extend() 메서드입니다. \$.extend() 메서드는 정말 중요한 메서드지만, 내용 자체는 간단합니다. 정말 jQuery를 깊게 공부하려는 순간 모르면 별목을 잡는 메서드이므로 확실하게 공부합시다. 우선 코드 13-38처럼 객체를 만들어주세요.

코드 13-38 객체 생성

```
<script>
$(document).ready(function () {
    var object = {};
});
</script>
```

기존의 자바스크립트로 빈 객체를 생성하고 이후에 새 속성을 추가할 때는 코드 13-39처럼 사용합니다.

코드 13-39 객체에 속성 추가

```
<script>
$(document).ready(function () {
    var object = {};
    object.name = 'RintianTta';
    object.gender = 'Male';
    object.part = 'Second Guitar';
});
</script>
```

이렇게 적은 수의 속성을 추가할 때는 문제가 없습니다. 하지만 많은 수의 속성을 추가할 때 이 방법을 사용하면 매우 귀찮고 코드가 지저분해집니다. 이러한 문제를 해결하고자 만들어진 메서드가 바로 `$.extend()` 메서드입니다. `$.extend()` 메서드는 다음 형태로 사용합니다.

```
$.extend(object, addObject, addObject, ...)
```

`$.extend()` 메서드의 첫 번째 매개변수로 입력한 객체에 두 번째 매개변수 이후의 객체를 합칩니다. 간단하게 코드 13-40처럼 사용할 수 있습니다.

코드 13-40 `$.extend()` 메서드

```
<script>
$(document).ready(function () {
    // 변수를 선언합니다.
    var object = { name: '윤인성' };
```

```

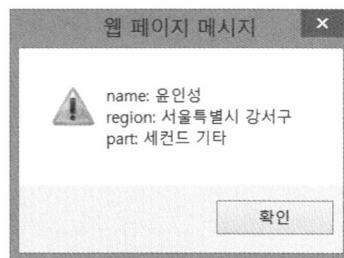
// $.extend() 메서드를 사용합니다.
$.extend(object, {
    region: '서울특별시 강서구',
    part: '세컨드 기타'
});

// 출력합니다.
var output = '';
$.each(object, function (key, item) {
    output += key + ': ' + item + '\n';
});
alert(output);
});
</script>

```

코드를 실행하면 그림 13-22처럼 `$.extend()` 메서드의 두 번째 매개변수에 입력한 객체의 속성이 `object` 객체에 합쳐진 것을 볼 수 있습니다.

그림 13-22 객체 결합



이 책의 4부에서 다시 살펴보는 내용이므로 기억해주세요.

13.10 jQuery 충돌 방지

jQuery 이외에도 여러 자바스크립트 프레임워크가 있습니다. 여러 플러그인을 함께 사용할 때는 플러그인 간의 충돌이 발생할 수 있습니다. 지금까지 식별자 `$`를 많이 사용했는데요. 예를 들어 Prototype 프레임워크에서도 식별자 `$`를 사용합니다. 따라서 jQuery 프레임워크와 Prototype 프레임워크를 함께 사용하면 프레임워크간의 충돌이 발생합니다.

여러 가지 프레임워크를 공용해서 사용하는 경우는 굉장히 많으므로 간단하게 정리합시다. 충돌을 방지할 때 사용하는 메서드는 `$.noConflict()` 메서드입니다. `$.noConflict()` 메서드를 사용하고 나면 더 이상 jQuery의 식별자 `$`를 사용할 수 없습니다.

코드 13-41 `$.noConflict()` 메서드

```
<script>
  $.noConflict();
  jQuery(document).ready(function () {
    });
</script>
```

따라서 조금 길지만 코드 13-41처럼 식별자 `jQuery`를 사용해야 합니다.

“계속 쓰는 녀석인데 너무 길어요!”

간단하게 쓰고 싶다면 코드 13-42처럼 `jQuery` 객체를 다른 변수에 저장해서 사용하세요.

코드 13-42 `$.noConflict()` 메서드 활용

```
<script>
  // 플러그인간의 충돌을 제거합니다.
  $.noConflict();
  var J = jQuery;

  // jQuery를 사용합니다.
  J(document).ready(function () {
    J('h1').removeClass('high-light');
  });
</script>
```
