

# 18장. jQuery I

강사 강태광

# jQuery 개념

1. jQuery는 모든 브라우저에서 동작하는 클라이언트 사이드 자바스크립트 라이브러리  
write less, do more
2. 특징
  - 크로스 브라우징을 지원
  - 오픈 소스 프로젝트
  - 사용자가 기능 확장 가능
  - 선택 기능이 우수해서 DOM과 관련된 처리를 쉽게 할 수 있습니다.
  - 이벤트 연결을 쉽게 구현
  - 시각적 효과 우수
  - Ajax 애플리케이션 개발이 쉽습니다.
  - <http://code.jquery.com/jquery-버전.js>
  - google CDN(Content Delivery Network) 이용
  - <https://ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js>  
MS의 CDN이용
  - <http://ajax.aspnetcdn.com/ajax/jQuery/jquery-버전.min.js>  
사용은 위의 주소를 link를 걸어도 되고 다운로드 받은 내용을 저장해서 하나의 파일로 만든 후 사용해도 됩니다.

# jQuery 개념

1. 모든 브라우저에서 동작하는 클라이언트 자바스크립트 라이브러리  
2006년 1월, 존 레식 John Resig 이 BarCamp NYC에서 발표
2. 무료로 사용 가능한 오픈소스 라이브러리
3. jQuery의 제작 목표
  - DOM과 관련된 처리 쉽게 구현
  - 일관된 이벤트 연결 쉽게 구현
  - 시각적 효과 쉽게 구현
  - Ajax 애플리케이션 쉽게 개발
4. JavaScript
  - Prototype.js, MooTools, jQuery등 중에서 jQuery가 77.82%점유
  - jQuery는 코딩이 간결하고 CSS와 코딩방법 유사

# jQuery 학습을 위해 준비할 것

## 1. 텍스트 편집기

- 이클립스

## 2. 웹 브라우저

- 모질라 파이어폭스
- 애플 사파리
- 마이크로 소프트 인터넷 익스플로러
- 크롬

## 3. jQuery 라이브러리

.

# jQuery 학습을 위한 환경 설정

1. JDK(Java Development Kit)
  - JDK는 자바 개발 툴이다.
  - JSP 웹 애플리케이션 개발을 위해서 사용하는 언어
  - JDK(JDK 5.0 이상)를 설치
2. 이클립스(eclipse)
  - 이클립스는 자바 개발을 지원해주는 자바 개발 도구
  - helios(eclipse-jee-helios-SR2-win32) 이상 버전
  - HTML 5를 기준으로 하기 위해
3. 웹 서버(톰캣)
  - JSP로 서버 페이지를 구현하기 위해서 필요한 웹 서버

# jQuery 다운로드와 CDN 방식

1. 다운받으려면 <http://jquery.com> 접속  
메인 화면에서 곧바로 jQuery 다운 가능

2. 다운받은 jquery.min.js파일을 js폴더에 저장하고  
<script src="js/ jquery.min.js">  
</script> 이렇게 사용 가능

3. CDN 이란?

CDN은 Content Delevery Network 의 약자

- 사용자에게 간편하게 콘텐츠 제공하는 방식 의미
- 구글, 마이크로소프트, jQuery측에서 사용자가 jQuery를 사용하기 편하게 콘텐츠 제공

4. jQuery CDN 호스트 사용해 이용  
HTML 페이지 구성 예시

```
<!DOCTYPE html>
<html>
<head>
  <script src="http://code.jquery.com/jquery-1.7.js"></script>
  <script>

  </script>
</head>
<body>

</body>
</html>
```

# jQuery 사용

## 5. script 태그의 src 속성에 제공되는 CDN 호스트 입력

- <http://code.jquery.com/jquery-1.7.js>
- <http://code.jquery.com/jquery-1.7.min.js>
- <http://ajax.googleapis.com/ajax/libs/jquery/1.7/jquery.min.js>
- <http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.7.js>
- <http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.7.min.js>

## 6. jQuery 파일명

- 1) ○○.js 파일 : Uncompressed 버전
- 2) ○○.min.js 파일
  - ○○.min.js 파일은 Minified 버전 (용량이 다섯 배 이상 차이)
  - Minified 버전은 파일의 용량을 최소화하려고 압축한 파일

## 7. 오프라인에서 jQuery 사용

- 반드시 다운받아 사용

# 처음으로 만들어 보는 jQuery 프로그램

## 1. jQuery 함수

- jQuery() 또는 \$()을 말한다.
- jQuery 래퍼(wrapper)라고도 불린다.
- 함수의 인자로 문자열(CSS 선택자로 표현)을 기술하여 선택한다.
- DOM 엘리먼트를 찾아서 결과 값으로 객체를 얻어준다.
- 이렇게 얻어진 객체를 **jQuery 확장 객체 집합**이라고 부른다

## 2. jQuery 확장 객체 집합

- jQuery 래퍼 집합이라고도 불린다.
- jQuery가 제공하는 모든 함수를 사용할 수 있게 된 상태이다.



# `$(document).ready()` 메서드

1. 브라우저에 HTML 문서가 로드되어 준비된 상태(ready)가 되었을 때
2. 셀렉터를 사용해서 특정 엘리먼트에 접근하여 스타일 시트를 적용

```
function testfn {  
  $('span').addClass('redtext');  
}
```

```
$(document).ready(testfn);
```

3. document 객체를 jQuery 함수로 둘러싼(wrapped) 후에 ready() 호출
  - ready()는 DOM이 로드되고 이미지가 로드되기 전에 호출되는 jQuery가 제공하는 이벤트 핸들러

# `$(document).ready()` 메서드

1. 단 한번만 호출되어 지는 함수일 경우에는 익명함수로 구현
2. 함수의 이름은 사용하지 않고 `function` 키워드를 사용하여 한정된 영역에서만 코드가 사용되도록 한다

```
$(document).ready(function() {  
    $('span').addClass('redtext');  
});
```

3. `$`의 의미 살피기
  - jQuery의 별칭으로 사용하는 기호이다.
  - 선택자를 따옴표로 묶어 괄호 안에 기술한다.
  - 결과로 얻어진 jQuery 객체 집합을 사용하여
  - 이벤트를 쉽게 바인딩하거나 효과를 함께 연결

# `$(document).ready()` 메서드

1. 단 한번만 호출되어 지는 함수일 경우에는 익명함수로 구현
2. 함수의 이름은 사용하지 않고 `function` 키워드를 사용하여 한정된 영역에서만 코드가 사용되도록 한다

```
$(document).ready(function() {  
    $('span').addClass('redtext');  
});
```

3. `$`의 의미 살피기
  - jQuery의 별칭으로 사용하는 기호이다.
  - 선택자를 따옴표로 묶어 괄호 안에 기술한다.
  - 결과로 얻어진 jQuery 객체 집합을 사용하여
  - 이벤트를 쉽게 바인딩하거나 효과를 함께 연결

# `$(document).ready()` 메서드

## 1. `$(document).ready()`

- 문서가 준비가 완료되면 매개 변수로 전달된 함수를 실행하라는 의미
- jQuery 이벤트 메서드 중 하나
- 아래 위 두 메서드는 비슷한 역할
- jQuery 이벤트 메서드는 이벤트로 여러 개의 함수 연결 가능한 장점

### 코드 13-1 document 객체의 ready 이벤트 연결

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

    });
</script>
```

---

### 코드 13-2 window 객체의 load 이벤트 연결

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

    };
</script>
```

---

# 예제 (jQuery I.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>jQuery Test</title>
    <script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
    <script type="text/javascript">
      $(document).ready(function() {
        $('h1').css('color', 'red');
      });
    </script>
  </head>
  <body>
    <h1>Hello</h1>
  </body>
</html>
```



# jQuery(document).ready( )

\$(document).ready()

3개의 경고창 띄우는 예제 alert3.html

```
<head>
<script src="http://code.jquery.com/jquery-1.7.js"> </script>
<script>
    $(document).ready(function () {
        alert('First READY');
    });
    $(document).ready(function () {
        alert('Second READY');
    });
    $(document).ready(function () {
        alert('Third READY');
    });
</script>
</head>
-----
<script>
    $(function () {
    });
</script>
```

. jQuery는 \$로 줄여 쓸 수 있음  
. jQuery(document).ready(function){};는  
\$(function){};로 줄여 쓸 수 있음

% 참고 <script type="text/javascript">

src="http://ajax.googleapis.com/ajax/libs/jquery/1.6.2/jquery.min.js"> </script>

# jQuery 기본 선택자

jQuery 메서드의 가장 기본적인 형태

- 문서 객체를 다룰 때 사용하는 형태
- jQuery에서 가장 많이 사용하는 형태

- jQuery('hl').css('color','red');

-----  
jQuery 선택자 메서드

# 셀렉터(선택자)란 무엇인가(\*\*\*\*\*)

## 1. 셀렉터(selector)

문서 내에서 원하는 엘리먼트를 쉽게 식별하고 이를 추출하기 위해서 jQuery에서 제공하는 기술이다.

## 2. 기본 CSS 셀렉터

셀렉터	설명
*	모든 엘리먼트와 일치
E	태그명이 E인 모든 엘리먼트와 일치
E F	E의 자손이면서 엘리먼트의 이름이 F인 모든 엘리먼트
E>F	E의 바로 아래 F 엘리먼트
E+F	E의 형제 엘리먼트로 바로 다음 형제 F 엘리먼트
E~F	E의 형제 엘리먼트로 다음에 나오는 모든 F 엘리먼트
E:has(F)	엘리먼트 F의 자손을 하나 이상 가지고 있는 E 모든 엘리먼트
E.C	클래스명 C를 가지고 있는 모든 E 엘리먼트.
*.C	클래스명 C를 가지고 있는 모든 엘리먼트
E#I	아이디가 I 인 모든 E 엘리먼트.
*#I	아이디가 I인 모든 엘리먼트
E[A]	어트리뷰트 A를 가지는 모든 E 엘리먼트
E[A=V]	어트리뷰트 A의 값이 V인 모든 E 엘리먼트
E[A^=V]	어트리뷰트 A의 값이 V로 시작하는 모든 E 엘리먼트
E[A\$=V]	값이 V로 끝나는 어트리뷰트 A를 가지고 있는 모든 E 엘리먼트
E[A*=V]	값에 V를 포함하는 어트리뷰트 A를 가지고 있는 모든 E 엘리먼트



# 셀렉터(선택자)란 무엇인가?

## 3. 위치 기반 셀렉터

셀렉터	설명
:first	페이지에서 처음으로 일치하는 엘리먼트를 반환한다.
:last	페이지에서 마지막으로 일치하는 엘리먼트를 반환한다.
:even	페이지 전체의 짝수 번째 엘리먼트를 반환한다.
:odd	페이지 전체의 홀수 번째 엘리먼트를 반환한다.
:eq(n)	n번째로 일치하는 엘리먼트를 반환한다.
:gt(n)	n번째 엘리먼트(포함하지 않음) 이후의 일치하는 엘리먼트를 반환한다.
:lt(n)	n번째 엘리먼트(포함하지 않음) 이전의 일치하는 엘리먼트를 반환한다.
:first-child	첫 번째 자식 엘리먼트를 반환한다.
:last-child	마지막 자식 엘리먼트를 반환한다.
:only-child	형제가 없는 모든 엘리먼트를 반환한다.
:nth-child(n)	n번째 자식 엘리먼트를 반환한다.
:nth-child(even)	짝수 자식 엘리먼트를 반환한다.
:nth-child(odd)	홀수 자식 엘리먼트를 반환한다.
:nth-child(Xn+Y)	전달된 공식에 따른 n번째 자식 엘리먼트를 반환한다. Y는 0인 경우 생략 가능하다.

# 셀렉터(선택자)란 무엇인가?

## 4. jQuery 정의 필터 셀렉터

셀렉터	설명
<code>:animated</code>	현재 애니메이션이 적용되고 있는 엘리먼트를 선택한다.
<code>:button</code>	모든 버튼을 선택한다( <code>input[type=submit]</code> , <code>input[type=reset]</code> , <code>input[type=button]</code> , <code>button</code> ).
<code>:checkbox</code>	체크박스 엘리먼트만 선택한다.
<code>:checked</code>	선택된 체크박스나 라디오 버튼만을 선택한다(CSS에서 지원).
<code>:contains(foo)</code>	텍스트 <code>foo</code> 를 포함하는 엘리먼트만 선택한다.
<code>:disabled</code>	인터페이스에서 비활성화 상태인 모든 폼 엘리먼트를 선택한다(CSS에서 지원).
<code>:enabled</code>	인터페이스에서 활성화 상태인 모든 폼 엘리먼트를 선택한다(CSS에서 지원).
<code>:file</code>	모든 파일 엘리먼트를 선택한다( <code>input[type=file]</code> ).
<code>:header</code>	헤더 엘리먼트만 선택한다. 예를 들어 <code>&lt;h1&gt;</code> 부터 <code>&lt;h6&gt;</code> 까지의 엘리먼트를 선택한다.
<code>:hidden</code>	감춰진 엘리먼트만 선택한다.
<code>:image</code>	폼 이미지를 선택한다( <code>input[type=image]</code> ).
<code>:input</code>	폼 엘리먼트만 선택한다( <code>input</code> , <code>select</code> , <code>textarea</code> , <code>button</code> ).
<code>:not(filter)</code>	필터의 값을 반대로 변경한다.
<code>:parent</code>	빈 엘리먼트를 제외하고, 텍스트도 포함해서 자식 엘리먼트를 가지는 엘리먼트를 선택한다.
<code>:password</code>	패스워드 엘리먼트만 선택한다( <code>input[type=password]</code> ).
<code>:radio</code>	라디오 버튼 엘리먼트만 선택한다( <code>input[type=radio]</code> ).
<code>:reset</code>	리셋 버튼을 선택한다( <code>input[type=reset]</code> 이나 <code>button[type=reset]</code> ).
<code>:selected</code>	선택된 엘리먼트만 선택한다.
<code>:submit</code>	전송 버튼을 선택한다( <code>button[type=submit]</code> 이나 <code>input[type=submit]</code> ).
<code>:text</code>	텍스트 엘리먼트만 선택한다( <code>input[type=text]</code> ).
<code>:visible</code>	보이는(visible) 엘리먼트만 선택한다.

# jQuery 기본 선택자 / 전체 선택자

1. 기본 선택자 사용은 CSS와 동일

전체 : \$('\*')

태그 : \$('div')

id : \$('#id1')

class : \$('.class1')

2. 자식 선택(한 단계 아래)

\$('body > \*')

3. 후손 선택(후손 모두)

\$('body \*')

4. 전체 선택자

- CSS의 가장 기본적인 선택자는 전체 선택자 Wildcard Selector

- HTML 페이지에 있는 모든 문서 객체를 선택하는 선택자 \*

  - 모든 문서 객체의 color 스타일 속성에 red 입력

  - **allcss.html**

```
<head>
```

```
<script src="http://code.jquery.com/jquery-1.7.js"></script>
```

```
<script>
```

```
    $(document).ready(function () { $('*').css('color', 'Red'); });
```

```
</script>
```

```
</head>
```

```
<body> <h1>Lorem ipsum</h1> </body>
```

# jQuery 기본 선택자

## 1. 태그 선택자

태그 선택자는 특정한 태그만 선택하는 선택자  
태그의 이름 그냥 사용 selectcss.html

```
<!DOCTYPE html>
<html>
  <head>
    <script src="http://code.jquery.com/jquery-1.7.js"> </script>
    <script>
      $(document).ready(function () {
        $('h1').css('color', 'Red');
      });
    </script>
  </head>
  <body>
    <h1>Lorem ipsum</h1>
    <p>Lorem ipsum dolor sit amet.</p>
    <h1>Lorem ipsum</h1>
    <p>consectetur adipiscing elit.</p>
  </body>
</html>
```

# jQuery 기본 선택자

## 1. 태그 선택자의 활용

- 하나 이상의 태그 선택자를 동시에 사용하고 싶을 때
  - 콤마로 선택자 구분
  - Selectcss2.html

```
<script src="http://code.jquery.com/jquery-1.7.js"> </script>
<script>
    $(document).ready(function () {
        $('h1, p').css('color', 'Orange');
    });
</script>
```

# jQuery 기본 선택자

## 1. 아이디 선택자

특정한 id 속성을 가지고 있는 문서 객체 선택하는 선택자

Ex) 예제 두 번째에 위치한 h1 태그가 id 속성으로 target 가짐  
id.html

```
<!DOCTYPE html>
<html>
  <head>
    <script src="http://code.jquery.com/jquery-1.7.js"> </script>
    <script>
      $(document).ready(function () {
        $('#target').css('color', 'Orange');
      });
    </script>
  </head>
  <body>
    <h1>Header-0</h1>
    <h1 id="target">Header-1</h1>
    <h1>Header-2</h1>
  </body>
</html>
```

# jQuery 기본 선택자

## 1. 아이디 선택자의 활용

- id 속성은 HTML 페이지 내에서 단 하나의 태그에만 적용
- 태그 선택자와 아이디 선택자 함께 사용하지 않아도 됨
- id2.html

```
<!DOCTYPE html>
<html>
<head>
  <script src="http://code.jquery.com/jquery-1.7.js"> </script>
  <script> $(document).ready(function () {
    $('h1#target').css('color', 'Orange');
  });
</script>
</head>
<body>
  <h1>Header-0</h1>
  <h1 id="target">Header-1</h1>
  <h1>Header-2</h1>
</body> </html>
```



# jQuery 기본 선택자

## 1. 클래스 선택자

- 1) 특정한 class 속성 가진 문서 객체를 선택하는 선택자
- 2) class l.html

```
<!DOCTYPE html>
<html>
<head>
<script src="http://code.jquery.com/jquery-1.7.js"></script>
<script>
    $(document).ready(function () {
        $('.item').css('color', 'Orange');
        $('h1.item').css('background', 'Red');
    });
</script>
</head>
<body>
    <h1 class="item">Header-0</h1>
    <h1 class="item select">Header-1</h1>
    <h1 class="item">Header-2</h1>
</body>
</html>
```



# jQuery 기본 선택자

1. 두 클래스 속성을 모두 갖는 문서 객체를 선택하고 싶을 때  
두 클래스 선택자 붙여서 사용

select2.html

```
<!DOCTYPE html>
<html>
  <head>
    <script src="http://code.jquery.com/jquery-1.7.js"></script>
    <script>
      $(document).ready(function () {
        $('.item.select').css('color', 'Orange');
      });
    </script>
  </head>
  <body>
    <h1 class="item">Header-0</h1>
    <h1 class="item select">Header-1</h1>
    <h1 class="item">Header-2</h1>
  </body>
</html>
```

# jQuery 자식 선택자와 후손 선택자

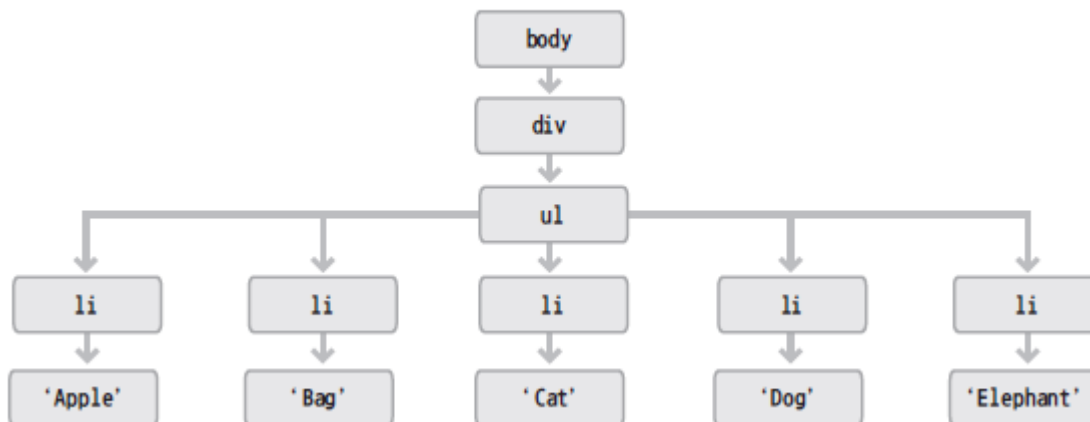
## 1. jQuery 자식 선택자와 후손 선택자

기본 선택자의 앞에 붙여 사용하며 기본 선택자의 범위 제한

코드 13-15 body 태그 구성

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

그림 13-9 자식과 후손



# jQuery 자식 선택자와 후손 선택자

## 1. 자식 선택자

1) 자식을 선택하는 선택자 **child.html**

2) '부모 > 자식' 의 형태로 사용

- body 태그의 자식으로 범위 한정해 전체 선택

```
<script src="http://code.jquery.com/jquery-1.7.js"></script>
```

```
<script>
```

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

```
</script>
```

```
</head>
```

```
<body>
```

```
    <div>
```

```
        <ul>
```

```
            <li>Apple</li>
```

```
            <li>Bag</li>
```

```
            <li>Cat</li>
```

```
            <li>Dog</li>
```

```
            <li>Elephant</li>
```

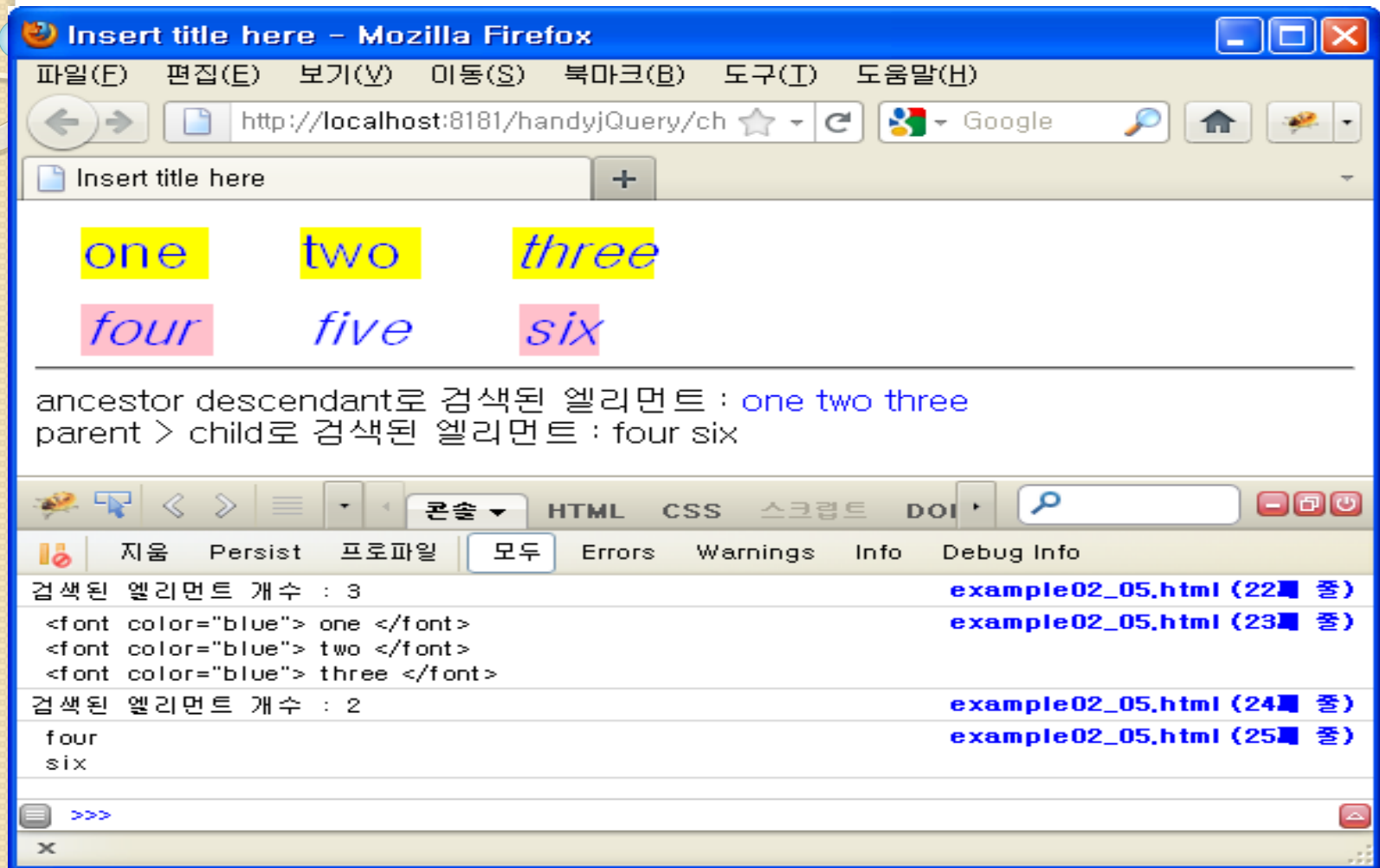
```
        </ul>
```

```
    </div>
```

```
</body>
```

# 인접한 자식 엘리먼트를 노드로 추가하기-1

## 1. \$('셀렉터 > 자손셀렉터')



## 인접한 자식 엘리먼트를 노드로 추가하기-2

`$('div > em') vs $('div em')`

```
<div>
  <em> four </em>
  <span>
    <em> five </em>
  </span>
  <em>six</em>
</div>
```

`$('셀렉터').append('추가할 내용')`  
append()의 인자로 기술한 내용을 `$()` 찾아 낸 엘리먼트에 추가

# 인접한 자식 엘리먼트를 노드로 추가하기\_2

```
<!DOCTYPE html> // append2.html
<html>
<head>
<meta charset="UTF-8">
<title>인접한 자손 엘리먼트를 노드로 추가하기</title>
<style type="text/css">
    em { font-size:20pt; line-height: 20pt; margin:20px; color:blue; }
</style>
<script src="../js/jquery.js" type="text/javascript"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('p em').css('background-color', 'yellow').each(function() {
            $('.result1').append($(this).text()+ "\n");
        });
        $('div > em').css('background-color', 'pink').each(function() {
            $('.result2').append($(this).text()+ "\n");
        });
        console.log("검색된 엘리먼트 개수 : " + $('p em').length);
        console.log("검색된 엘리먼트 개수 : " + $('div > em').length); });
</script>
</head>
```

# 인접한 자식 엘리먼트를 노드로 추가하기\_2

```
<body>
  <p>
    <em> one </em>
    <em> two </em>
    <span> <em> three </em> </span>
  </p>
  <div>
    <em> four </em>
    <span> <em> five </em> </span>
    <em>six</em>
  </div>
  <hr>
  <div>
    <span> ancestor descendant로 검색된 엘리먼트 : </span>
    <span class="result1"></span>
  </div>
  <div>
    <span> parent &gt; child로 검색된 엘리먼트 : </span>
    <span class="result2"></span>
  </div>
</body>
</html>
```

# jQuery 자식 선택자와 후손 선택자

## 1. 후손 선택자

### 1) 후손을 선택하는 선택자

### 2) '요소 A 요소 B'의 형태로 사용

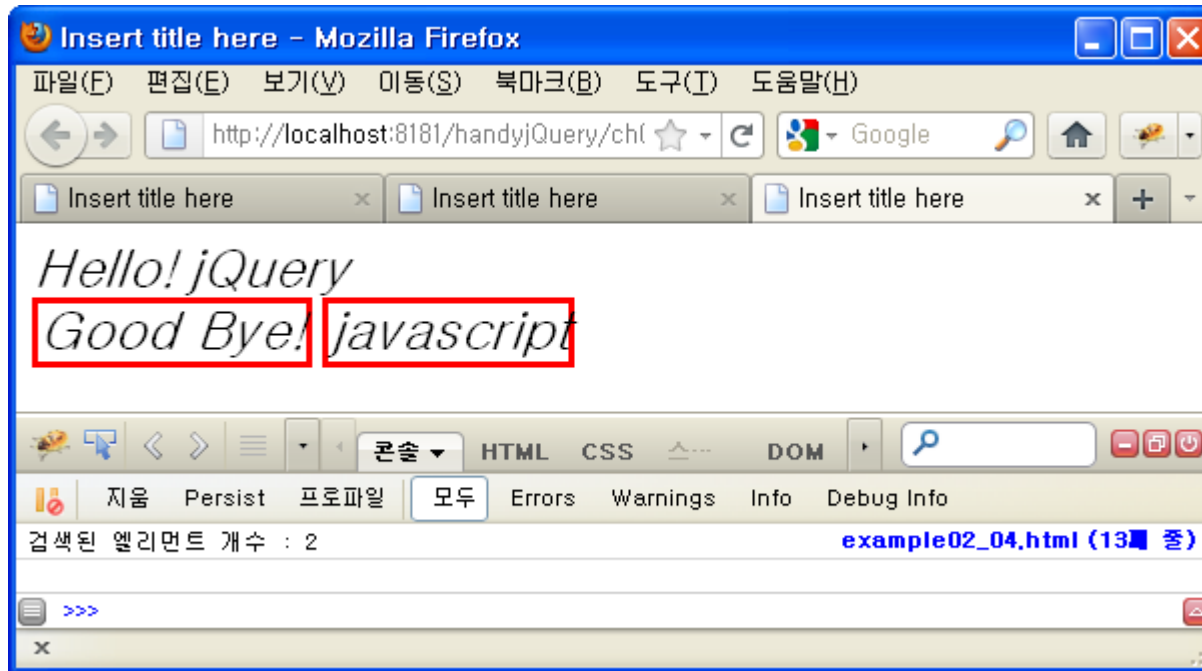
- 요소 A의 후손으로 범위 한정
- body 태그의 모든 후손 선택 descendant.html

```
<head>
  <script src="http://code.jquery.com/jquery-1.7.js"> </script>
  <script>
    $(document).ready(function () {
      $('body *').css('color', 'red');
    });
  </script>
</head>
<body>
  <div>
    <ul>
      <li>Apple</li>
      <li>Bag</li>
      <li>Cat</li>
      <li>Dog</li>
      <li>Elephant</li>
    </ul>
  </div>
```



# 선택된 래퍼 집합에서 자손 엘리먼트에 스타일시트 적용-I

1. \$('선택터 자손선택터') → \$('div em')
2. css()  
인자로 지정한 CSS 스타일을 선택한 엘리먼트에 적용
3. console.log("출력할 대상")  
- 콘솔(FireBug의 기능)에 인자로 기술한 문자열을 출력



## 선택된 래퍼 집합에서 자손 엘리먼트에 스타일시트 적용-2

```
$(document).ready(function() {  
  $('div em').css('border', '3pt solid #f00') .css('padding', '7pt 7pt 7pt 7pt');  
  
  $('span em').css('border', '1pt dotted #0f0') .css('padding', '7pt 7pt 7pt 7pt');  
  console.log("검색된 엘리먼트 개수 : " + $('div em').length);  
  console.log("검색된 엘리먼트 개수 : " + $('span em').length);  
});
```

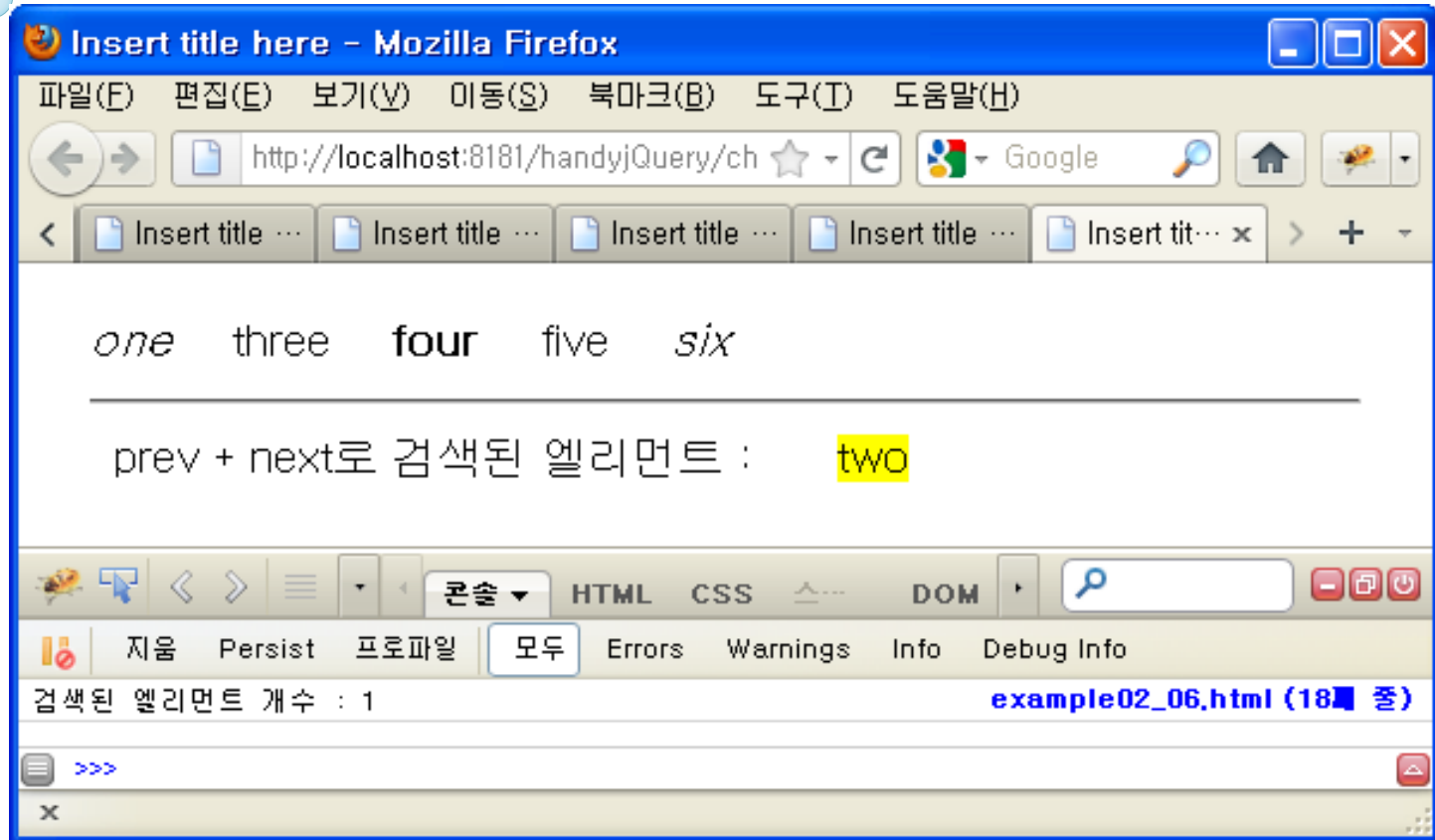
## 선택된 래퍼 집합에서 자손 엘리먼트에 스타일시트 적용-2

descent2.html

```
<!DOCTYPE html><html><head><meta charset="EUC-KR">
<title>Insert title here</title>
<script src="http://code.jquery.com/jquery-1.7.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('div em').css('border', '3pt solid #f00') .css('padding', '7pt 7pt 7pt 7pt');
        $('span em').css('border', '1pt dotted #0f0') .css('padding', '7pt 7pt 7pt 7pt');
        console.log("검색된 엘리먼트 개수 : "+ $('div em').length);
        console.log("검색된 엘리먼트 개수 : "+ $('span em').length);
    });
</script>
</head>
<body>
    <div>
        <em>Hello!</em>
        <em>jQuery</em>
        <em>forever</em>
    </div>
    <span>
        <em>Good Bye!</em>
        <em>javascript</em>
    </span>
</body></html>
```

## 모든 엘리먼트에 스타일을 적용하고 인접한 형제들 노드로 추가하기\_1

`$('*')`, `$('이전 선택터 + 다음 선택터')`



## 모든 엘리먼트에 스타일을 적용하고 인접한 형제들 노드로 추가하기\_2

`$('*', $('이전셀렉터' + 다음 셀렉터'))`

`$('em + a')` -> em의 형제 엘리먼트 중 다음 형제 a 엘리먼트를 반환

`$('추가할 내용').appendTo('셀렉터')`

`$()` 로 찾은 내용을 `appendTo()`의 인자로 기술한 엘리먼트에 추가

`$('셀렉터').append('추가할 내용')`와

`$('추가할 내용').appendTo('셀렉터')`의 차이점

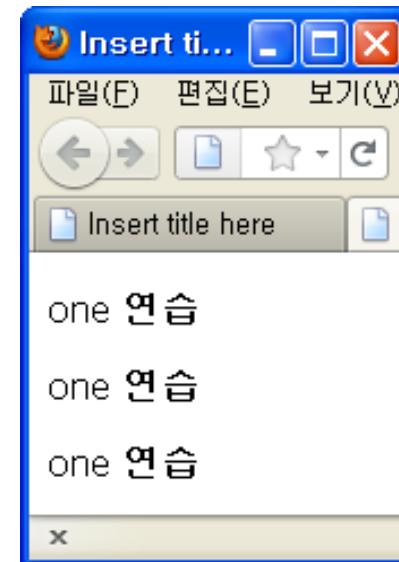
`<p>one</p>`

`<p>one</p>`

`<p>one</p>`

`$('p').append('<b>연습</b>');`

`$('<b>연습</b>').appendTo('p');`



## 모든 엘리먼트에 스타일을 적용하고 인접한 형제를 노드로 추가하기-2

```
<!DOCTYPE html>
<html> <head>
<meta charset="UTF-8">
<title>모든 엘리먼트에 스타일을 적용하고 인접한 형제를 노드로 추가하기-"*",
"prev + next"
</title>
<style type="text/css">
    .textstyle { font-size:14pt; line-height: 20pt; margin:10px; }
</style>
<script src="../js/jquery.js" type="text/javascript"> </script>
<script type="text/javascript">
    $(document).ready(function() {
        $('*').addClass('textstyle'); $('em + a').css('background-color',
            'yellow').each(function(){
                $(this).appendTo('.result1');
            });
        console.log("검색된 엘리먼트 개수 : " + $('em + a').length); });
</script>
</head>
```

## 모든 엘리먼트에 스타일을 적용하고 인접한 형제를 노드로 추가하기-2

```
<body>
  <em> one </em>
  <a> two </a>
  <a> three </a>
  <b> four </b>
  <a> five </a>
  <em> six </em>
  <hr>
  <div>
    <span> prev + next로 검색된 엘리먼트 : </span>
    <span class="result"></span>
  </div>
</body>
</html>
```

# jQuery를 사용한 배열 관리 (I)

## 1. jQuery 사용한 배열 관리

### 1) each() 메서드 사용

- 매개 변수로 입력한 함수 사용
- for in 반복문처럼 객체나 배열의 요소 검사하는 메서드

### 2) each() 메서드의 형태

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

### 3) 자바스크립트 배열에 들어 있는 내용 HTML 페이지에 표시

```
<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>
```



# jQuery를 사용한 배열 관리 (1)

## 1. \$.each() 메서드

- 첫 번째 매개 변수에는 배열 넣음
- 두 번째 매개 변수는 매개 변수로 index와 item 갖는 함수 넣음
  - Index - 배열의 인덱스 또는 객체의 키 의미
  - 매개 변수 item - 해당 인덱스나 키가 가진 값 의미

<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>

## jQuery를 사용한 배열 관리 (1)

\$.each() 메서드의 콜백 함수

- item 객체 안에 들어 있는 name, link 속성
  - 링크를 만들어 body 태그 뒷부분에 넣음 each.html

array.html

```
<script src="http://code.jquery.com/jquery-1.7.js"> </script>
```

```
<script>
```

```
    $(document).ready(function () { // 변수를 선언합니다.
```

```
        var array = [
```

```
        { name: 'Choongang', link: 'http://www.choongang.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) { // 변수를 선언합니다.
```

```
        var output = ""; // 문자열을 만듭니다.
```

```
        output += '<a href="' + item.link + '">';
```

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

```
        output += '</a>'; // 집어넣습니다.
```

```
        document.body.innerHTML += output;
```

```
    });
```

```
});
```

```
</script>
```

## jQuery를 사용한 배열 관리 (2)

jQuery의 배열 객체

- 선택자 사용해 여러 개의 문서 객체 선택할 때 생성
- style 태그에서 high\_light 클래스의 background 속성 지정

```
<head>
<style>
    .high_light {
        background:Yellow;
    }
</style>
<script src="http://code.jquery.com/jquery-1.7.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>
```

## jQuery를 사용한 배열 관리 (2)

### 1. addClass() 메서드

- 문서 객체에 class 속성 추가하는 메서드

```
<script src="http://code.jquery.com/jquery-1.7.js"> </script>
<script>
$(document).ready(function () {
$('h1').addClass('high_light');
});
</script>
```

### 2. removeClass() 메서드

문서 객체의 class 속성 제거하는 메서드

### 3. each() 메서드

- \$() 메서드 사용해 h1 객체 선택
- body 태그 안에 h1 태그가 다섯 개
  - 다섯 개의 문서 객체 가져옴
- 각 객체에 다르게 설정하고 싶을 때 each() 메서드 사용

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

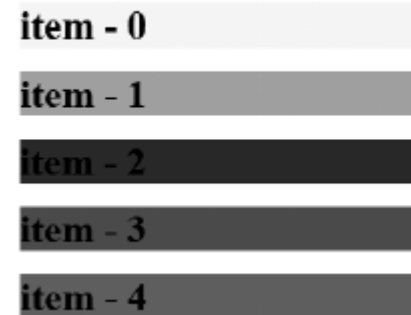
## jQuery를 사용한 배열 관리 (2)

### 1.each() 메서드

- 각 문서 객체에 다른 클래스 적용하는 간단한 예제  
Each2.html

```
<!DOCTYPE html> <html> <head>
<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>
<script src="http://code.jquery.com/jquery-1.7.js"> </script>
<script>
    $(document).ready(function () {
        $('h1').each(function (index, item) {
            $(this).addClass('high_light_' + index);
        });
    });
</script>
</head>
```

그림 13-17 각 클래스가 적용돼 색상이 나타납니다.



# jQuery를 사용한 배열 관리 (2)

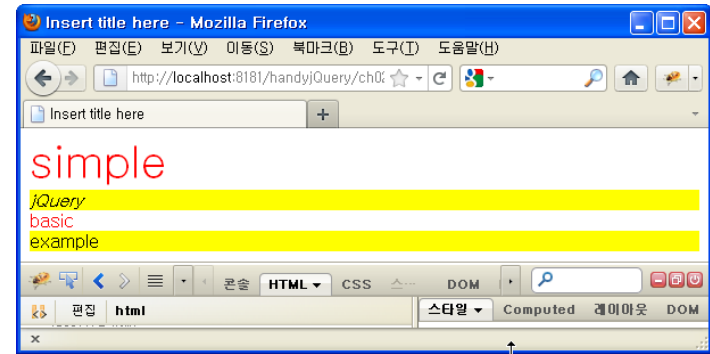
addClass() 메서드의 매개 변수 활용  
함수도 입력 가능

```
<script src="http://code.jquery.com/jquery-1.7.js"></script>
<script>
    $(document).ready(function () {
        $('h1').addClass(function (index) {
            return 'high_light_' + index;
        });
    });
</script>
```

# 엘리먼트에 접근해서 스타일 지정하기

```

$('span').addClass('redtext');
$('div').addClass('spotlight');
$('#simpletext1').addClass('largetext');
$('.simpletext1').addClass('italictext');
    
```



선택자	CSS	jQuery	설명
태그명	span	<code>\$('span')</code>	문서에 나오는 모든 HTML span 엘리먼트
ID	<code>#simpletext1</code>	<code>\$('#simpletext1')</code>	simpletext1라는 ID를 갖는 엘리먼트
CLASS	<code>.simpletext1</code>	<code>\$('.simpletext1')</code>	simpletext1라는 클래스 이름을 갖는 엘리먼트

addClass() 메소드

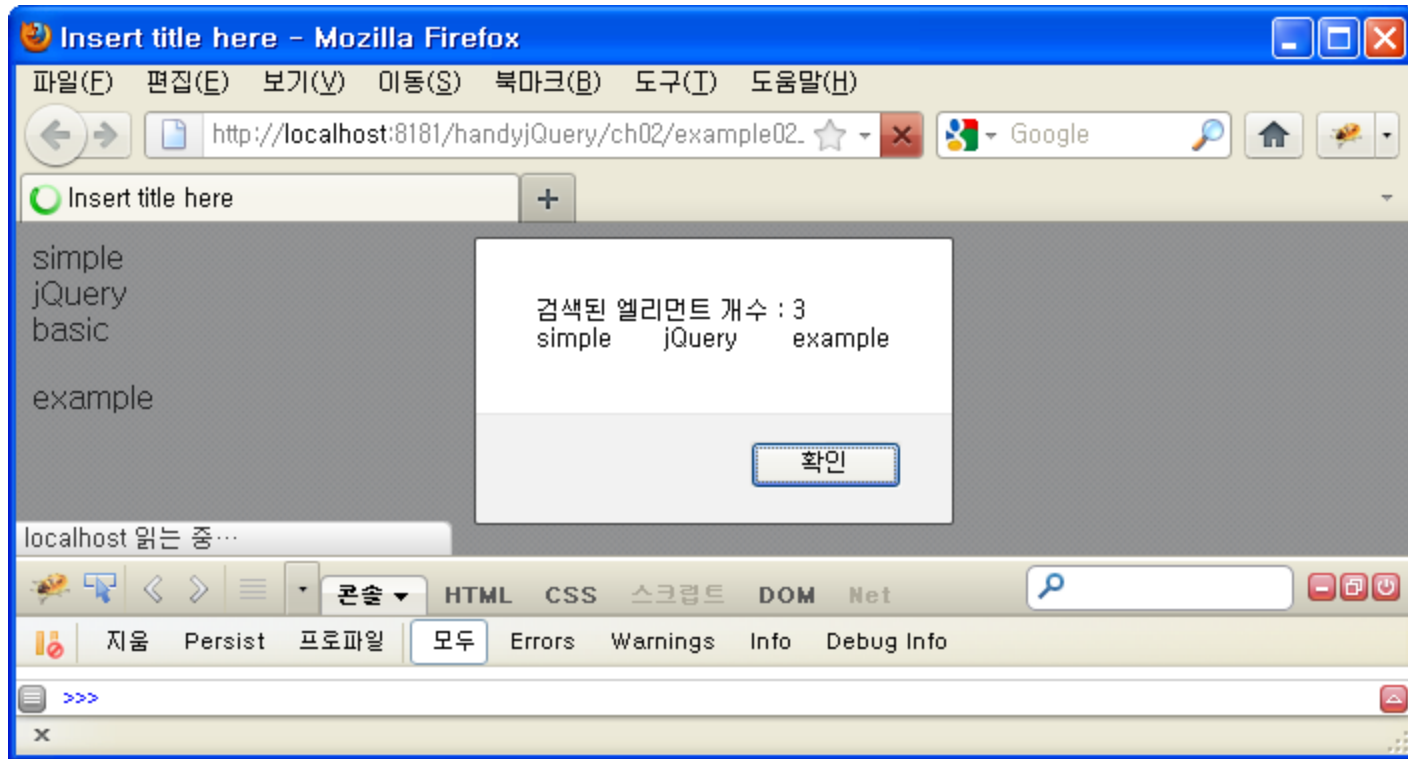
jQuery 집합에 인자로 설정한 클래스를 적용한다

# 엘리먼트에 접근해서 스타일 지정하기

```
<!DOCTYPE html> <html>
<head>
  <meta charset="UTF-8">
  <title>엘리먼트에 스타일 지정하기-"element", "#id", ".class"</title>
  <style type="text/css">
    .spotlight { background-color: #ff0 }
    .redtext { color: #f00 }
    .largetext { font-size: 30pt }
    .italictext{ font-style: italic; }
  </style>
  <script src="../js/jquery.js" type="text/javascript"> </script>
  <script type="text/javascript">
    $(document).ready(function() {
      $('span').addClass('redtext');
      $('div').addClass('spotlight');
      $('#simpletext1').addClass('largetext');
      $('.simpletext1').addClass('italictext');
    });
  </script>
</head>
<body>
  <span id="simpletext1">simple</span>
  <div class="simpletext1">jQuery</div>
  <span id="simpletext2">basic</span>
  <div id="simpletext3">example</div>
</body>
</html>
```



## 한 번에 다양한 엘리먼트에 접근하여 개수와 텍스트 얻기-1



## 한 번에 다양한 엘리먼트에 접근하여 개수와 텍스트 얻기-1

`$('선택터1, 선택터2, 선택터3 ...')`

`$('p, span, div. simpletext1');`

### 1. length, size()

- 선택터로 찾은 엘리먼트들의 묶음인 확장 집합은 배열과 유사
- 엘리먼트 개수를 얻기 위해서는 length나 size() 를 사용

### 2. each()

- 인자로 기술한 함수는 래퍼 집합에 속한 엘리먼트의 개수만큼 반복적으로 호출된다

### 3. \$(this)

- each() 함수 내에서 현재 접근 가능한 엘리먼트
- \$(this)와 같이 사용하여 엘리먼트를 jQuery 객체화하여 사용

### 4. text()

- 엘리먼트에 대해 텍스트 내용을 얻는다.

## 한 번에 다양한 엘리먼트에 접근하여 개수와 텍스트 얻기-1

```
$(document).ready(function() {  
    var resultText = "";  
    var $searchEles = $('p, span, div.simpletext1');  
  
    resultText+="검색된 엘리먼트 개수 : " + $searchEles.length+"Wn";  
  
    $searchEles.each(function() {  
        resultText+= $(this).text() + "Wt";  
    });  
    alert($.trim(resultText));  
});
```

## 한 번에 다양한 엘리먼트에 접근하여 개수와 텍스트 얻기-1

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>한 번에 다양한 엘리먼트에 접근하여 개수와 텍스트 얻기-"selector1, selector2,
selectorN"</title>
    <script src="../js/jquery.js" type="text/javascript"></script>
    <script type="text/javascript">
      $(document).ready(function() { });
      var resultText = "";
      var $searchEles = $('p,span,div.simpletext1');
      resultText+="검색된 엘리먼트 개수 : " + $searchEles.length+"Wn";
      $searchEles.each(function() {
        resultText+= $(this).text() + "Wt";
      });
      alert($.trim(resultText));
    </script>
  </head>
  <body>
    <span>simple</span>
    <div class='simpletext1'>jQuery</div>
    <div>basic</div>
    <p>example</p>
  </body>
</html>
```

# jQuery를 사용한 객체 확장

• \$.extend() 메서드

객체 생성 후 속성 추가 – 데이터 수가 적을 때는 편리

```
<script src="http://code.jquery.com/jquery-1.7.js"> </script>
```

```
<script>
```

```
    $(document).ready(function () {
```

```
        var object = {};
```

```
        object.name = 'RintlanTta';
```

```
        object.gender = 'Male';
```

```
        object.part = 'Second Guitar';
```

```
    });
```

```
</script>
```

# jQuery를 사용한 객체 확장

\$.extend() 메서드

많은 수의 속성 추가할 때 생기는 문제를 해결하는 메서드

사용법 -> \$.extend(object, addObject, addObject, ...)

```
<head>
```

```
<script src="http://code.jquery.com/jquery-1.7.js"> </script>
```

```
<script>
```

```
    $(document).ready(function () {           // 변수를 선언합니다.
```

```
        var object = { name: 'RintlanTta' }; // $.extend() 메서드를 사용합니다.
```

```
        $.extend(object, { gender: 'Male', part: 'Second Guitar' }); // 출력합니다.
```

```
        var output = '';
```

```
        $.each(object, function (key, item) {
```

```
            output += key + ': ' + item + '\n';
```

```
        });
```

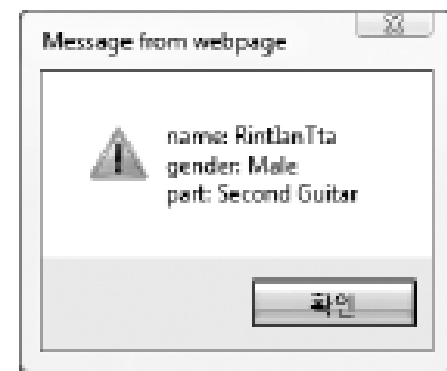
```
        alert(output);
```

```
    });
```

```
</script>
```

```
</head>
```

그림 13-18 객체의 결합



# jQuery 프레임워크 충돌 방지

\$.noConflict()

- 충돌을 방지할 때 사용하는 메서드

- \$.noConflict() 메서드 사용

  - 더 이상 jQuery의 식별자 \$를 사용할 수 없음

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<script src="http://code.jquery.com/jquery-1.7.js"> </script>
```

```
<script>
```

```
    // 플러그인 간의 충돌을 제거합니다.
```

```
    $.noConflict();
```

```
    var J = jQuery;
```

```
    // jQuery를 사용합니다.
```

```
    J(document).ready(function () {
```

```
        J('h1').removeClass('hight_light');
```

```
    });
```

```
</script>
```

```
</head>
```

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