

The background of the image shows a rural landscape with rolling hills covered in agricultural fields. Wind turbines are scattered across the hills, and a clear blue sky is visible above.

jQuery (ver3.4.0)

목 차

1. jQuery 소개
2. jQuery 프로그래밍
3. DOM 요소 탐색
4. DOM 객체 다루기
5. 이벤트 바인딩과 처리
6. Ajax 프로그래밍



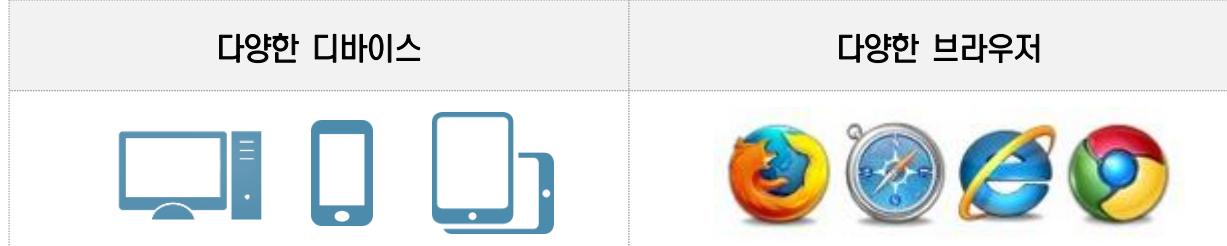
1. jQuery 소개

- 1.1 UI 개발과 UI 아키텍처
- 1.2 jQuery 개요
- 1.3 jQuery 소개
- 1.4 jQuery 사용하기
- 1.5 요약

1.1 UI 개발과 UI 아키텍처 [1/3]

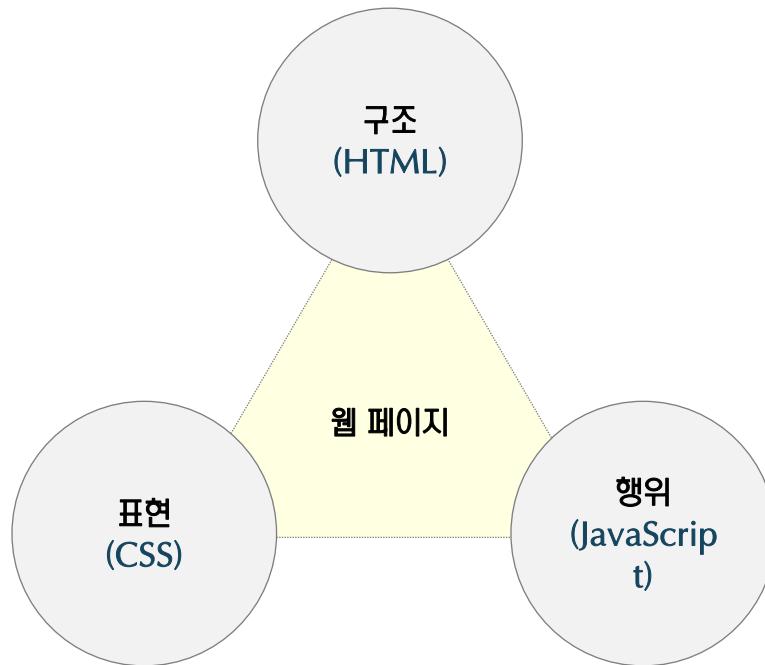
- ✓ 웹 기술은 발전에 발전을 거듭하여 우리 생활의 전반에 영향을 주고 있습니다.
- ✓ 웹의 활용 범위가 넓어지고 사용자 늘어 갈수록 사람들이 웹에 기대하는 수준이 높아지고 있습니다.
- ✓ 웹을 활용하는 디바이스는 PC 를 넘어서 스마트폰, 태블릿 등으로 다양해지고 있습니다.
- ✓ 오늘날 수준 높은 요구 사항, 다양한 디바이스 등으로 UI(User Interface) 기술 구조는 점점 더 복잡해지고 있습니다.

점점 복잡해져 가는 UI(User Interface) 기술들



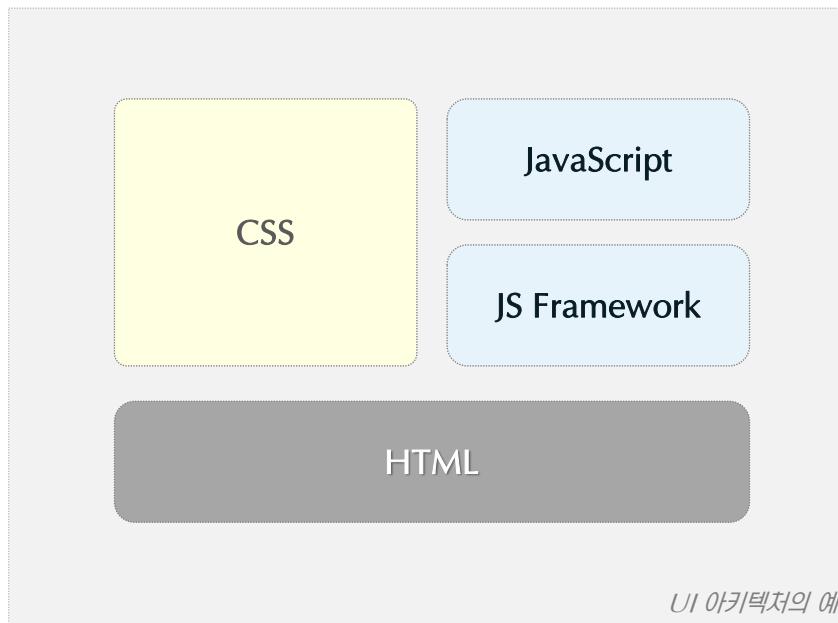
1.1 UI 개발과 UI 아키텍처 [2/3]

- ✓ 복잡하고 방대해진 UI 코드는 관리를 어렵게 합니다. 크로스 브라우징 및 멀티 디바이스는 코드 패편화를 가져옵니다.
- ✓ 이러한 문제는 웹 페이지의 구조(HTML)와 행위(JS) 그리고 표현(CSS)을 분리하면 어느 정도 해결할 수 있습니다.
- ✓ 또한, 검증된 프레임워크와 라이브러리를 사용하여 시스템의 안정성과 유지보수성을 높일 수 있습니다.
- ✓ W3C에서 제정한 웹 표준을 준수하면, 사용자의 웹 접근성을 높이고 코드를 분리할 수 있습니다.



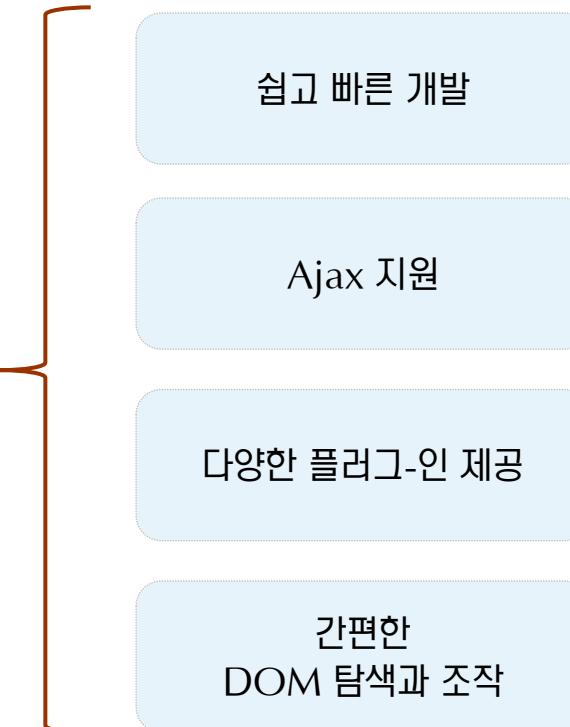
1.1 UI 개발과 UI 아키텍처 [3/3]

- ✓ 구조를 나타내는 HTML 코드에서 디자인 요소는 CSS 파일로 분리하고, 행위를 나타내는 코드는 JS 파일로 분리합니다.
- ✓ HTML 각 요소에서 이벤트 부분을 제거하고, JavaScript에서 동적으로 이벤트를 바인딩 합니다.
- ✓ HTML DOM 구조를 탐색하는 코드가 복잡해지는 경우, 이를 간편하게 줄여주는 라이브러리를 사용합니다.
- ✓ 동일한 화면을 해상도 및 디바이스 별로 중복 개발하지 않도록, 이를 지원하는 프레임워크를 사용합니다.



1.2 jQuery개요 [1/3]

- ✓ jQuery는 적은 코딩으로 많은 일을 할 수 있는 가벼운 자바 스크립트 라이브러리입니다.
- ✓ jQuery는 배우기 쉬우며 개발이 적용하기도 쉬워서 초보자도 어렵지 않게 배울 수 있습니다.
- ✓ 복잡한 Ajax 처리 및 DOM(Document Object Model) 탐색과 조작을 간편하게 처리할 수 있습니다.
- ✓ 다양한 jQuery 플러그-인(plug-in)을 제공할 뿐만 아니라, 사용자 정의 플러그-인도 만들 수 있습니다.



1.2 jQuery개요 [2/3]

- ✓ jQuery는 경량 JavaScript Library로, John Resig이 2006년 1월 Bar Camp에서 발표하였습니다.
- ✓ CSS가 디자인 특성을 HTML로부터 분리한 것처럼, jQuery는 행위 특성을 HTML로부터 분리합니다.
- ✓ jQuery는 무료로 사용할 수 있는 오픈 소스 소프트웨어이며 MIT 라이센스를 갖고 있습니다.
- ✓ Microsoft와 Nokia는 자사 제품에 jQuery를 포함시켰습니다.



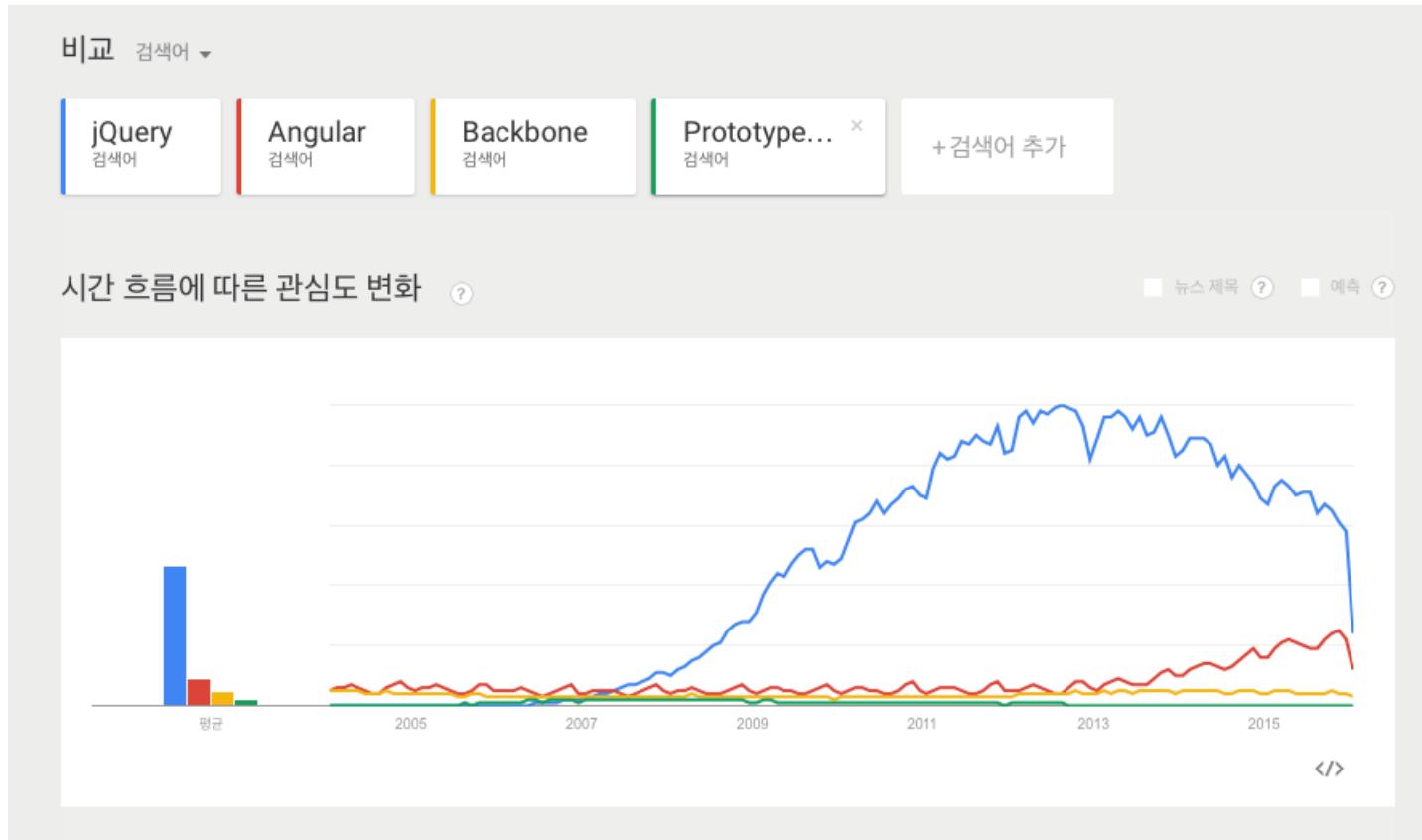
1.2 jQuery개요 [3/3]

- ✓ 선택자(Selector) 와 필터는 DOM 객체를 탐색하고, 다양한 함수는 DOM을 조작하고 스타일을 변경합니다.
- ✓ 웹 브라우저와 HTML 문서에 대한 이벤트 바인딩 및 이벤트 처리를 간편하게 할 수 있습니다.
- ✓ 브라우저마다 서로 다르고 복잡하던 Ajax 프로그래밍을 일관성 있고 단순하게 처리합니다.
- ✓ 다양한 JQueryEffect 함수를 제공하여, 동적인 웹 페이지를 쉽게 구현할 수 있습니다.



1.3 jQuery 소개 (1/3) – 트랜드 분석

- ✓ jQuery에 대한 관심도를 알아보기 위하여 Google Trend를 검색했습니다.
- ✓ jQuery와 비슷한 다른 JavaScript 라이브러리 또는 프레임워크를 비교 대상으로 지정했습니다.
- ✓ 2009년 이후로 jQuery는 다른 라이브러리보다 월등히 높은 관심도를 나타냅니다.
- ✓ 그만큼 jQuery는 웹 프론트 개발에서 대중적인 인기를 끌고 있는 JavaScript 라이브러리라 할 수 있습니다.



1.3 jQuery 소개 [2/3] – 특징

- ✓ jQuery 선택자(Selector)는 DOM을 탐색하고, 다양한 메소드를 제공하는 특별한 객체(WrapperSet)를 반환합니다.
- ✓ 래퍼세트 객체는 메소드 체인(Method Chain)을 제공하여, 메소드 호출에서의 반복적인 코딩을 줄입니다.
- ✓ 각 브라우저 별로 다르게 구현하는 이벤트 처리, DOM 조작 코드를 jQuery제공 메소드로 간단하게 해결합니다.
- ✓ 이미 개발된 많은 플러그-인을 사용하면 jQuery기능을 확장할 수 있습니다. 물론, 직접 만들 수도 있습니다.



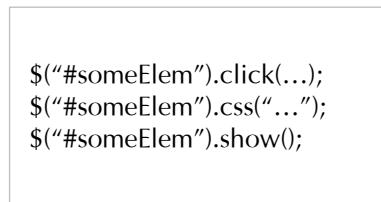
jQuerySelector는 탐색한 DOM 배열을 다루는 특별한 객체를 제공합니다. 이 객체를 래퍼세트(WrapperSet)라 합니다. jQuery래퍼집합 객체를 통하여 DOM을 다루는 다양한 함수를 수행할 수 있습니다.

jQuerySelector는 탐색결과가 없는 경우에도 null을 반환하지 않습니다.



선택자와 jQuery래퍼세트

`$(selector).action()`



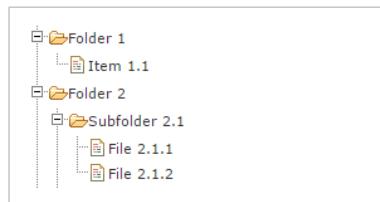
메소드 체인

`$('#someElem').click(...)
 .css('...')
 .show();`



Cross-browsing 지원

일관성 있는
이벤트 처리와 DOM 조작

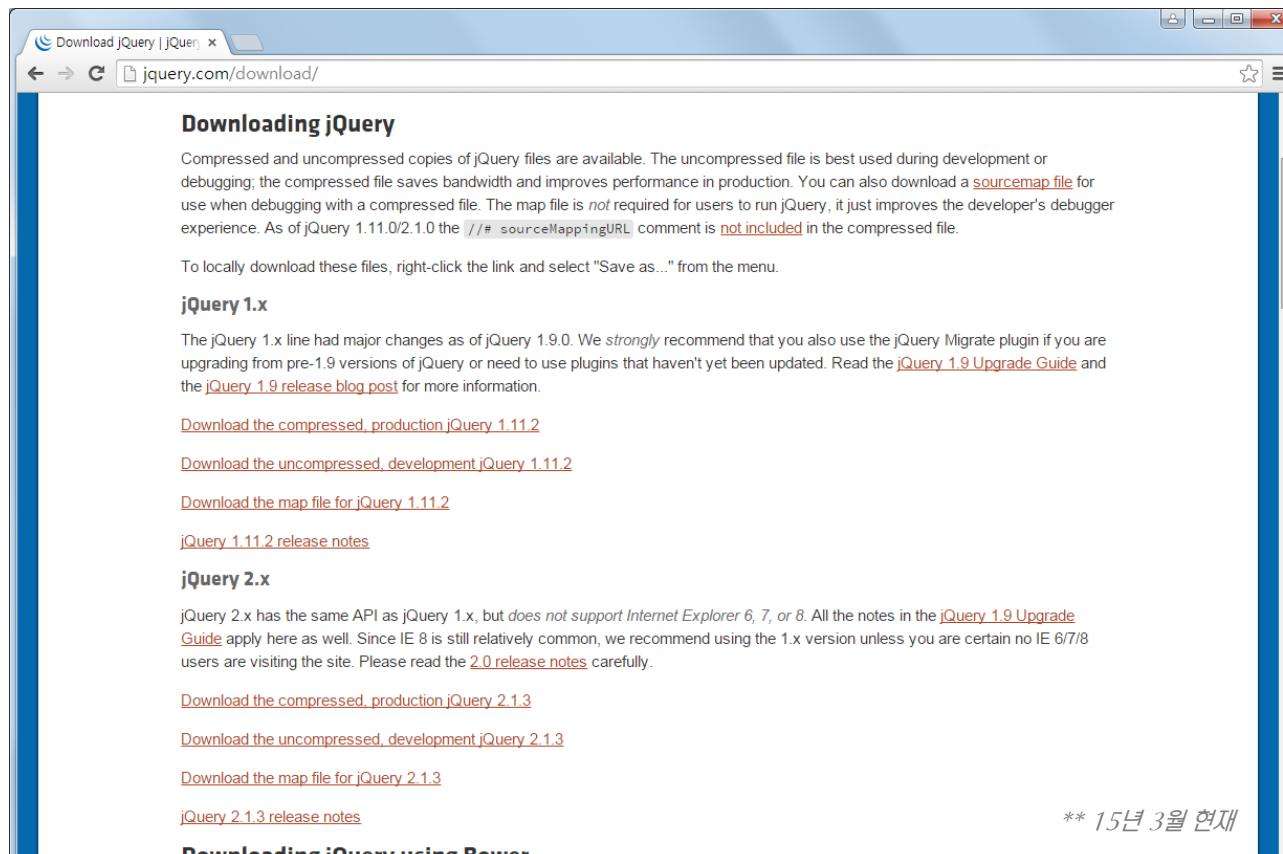


다양한 플러그-인 제공

`$("#filestructure").treeview();`

1.3 jQuery 소개 (3/3) – 버전

- ✓ jQuery 라이브러리는 다양한 브라우저를 지원하기 위해, 두 가지 버전으로 제공합니다.
- ✓ jQuery1.* 는 IE 8 이하 버전을 지원합니다. IE 8 이하 버전이 사라질 때 까지 계속 버전 업 예정입니다.
- ✓ jQuery2.* 는 HTML5에서 추가된 기능을 제공하며, IE 8 이하 버전을 지원하지 않습니다.
- ✓ 구형 브라우저까지 안정적으로 지원하려면 1.* 버전을 선택하고, HTML5 기능을 우선하려면 2.* 버전을 선택합니다.



출처 : <http://jquery.com/download>

1.4 jQuery 사용하기 (1/3) – jQuery 설치

- ✓ jQuery는 공식 사이트에서 다운로드 받아 사용하거나, CDN 을 사용합니다.
- ✓ CDN (Content Delivery Network) 은 컨텐츠를 네트워크를 통해 제공하는 방식입니다.
- ✓ jQuery를 제공하는 CDN은 Google, Microsoft, jQuery사이트 등 입니다.
- ✓ 네트워크 환경 및 운영 안정성을 위해서 CDN 보다는 다운로드 하여 사용하는 방식을 선호합니다.

```
<head>
  <script src="../Lib/jquery-3.0.0.min.js"></script>
</head>
```

다운로드하여 사용

```
<head>
  <script src="http://ajax.googleapis.com/ajax/libs/jquery/3.0.0/jquery.min.js"></script>
</head>
```

Google CDN

```
<head>
  <script src="http://ajax.aspnetcdn.com/ajax/jquery/jquery-3.0.0.min.js"></script>
</head>
```

Microsoft CDN

```
<head>
  <script src="http://code.jquery.com/jquery-3.0.0.min.js"></script>
</head>
```

jQueryCDN

1.4 jQuery 사용하기 (2/3) – 기본구문

- ✓ 기본 구문은 Selector 를 사용하여 DOM 객체를 탐색하고, 반환된 래퍼세트를 통해 함수를 수행합니다.
- ✓ Selector 표현식과 Action 메소드를 조합한 형태로 구문을 작성합니다. → ex) `$(selector).action();`
- ✓ jQuery로 DOM 을 탐색하기 전에, 웹 브라우저에 문서가 모두 로드되어 있어야 합니다.
- ✓ jQuery는 Document Ready 이후 처리할 수 있는 두 가지 방법을 제공합니다.

jQuery 기본 구문	설명
<code>\$(selector).action();</code>	<p><code>\$(selector)</code>는 탐색한 DOM 객체들을 담은 래퍼세트(WrapperSet)을 반환합니다. 래퍼세트 객체를 통해 기능을 처리하는 액션 메소드를 호출합니다.</p>

Document Ready 이후 처리 #1

```
$(document).ready(function(){  
    // Do something...  
});
```

Document Ready 이후 처리 #2

```
$(function(){  
    // Do something...  
});
```

1.4 jQuery 사용하기 (3/3) – jQuery 프로그래밍

- ✓ jQuery를 사용하여 DOM 객체에 이벤트를 바인딩하고 DOM 객체를 조작하는 예제입니다.
- ✓ jQuery CDN 을 사용하여 jQuery라이브러리를 문서에 포함합니다.
- ✓ \$(document).ready() 를 사용하여 문서를 로드 한 후, DOM 객체를 처리합니다.
- ✓ jQuery는 Selector 와 Action 메소드를 조합하여 기능을 수행합니다.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <script src="http://code.jquery.com/jquery-3.0.0.min.js">
5 </script>
6 <script>
7   $(function() {
8     $("button").click(function () {
9       $("p").hide();
10    });
11  });
12 </script>
13 </head>
14
15 <body>
16   <h2>이 부분은 타이틀입니다.</h2>
17   <p>이 부분은 단락입니다.</p>
18   <p>이 부분은 또 다른 단락입니다.</p>
19   <button>Click me</button>
20 </body>
21 </html>
```

01-sample.html

코드설명

[Line4] jQuery라이브러리를 문서에 포함합니다.
jQueryCDN에서 제공하는 라이브러리를 사용합니다.
[Line7] 문서가 모두 로드된 후에 ready() 안에 정의된
함수를 호출합니다.
[Line8] button 요소를 찾아 click 이벤트를 바인딩합
니다.
[Line9] (버튼이 클릭되면) 모든 p 요소를 찾아 숨깁니다.

실행결과

이 부분은 타이틀입니다.

이 부분은 단락입니다.

이 부분은 또 다른 단락입니다.

Click me

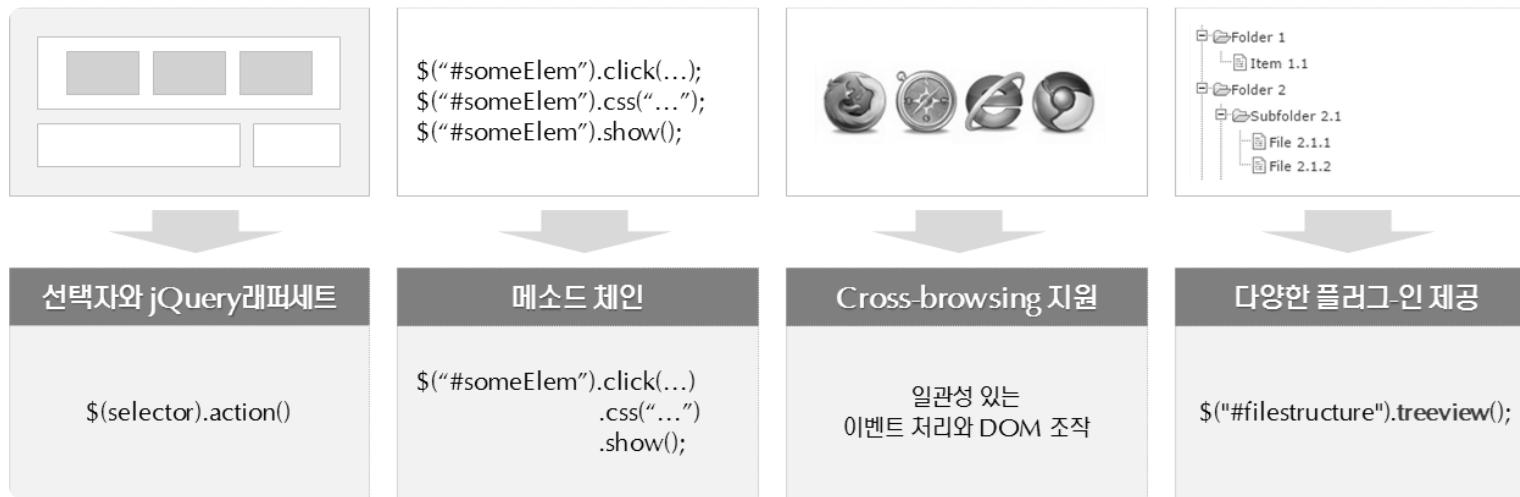
클릭하면 모든 p 요소를 숨깁니다.

이 부분은 타이틀입니다.

Click me

1.5 요약

- ✓ jQuery는 가벼운 자바스크립트 라이브러리로 DOM 조작, Ajax 지원 등 쉽고 빠른 개발을 지원합니다.
- ✓ jQuery 선택자는 DOM을 탐색하고, 다양한 메소드를 제공하는 래퍼세트(WrapperSet) 객체를 반환합니다.
- ✓ 래퍼세트 객체는 메소드 체인을 제공하여, 메소드 호출에서의 반복적인 코딩을 줄여줍니다.
- ✓ jQuery는 공식사이트에서 다운로드 하여 사용하거나, CDN에서 직접 사용하는 방법이 있습니다.





2. jQuery프로그래밍

2.1 jQuery프로그래밍

2.2 요약

2.1 jQuery프로그래밍 (1/4) – 고객관리 프로그램

- ✓ “namoosori” 주식회사는 수작업으로 고객정보를 관리하고 있습니다.
- ✓ 그러나, 사업이 번창하면서 늘어나는 고객정보를 효과적으로 관리하기 위한 프로그램이 필요하게 되었습니다.
- ✓ namoosori 주식회사를 위하여 수기로 작성하던 고객명부를 웹 기반으로 관리할 수 있는 프로그램을 작성해 봅니다.
- ✓ 고객관리 프로그램은 고객목록을 조회하고, 이름으로 고객정보를 찾는 기능을 제공합니다.

수작업으로 관리하던 고객명부

고객명	고객주소	전화번호
하영화	전남 장성	010-2222-2826
김민철	충남 아산	010-1234-8888
이화진	서울 관악구	017-134-6211
최윤호	함경도 영흥	016-111-2222
강형석	충남 천안	019-400-9877
...



이 많은 고객목록에서
원하는 정보를 찾는 일은 너무
힘들어요.. ㅠㅠ

웹 기반 고객관리 화면

▶ 고객목록

총 5명의 고객이 검색되었습니다.

이름 ▾ 검색

	이름(이메일)	주소	전화번호	고객등급
<input type="checkbox"/>	하영화(yhha@namoosori.com)	전남 장성	010-2222-2826	우수고객
<input type="checkbox"/>	김민철(mckim@namoosori.com)	충남 아산	010-1234-8888	일반고객
<input type="checkbox"/>	이화진(hjlee@namoosori.com)	서울 관악…	017-134-6211	일반고객
<input type="checkbox"/>	최윤호(yhchoi@namoosori.com)	서울 강남…	016-111-2222	우수고객
<input type="checkbox"/>	강형석(hskang@namoosori.com)	충남 천안	019-400-9877	일반고객

©2016 Namoosori. All rights reserved.

2.1 jQuery프로그래밍 (2/4) – HTML 소스

```
1 <div class="cntHeader mgt15">
2   <h3 class="tTit">고객목록</h3>
3   <div class="functionBar">
4     <q>총 <strong><span id="searchCount">5</span>명</strong>의 고객이 검색되었습니다.</q>
5   </div>
6 </div>
7 <div class="tWrap02">
8   <div class="tOption tHead">
9     <select id="searchType">
10    <option value="1" selected="selected">이름</option>
11  </select>
12  <input type="text" name="searchValue" value="" />
13  <a class="t21" id="searchBtn" href="javascript:;"><span>검색</span></a>
14 </div>
15 <table class="agrid01 fixed" id="memberTable">
16   <thead>
17     <tr>
18       <th class="first"><input type="checkbox" id="allCheck" ></th>
19       <th>이름(이메일)</th>
20       <th>주소</th>
21       <th>전화번호</th>
22       <th class="last">고객등급</th>
23     </tr>
24   </thead>
25   <tbody>
26     <tr data-name="하영화">
27       <td class="L"><input type="checkbox" /></td>
28       <td class="L">하영화(yhha@namoosori.com)</td>
29       <td>전남 장성</td>
30       <td>010-2222-2826</td>
31       <td><strong class="red">우수고객</strong></td>
32     </tr>
33     ...
34   </tbody>
34 </table>
</div>
```

01-customer.html

실행결과

▶ 고객 목록

총 5명의 고객이 검색되었습니다.

	이름(이메일)	주소	전화번호	고객등급
<input checked="" type="checkbox"/>	하영화(yhha@namoosori.com)	전남 장성	010-2222-2826	우수고객
<input type="checkbox"/>	김민철(mckim@namoosori.com)	충남 마산	010-1234-8888	일반고객
<input type="checkbox"/>	미화진(hjlee@namoosori.com)	서울 관악...	017-134-6211	일반고객
<input type="checkbox"/>	최윤호(yhchoi@namoosori.com)	서울 강남...	016-111-2222	우수고객
<input type="checkbox"/>	강현석(hskang@namoosori.com)	충남 천안	019-400-9877	일반고객

©2016 Namoosori. All rights reserved.

2.1 jQuery프로그래밍 (3/4) – 체크박스 모두선택/해제

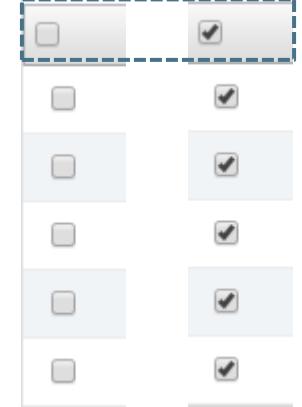
- ✓ 메시지 보내기 등과 같은 향후에 추가될 기능들은 특정 고객을 선택하여 처리해야 합니다.
- ✓ 전체 고객을 쉽게 선택할 수 있도록, 모두 선택 및 선택 취소 기능을 구현합니다.
- ✓ 먼저, 전체 선택용 체크박스(id가 allCheck)를 찾아 클릭 이벤트를 처리하는 함수를 연결합니다.
- ✓ 이벤트 처리함수에서는 allCheck 체크박스가 눌림 여부에 따라 테이블의 체크박스를 체크 또는 해제합니다.

```
1 <script>
2
3 $(function(){
4
5     // 체크박스 모두 선택 기능
6     $("#allCheck").click(function(){
7         if ($(this).is(":checked")) {
8             $("#memberTable tbody input:checkbox").prop("checked", true);
9         } else {
10             $("#memberTable tbody input:checkbox").prop("checked", false);
11         }
12     });
13 });
14
15 </script>
```

01-customer.html

실행결과

체크박스를 선택하면 모두 선택
체크박스를 해제하면 모두 해제



<input type="checkbox"/>	<input checked="" type="checkbox"/>

2.1 jQuery프로그래밍 (4/4) – 이름으로 고객 검색

- ✓ 고객의 이름을 입력하고 [조회] 버튼을 클릭하면 해당 고객을 찾아 목록에 표시하는 기능을 구현합니다.
- ✓ 단, 서버와 실제로 통신하지는 않으며, HTML에 있는 데이터를 숨기거나 보여주는 방식으로 처리합니다.
- ✓ 테이블에는 검색 결과를 목록으로 보여주고, 검색된 고객 수는 테이블 상단에 보여줍니다.
- ✓ show() 메소드와 hide() 메소드를 통해 DOM 객체를 보여주거나 숨길 수 있습니다.

```
1  $(function(){
2      // Attribute 선택자를 사용한 고객 검색기능
3      $("#searchBtn").click(function(){
4          var searchValue = $("input[name=searchValue]").val();
5
6          // 검색어가 입력되었을 때만 이름으로 필터링
7          if (searchValue) {           가    null      true
8              $("#memberTable tbody tr")  
                .hide()  
                .filter("[data-name='"+searchValue+"']").show();
9          } else {
10              $("#memberTable tbody tr").show();
11          }
12
13          // 검색된 고객 수 출력하기
14          var count = $("#memberTable tbody tr:visible").length;
15          $("#searchCount").text(count);
16      });
17  });
18 });
19});
```

01-customer.html

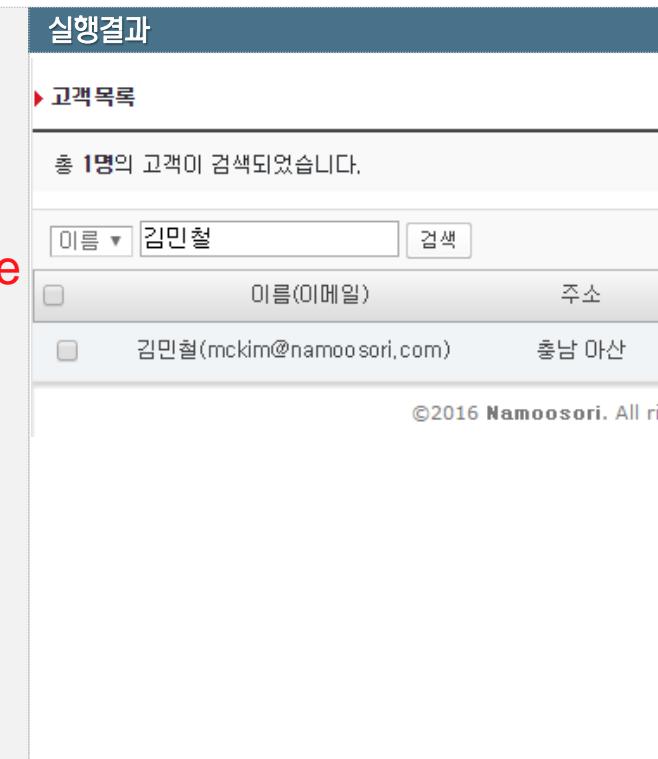
실행결과

▶ 고객 목록

총 1명의 고객이 검색되었습니다.

선택	이름(이메일)	주소
<input type="checkbox"/>	김민철(mckim@namoosori.com)	충남 마산

©2016 Namoosori. All rights reserved.



2.2 요약

- ✓ 고객을 관리하는 웹 기반 프로그램을 작성하면서, jQuery에서 제공하는 기능을 간단하게 살펴보았습니다.
- ✓ DOM 객체를 ID로 탐색하기 위하여 `$("#allCheck")`와 같이 jQuery선택자를 사용합니다
- ✓ 클릭 이벤트를 처리하기 위해 `click` 메소드를 호출하고, 원하는 처리를 하는 콜백 함수를 인자로 전달합니다.
- ✓ `show()` 메소드와 `hide()` 메소드는 DOM 객체를 화면에 보여주거나 숨길 수 있습니다

수직으로 관리하던 고객명부		
고객명	고객주소	전화번호
하영화	전남 장성	010-2222-2826
김민철	충남 아산	010-1234-8888
이화진	서울 관악구	017-134-6211
최윤호	함경도 영흥	016-111-2222
강형석	충남 천안	019-400-9877
...

웹 기반 고객관리 화면			
▶ 고객 목록			
총 5명의 고객이 검색되었습니다.			
이름	이메일	주소	전화번호
<input type="checkbox"/>	이름(이메일)	고객등급	
<input type="checkbox"/>	하영화(yhha@namoosori.com)	전남 장성	010-2222-2826
<input type="checkbox"/>	김민철(mckim@namoosori.com)	충남 아산	010-1234-8888
<input type="checkbox"/>	이화진(hjlee@namoosori.com)	서울 관악…	017-134-6211
<input type="checkbox"/>	최윤호(yhchoi@namoosori.com)	서울 강남…	016-111-2222
<input type="checkbox"/>	강형석(hskang@namoosori.com)	충남 천안	019-400-9877



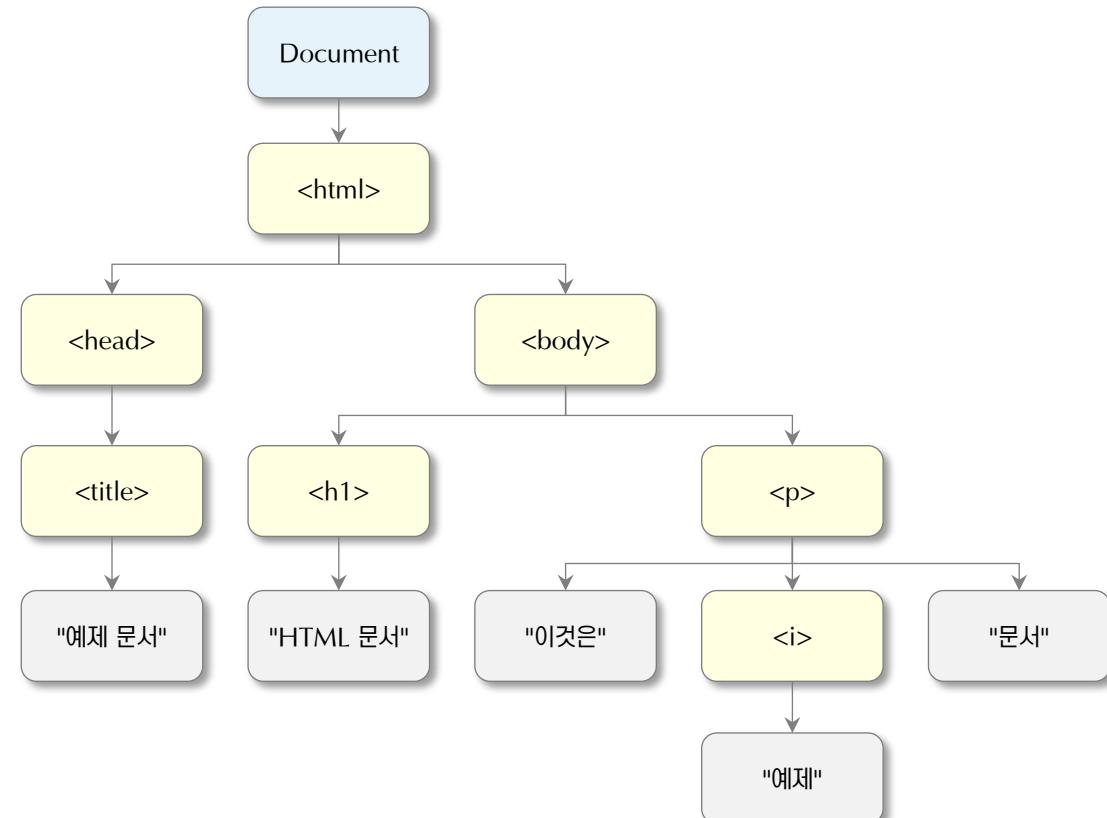
3. DOM 요소 탐색

- 3.1 DOM 탐색 개요
- 3.2 DOM 요소 탐색
- 3.3 DOM 계층구조 탐색
- 3.4 DOM 속성 탐색
- 3.5 DOM 요소 필터링
- 3.6 jQuery 메소드
- 3.7 요약

3.1 DOM 탐색 개요 (1/3) – DOM (Document Object Model)

- ✓ DOM(Document Object Model)은 HTML과 같이 구조화된 문서의 요소를 제어하는 API입니다.
- ✓ DOM은 HTML 문서 뿐만 아니라 XML 문서를 표현하고 조작하는 표준 API를 제공합니다.
- ✓ DOM은 문서 요소 집합을 트리 형태의 계층 구조로 표현합니다.
- ✓ DOM은 HTML 요소를 표현하는 노드와 문자열을 나타내는 노드 등을 포함합니다.

```
<html>
  <head>
    <title>예제 문서</title>
  </head>
  <body>
    <h1>HTML 문서</h1>
    <p>이것은<i>예제</i>문서</p>
  </body>
</html>
```



3.1 DOM 탐색 개요 [2/3] – DOM API

- ✓ DOM 객체는 JavaScript를 사용하여 DOM을 탐색하고 조작하는 API를 제공합니다.
- ✓ 주로 document 객체를 사용하여 요소의 id, name 또는 태그 이름을 이용하여 DOM 요소를 탐색합니다.
- ✓ 탐색한 DOM 요소 객체를 통해, 필요한 정보를 조회하거나 조작할 수 있습니다.
- ✓ 그러나, DOM 객체에서 제공하는 API를 사용하면, 단순한 작업임에도 적지 않은 코드를 작성해야 합니다.

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4   <span id="div1">div1 content</span>
5   <div>div2 content</div>
6   <span name="div3">div3 content</span>
7
8   <script>
9     var element1 = window.document.getElementById("div1");
10    var element2 = window.document.getElementsByTagName("div")[0];
11    var element3 = window.document.getElementsByName("div3")[0];
12
13    console.log(element1.innerHTML);
14    console.log(element2.innerHTML);
15    console.log(element3.innerHTML);
16  </script>
17 </body>
18 </html>
```

코드설명

[Line4] span 태그의 id를 div1로 정의합니다.
[Line6] span 태그의 name을 div3으로 정의합니다.
[Line9] document 객체의 getElementById 메소드를 사용하여 요소의 id 속성 값이 div1인 DOM 객체를 탐색하여 element1 변수에 할당합니다.
[Line10] document 객체의 getElementsByTagName 메소드를 사용하여 태그 이름이 div인 DOM 객체를 탐색하여 element2 변수에 할당합니다.
[Line11] document 객체의 getElementsByName 메소드를 사용하여 요소의 name 속성 값이 div3인 DOM 객체를 찾아 element3 변수에 할당합니다.
[Line13~15] 탐색한 DOM 객체의 innerHTML 프로퍼티 값을 콘솔에 출력합니다.

실행결과

```
div1 content
div2 content
div3 content
```

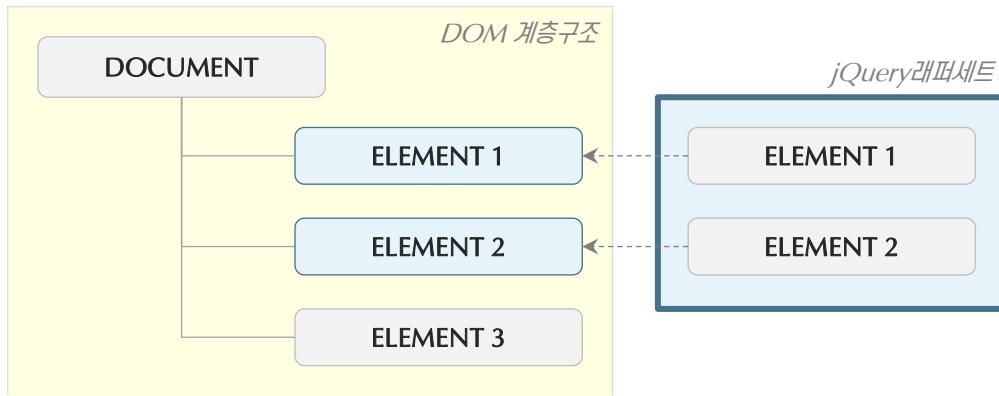
01-dom-api.html

3.1 DOM 탐색 개요 (3/3) – jQuery에서 DOM 탐색

- ✓ jQuery는 DOM 탐색을 위하여 CSS에서 사용하는 Selector 표현 방식을 사용합니다.
- ✓ 브라우저가 표준 CSS 선택자를 올바르게 구현하지 않았더라도, jQuery는 W3C 표준에 맞게 요소를 탐색합니다.
- ✓ jQuery Selector는 CSS Selector와 jQuery에서 정의한 다양한 방법으로 DOM 요소를 탐색합니다.
- ✓ jQuery는 DOM 탐색 결과로 래퍼세트(Wrapper Set)라는 DOM을 래핑한 객체를 반환합니다.

jQuerySelector 문법

`$(selector)` 또는 `jQuery(selector)`



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <script
5     src="http://code.jquery.com/jquery-3.0.0.min.js">
6 </script>
7 </head>
8 <body>
9     <span id="div1">div1 content</span>
10    <div>div2 content</div>
11    <span name="div3">div3 content</span>
12
13    <script>
14        var element1 = $("#div1");
15        var element2 = $("div").eq(0);
16        var element3 = $("[name=div3]").eq(0);
17
18        console.log(element1.html());
19        console.log(element2.html());
20        console.log(element3.html());
21    </script>
22 </body>
23 </html>
```

02-jquery-selector.html

3.2 DOM 요소 탐색 (1/5) – 요소 선택자

- ✓ jQuery 선택자(Selector)는 DOM 객체를 탐색하여 래퍼세트 객체로 반환합니다.
- ✓ jQuery 선택자에는 요소, ID, 클래스, 다중, 복합 그리고 모두 선택자 등이 있습니다.
- ✓ 요소 선택자(Element Selector)는 DOM 요소 명(태그 이름)이 일치하는 것들을 모두 찾습니다.
- ✓ DOM API 중 document.getElementsByTagName() 과 비슷합니다.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <script src="http://code.jquery.com/jquery-3.0.0.js"></script>
6
7 <script>
8
9 $(function () {
10   $("h1").css("color", "orange");
11 });
12
13 </script>
14 </head>
15 <body>
16   <h1>JavaScript</h1>
17   <p>JavaScript 언어를 공부합니다.</p>
18   <h1>jQuery</h1>
19   <p>jQuery라이브러리를 공부합니다.</p>
20 </body>
21 </html>
```

03-tag-selector.html

코드설명

[Line10] 요소 선택자를 사용하여 h1 태그명을 가진 DOM 객체를 탐색합니다. 탐색결과에 CSS 스타일로 글자색 orange를 적용합니다.

실행결과

JavaScript

JavaScript 언어를 공부합니다.

jQuery

jQuery 라이브러리를 공부합니다.

3.2 DOM 요소 탐색 [2/5] – ID 선택자

- ✓ ID 선택자(ID Selector)는 HTML DOM 객체 중에서 ID가 일치하는 요소를 찾아 하나만 반환합니다.
- ✓ ID 선택자는 다양한 선택자 중에서 DOM 객체를 가장 빨리 탐색합니다.
- ✓ ID는 한 페이지 내에서 중복되지 않는 것이 원칙이나 만일 ID가 중복된 경우, 첫 번째 요소를 반환합니다.
- ✓ DOM API 중 document.getElementById() 와 비슷합니다.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <script src="http://code.jquery.com/jquery-3.0.0.js"></script>
6
7 <script>
8
9 $(function () {
10   $("#target").css("color", "blue");
11 });
12
13 </script>
14 </head>
15 <body>
16   <h1>Header-0</h1>
17   <h1 id="target">Header-1</h1>
18   <h1>Header-0</h1>
19 </body>
20 </html>
```

04-id-selector.html

코드설명

[Line10] ID 선택자를 사용하여 target이라는 ID를 가진 DOM 객체를 탐색합니다. 탐색결과에 CSS 스타일로 글자색 blue를 적용합니다.

실행결과

Header-0

Header-1

Header-0

3.2 DOM 요소 탐색 [3/5] – 클래스 선택자, 복합 선택자

- ✓ 클래스 선택자(Class Selector)는 DOM 요소의 class가 일치하는 요소들을 찾아 모두 반환합니다.
- ✓ DOM API 중 document.getElementsByClassName() 과 비슷합니다. (IE8 이상에서만 지원)
- ✓ 복합 선택자(Complex Selector)는 여러 탐색 조건들을 조합한 선택자입니다. AND 조건을 표현합니다.
- ✓ 복합 선택자는 주로 Selector + Class Selector 조합으로 사용합니다.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <script src="http://code.jquery.com/jquery-3.0.0.js"></script>
6
7 <script>
8
9 $(function () {
10     $(".item").css("color", "gray");
11     $(".item.select").css("color", "orange");
12 });
13
14 </script>
15 </head>
16 <body>
17     <h1 class="item">Header-0</h1>
18     <h1 class="item select">Header-1</h1>
19     <h1 class="item">Header-0</h1>
20 </body>
21 </html>
22
```

05-class-selector.html

코드설명

[Line10] Class 선택자를 사용하여 item 이라는 클래스를 가진 DOM 객체를 탐색합니다. 탐색결과에 CSS 스타일로 글자색 gray를 적용합니다.

[Line11] 복합 선택자를 사용하여 item 과 select 클래스를 모두 갖는 DOM 객체를 탐색합니다. 탐색결과에 CSS 스타일로 글자색 orange를 적용합니다.

실행결과

Header-0

Header-1

Header-0

3.2 DOM 요소 탐색 (4/5) – 다중 선택자

- ✓ 다중 선택자(Multiple Selector)는 여러 Element 탐색 조건들을 나열한 선택자입니다.
- ✓ 콤마(,)로 여러 선택자들을 나열합니다. 예) \$(선택자1, 선택자2, 선택자3)
- ✓ 나열된 선택자를 하나라도 만족하는 DOM 객체를 찾아 반환합니다.
- ✓ 복합 선택자가 AND 조건을 나타내는 것과 달리, 다중 선택자는 OR 조건을 표현합니다.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <script src="http://code.jquery.com/jquery-3.0.0.js"></script>
6
7 <script>
8
9 $(function () {
10   $("h1, p").css("color", "orange");
11 });
12
13 </script>
14 </head>
15 <body>
16   <h1>JavaScript</h1>
17   <p>JavaScript 언어를 공부합니다.</p>
18   <h1>jQuery</h1>
19   <p>jQuery 라이브러리를 공부합니다.</p>
20 </body>
21 </html>
```

06-multiple-selector.html

코드설명

[Line10] 다중 선택자를 사용하여 h1, p 태그명을 가진 DOM 객체를 탐색합니다. 탐색결과에 CSS 스타일로 글자색 orange를 적용합니다.

실행결과

JavaScript

JavaScript 언어를 공부합니다.

jQuery

jQuery 라이브러리를 공부합니다.

3.2 DOM 요소 탐색 (5/5) – 전체 선택자

- ✓ 전체 선택자(All Selector)는 모든 요소를 탐색하여 jQuery 객체로 반환합니다.
- ✓ 주로 HTML 레이아웃을 확인하기 위한 용도로 사용합니다.
- ✓ 전체 선택자는 성능을 크게 저하시키므로 실무에서는 잘 사용하지 않습니다.
- ✓ 전체 선택자를 Universal Selector 라고도 합니다.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <meta charset="utf-8">
5  <script src="http://code.jquery.com/jquery-3.0.0.js"></script>
6
7  <script>
8
9  $(function () {
10    $("*").css("border", "1px solid red");
11  });
12
13 </script>
14 </head>
15 <body>
16   <h2>이 부분은 타이틀입니다.</h2>
17   <p>이 부분은 단락입니다.</p>
18   <p>이 부분은 또다른 단락입니다.</p>
19   <div>
20     <div id="content">이 부분은 내용입니다.</div>
21   </div>
22 </body>
23 </html>
```

코드설명

[Line10] All 선택자를 사용하여 문서 내의 모든 DOM 객체를 탐색합니다. 탐색결과에 CSS 스타일로 윤곽선 1px 굵기로 빨간색 실선을 적용합니다.

실행결과

이 부분은 타이틀입니다.

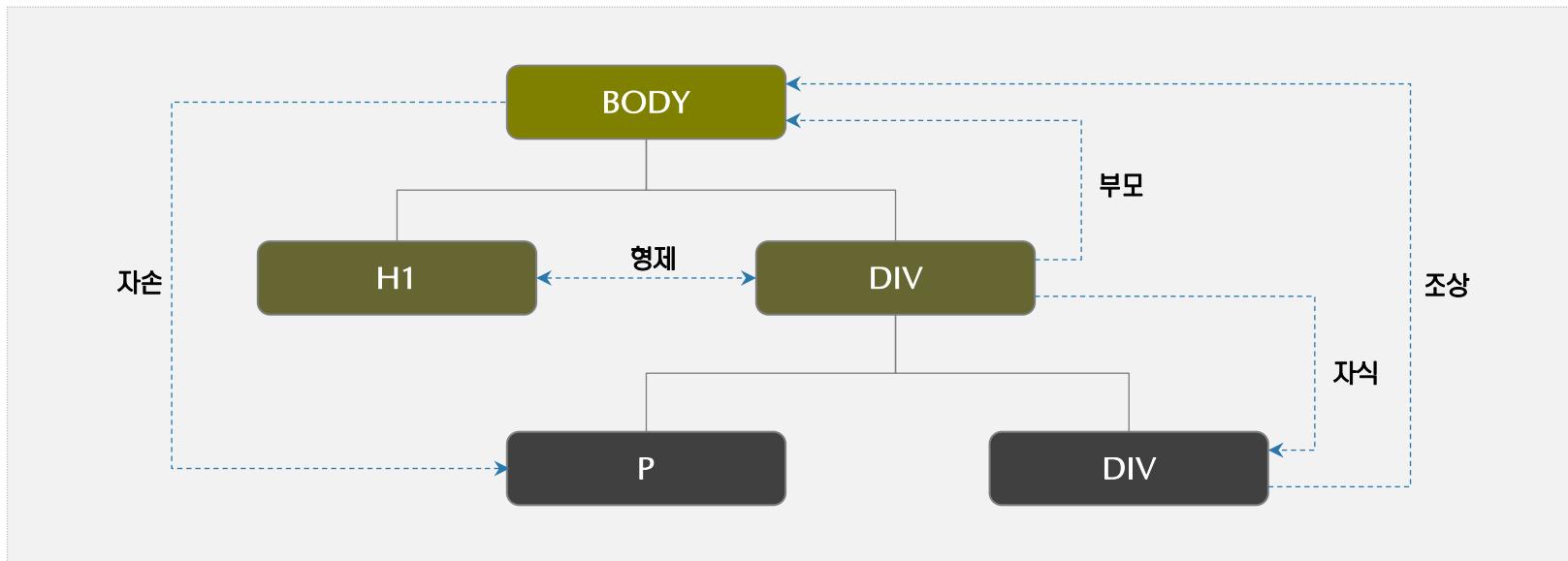
이 부분은 단락입니다.

이 부분은 또다른 단락입니다.

이 부분은 내용입니다.

3.3 DOM 계층구조 탐색 (1/2) – 개요

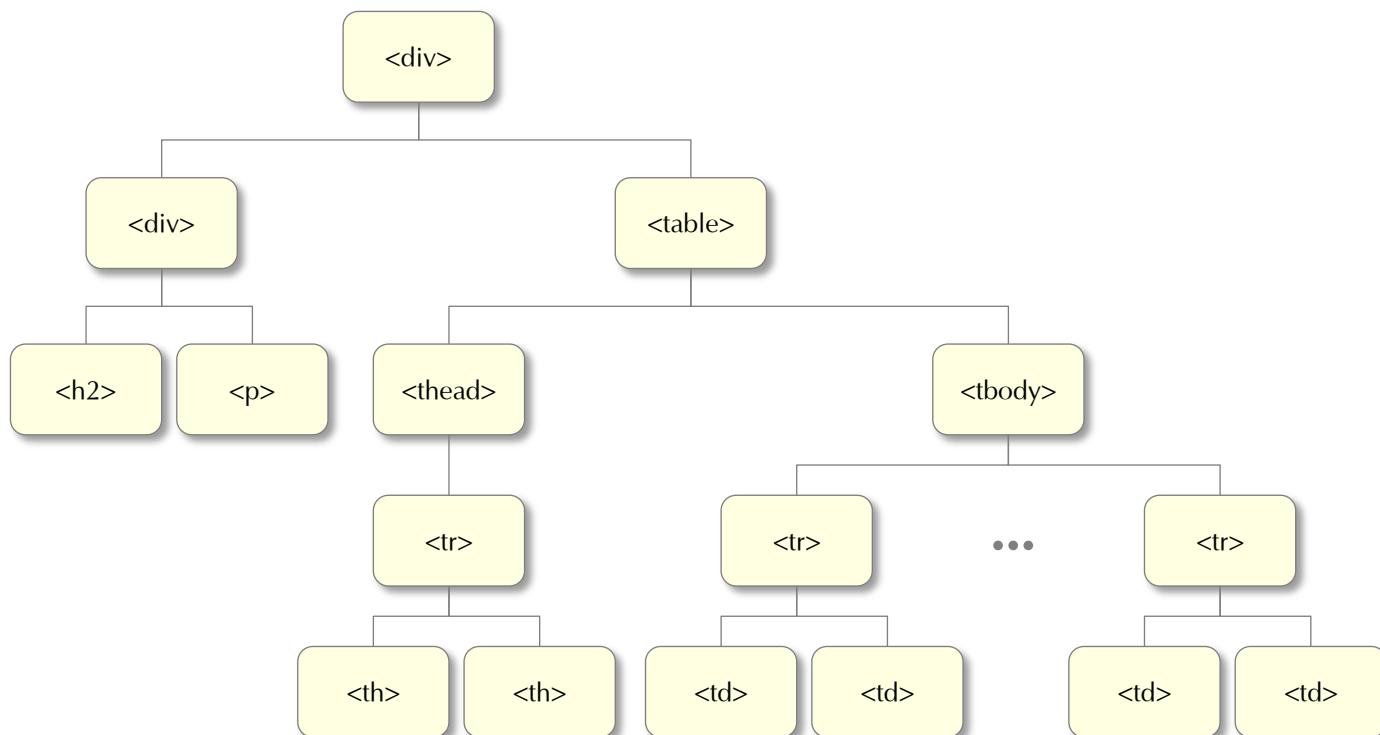
- ✓ jQuery는 HTML DOM 계층 구조에 접근하고 제어하는 쉬운 방법을 제공합니다.
- ✓ DOM을 탐색할 때, 트리 형태의 DOM 계층구조를 이해하는 것은 중요합니다.
- ✓ DOM 계층 구조가 나타내는 다양한 관계를 이해하면, DOM을 탐색하는데 도움이 됩니다.
- ✓ DOM 요소들은 수평적으로 형제요소, 수직적으로는 부모, 자식, 자손, 조상요소로 관계를 구분합니다.



부모요소 : 바로 인접한 상위요소
자식요소 : 바로 인접한 하위요소
형제요소 : 같은 부모를 갖는 요소
자손요소 : 직계자식을 포함한 모든 하위요소
조상요소 : 직계부모를 포함한 모든 상위요소

3.3 DOM 계층구조 탐색 (2/2) – jQuery선택자

- ✓ \$("Parent > Child")는 자식(Child) 선택자로, 바로 인접한 하위 자식들을 탐색합니다.
- ✓ \$("Ancestor Descendant")는 자손(Descendant) 선택자로, 모든 하위 자식들을 탐색합니다.
- ✓ \$("Prev + Next")는 인접(Next Adjacent) 선택자로, 인접한 바로 다음 형제를 탐색합니다.
- ✓ \$("Prev ~ Next")는 형제(Next Siblings) 선택자로, Next에 대한 모든 형제들을 탐색합니다.



```
<div>
  <div>
    <h2>DOM 계층구조 탐색</h2>
    <p>DOM 계층구조 탐색 jQuery선택자</p>
  </div>
  <table id="domTable">
    <thead>
      <tr>
        <th>이름</th><th>표현식</th>
      </tr>
    </thead>
    <tbody>
      <tr id="firstTr">
        <td>자식 선택자</td>
        <td><pre>$("A > B")</pre></td>
      </tr>
      <tr>
        <td>자손 선택자</td>
        <td><pre>$("A B")</pre></td>
      </tr>
      <tr>
        <td>인접 선택자</td>
        <td><pre>$("A + B")</pre></td>
      </tr>
      <tr>
        <td>형제 선택자</td>
        <td><pre>$("A ~ B")</pre></td>
      </tr>
    </tbody>
  </table>
</div>
```

08-dom-hierarchy.html

3.4 DOM 속성 탐색 [1/2] – 속성 선택자

- ✓ 속성 선택자(Attribute Selector)를 사용하여 DOM 요소가 가진 속성 값으로 DOM 객체를 탐색할 수 있습니다.
- ✓ 속성 선택자 표현식은 기본 선택자 바로 뒤에 나오는 대괄호([])안에 표현합니다.
- ✓ 기본 선택자는 선택사항입니다. 기본 선택자를 생략하면, 모든 요소에 대해 주어진 속성 선택자를 비교합니다.
- ✓ 예시) 아이피링크가 http://로 시작하는 a 태그를 탐색하려면?

속성 선택자 표현식	설명
<code>\$(Selector[attribute])</code>	attribute 속성을 갖는 Element
<code>\$(Selector[attribute = "Value"])</code>	attribute 속성 값이 Value와 일치하는 Element
<code>\$(Selector[attribute!= "Value"])</code>	attribute 속성 값이 Value와 일치하지 않는 Element
<code>\$(Selector[attribute~= "Value"])</code>	attribute 속성 값 중 단어가 Value로 시작하는 Element(공백과 함께 Value를 포함하는)
<code>\$(Selector[attribute^= "Value"])</code>	attribute 속성 값이 Value로 시작하는 Element
<code>\$(Selector[attribute\$= "Value"])</code>	attribute 속성 값이 Value로 끝나는 Element
<code>\$(Selector[attribute*= "Value"])</code>	attribute 속성 값이 Value를 포함하는 Element

출처 : http://www.w3schools.com/jquery/jquery_ref_selectors.asp

[예시] a 태그 중 href 속성 값이 http://로 시작하는 태그

```
$(‘a[href^=http://]’)
```

3.4 DOM 속성 탐색 [2/2] – 예제

- ✓ 입력 폼 요소(input)에 들어 있는 내용은 value 속성을 통해 접근할 수 있습니다.
- ✓ 다양한 속성 선택자를 사용하여 각 속성 선택자로 탐색한 입력 폼 요소의 배경색을 변경해 봅니다.

```
<body>
<h2>jQueryAttribute 셀렉터</h2>
<p>jQueryAttribute 셀렉터 테스트</p>
<div id="targets">
  <input type="text" name="case1" value="jQuery"/><br/>
  <input type="text" name="case2" value="jQuery 셀렉터"/><br/>
  <input type="text" name="case3" value="jQuery 셀렉터 테스트"/><br/>
  <input type="text" name="case4" value="셀렉터 테스트"/><br/>
  <input type="text" name="case5" value="셀렉터테스트"/><br/>
  <input type="text" name="case6" value="테스트"/><br/>
</div>
<div>
  <input type="button" id="button1" value="name 속성을 가짐"/>
  <input type="button" id="button2" value="jQuery' 일치"/>
  <input type="button" id="button3" value="jQuery' 미일치"/>
  <input type="button" id="button4" value="''셀렉터' 단어 포함"/>
  <input type="button" id="button5" value="''셀렉터' 포함"/>
  <input type="button" id="button6" value="''셀렉터'로 시작"/>
  <input type="button" id="button7" value="''셀렉터'로 끝남"/>
  <input type="button" id="btn_clear" value="clear"/>
</div>
</body>
```

09-attribute-selector.html

jQuery Attribute 셀렉터

jQuery Attribute 셀렉터 테스트

jQuery
jQuery 셀렉터
jQuery 셀렉터 테스트
셀렉터 테스트
셀렉터 테스트
테스트

```
$function () {
  $("#button1").click(function () {
    clear();
    $("input[name]").css("background", "silver");
  });
  $("#button2").click(function () {
    clear();
    $("input[value='jQuery']").css("background", "silver");
  });
  $("#button3").click(function () {
    clear();
    $("#targets > input[value!= 'jQuery']").css("background", "silver");
  });
  $("#button4").click(function () {
    clear();
    $("input[value~= '셀렉터']").css("background", "silver");
  });
  $("#button5").click(function () {
    clear();
    $("#targets > input[value*= '셀렉터']").css("background", "silver");
  });
  $("#button6").click(function () {
    clear();
    $("input[value^= '셀렉터']").css("background", "silver");
  });
  $("#button7").click(function () {
    clear();
    $("input[value$= '셀렉터']").css("background", "silver");
  });
  $("#btn_clear").click(function () {
    clear();
  });
  function clear() {
    $("input[type='text']").css("background", "white");
  }
}
```

3.5 DOM 요소 필터링 (1/6) – 필터 선택자

- ✓ 필터 선택자는 DOM 요소를 탐색한 결과에서 원하는 요소를 걸러내기 위하여 사용합니다.
- ✓ 기본 선택자는 선택사항입니다. 필터 선택자는 기본 선택자 뒤에 콜론(:) 기호와 함께 표기합니다.
- ✓ 필터 선택자에는 입력 폼 유형을 선택하는 필터 선택자와, 요소의 특성으로 선택하는 필터 선택자가 있습니다.
- ✓ 이 밖에도, jQuery에는 위치 기반 필터 선택자와 함수 기반 필터 선택자가 있습니다.

필터 선택자	설 명	필터 선택자	설 명
요소:button	type이 button인 input 요소 또는 button 요소 선택	요소:checked	체크된 입력 폼 선택
요소:checkbox	type이 checkbox인 input 요소 선택	요소:disabled	비활성화된 입력 폼 선택
요소:file	type이 file인 input 요소 선택	요소:enabled	활성화된 입력 폼 선택
요소:image	type이 image인 input 요소 선택	요소:focus	초점이 맞추어진 입력 폼 선택
요소:password	type이 password인 input 요소 선택	요소:input	모든 입력 폼 선택 (input, textarea, select, button)
요소:radio	type이 radio인 input 요소 선택	요소:selected	option 객체 중 선택된 요소 선택
요소:reset	type이 reset인 input 요소 선택	요소:hidden	감춰진 요소 선택
요소:submit	type이 submit인 input 요소 선택	요소:visible	보이는 요소 선택
요소:text	type이 text인 input 요소 선택	요소:header	헤더 엘리먼트만 선택 (<h1>~<h6>)

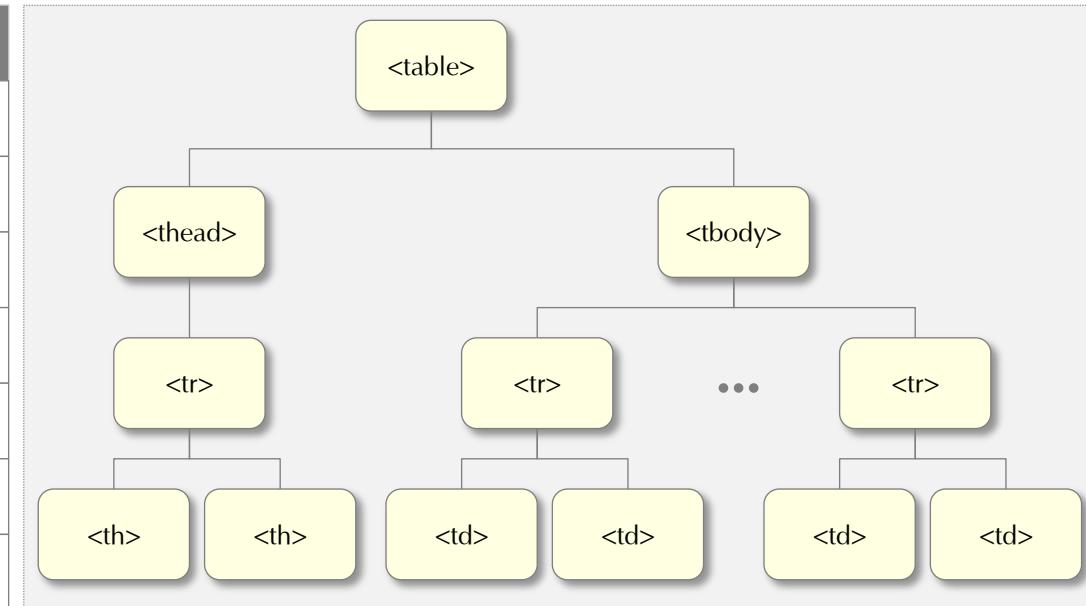
`$('input:checkbox')` : 입력 폼이 체크박스인 요소
`$(':radio')` : 라디오버튼 형식의 입력 폼 요소

`$(':radio:checked')` : 라디오버튼 중 체크 된 요소
`$(':checkbox:checked:enabled')` : 체크박스 중 활성화되고 체크된 요소

3.5 DOM 요소 필터링 [2/6] – 위치기반 선택자 [1/2]

- ✓ 위치 기반 필터 선택자는 선택한 요소들의 위치를 기반으로 필터를 수행합니다.
- ✓ [Quiz1] 아래 테이블을 표현한 DOM 객체에서 짝수 번째 행의 배경색을 silver 색상으로 적용해 봅시다.
- ✓ [Quiz2] 홀수 번째 행의 배경색을 yellow 색상으로 적용해 봅시다.
- ✓ [Quiz3] 테이블 첫 번째 행의 배경색을 black으로, 폰트 색상을 white로 적용해 봅시다.

필터 선택자	설명
:first	첫 번째 요소
:last	마지막 요소
:first-child	첫 번째 자식 요소
:last-child	마지막 자식 요소
:only-child	형제가 없는 모든 요소
:even	짝수 번째 요소
:odd	홀수 번째 요소



3.5 DOM 요소 필터링 [3/6] – 위치기반 선택자 [2/2]

- ✓ 아래는 위치 기반 선택자를 적용한 결과입니다.
- ✓ :even 필터 선택자로 짝수 번째 행을 선택하여, 배경색을 silver 색상으로 적용합니다.
- ✓ :odd 필터 선택자로 홀수 번째 행을 선택하여, 배경색을 yellow 색상으로 적용합니다.
- ✓ :first 필터 선택자로 테이블 첫 번째 행을 선택하여, 배경색을 black으로, 글자색을 white로 적용합니다.

HTML 구조

```
<table id="domTable">
  <thead>
    <tr>
      <th>제목</th><th>내용</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>A1</td><td>A2</td>
    </tr>
    <tr>
      <td>B1</td><td>B2</td>
    </tr>
    <tr>
      <td>C1</td><td>C2</td>
    </tr>
    <tr>
      <td>D1</td><td>D2</td>
    </tr>
  </tbody>
</table>
```

10-location-filter.html

위치 기반 필터 선택자 적용

```
$(function(){
  // #1. 짝수 번째 행 배경색(silver)
  $("#domTable tr:even").css("background", "silver");

  // #2. 홀수 번째 행 배경색(yellow)
  $("#domTable tr:odd").css("background", "yellow");

  // #3. 첫 번째 행 배경색(black), 글자색(white)
  $("#domTable tr:first").css("background", "black")
    .css("color", "white");
});
```

실행결과

제목 내용
A1 A2
B1 B2
C1 C2
D1 D2



제목	내용
A1	A2
B1	B2
C1	C2
D1	D2

3.5 DOM 요소 필터링 [4/6] – 함수기반 필터 선택자

- ✓ jQuery는 함수 형태의 필터 선택자 표현식을 제공합니다.
- ✓ :not() 필터 선택자는 주어진 선택자와 일치하지 않는 모든 요소를 선택합니다.
- ✓ :nth-child() 필터 선택자는 CSS에서 개념을 가져온 것으로 n번째 자식 요소를 선택합니다.
- ✓ :eq() 필터 선택자는 n번째로 일치하는 요소를 선택합니다.

필터 선택자	설명
:not(filter)	주어진 선택자와 일치하지 않는 모든 요소를 선택
:contains(foo)	텍스트 foo를 포함하는 요소만 선택
:nth-child(n)	n번째 자식 요소를 선택 (li:nth-child(2)는 두 번째, 3n+1이면 4, 7, 11...)
:eq(n)	n번째로 일치하는 요소를 선택
:gt(n)	n번째(포함하지 않음) 이후 요소를 선택
:lt(n)	n번째(포함하지 않음) 이전 요소를 선택
:has(f)	주어진 선택자와 일치하는 하나 이상의 요소를 포함하는 요소를 선택



nth-* 계열 필터는 CSS에서 개념을 가져온 것으로 n 값은 1부터 시작하는 인덱스를 사용합니다. 그러나, eq와 같은 필터는 n 값이 0부터 시작하는 JavaScript의 0-기반 인덱스를 사용합니다.

3.5 DOM 요소 필터링 [5/6] – 함수기반 필터 선택자 예제 [1/2]

실행결과

jQuery 필터 셀렉터

jQuery 필터 셀렉터의 종류를 나열하였습니다.

이름	표현식	설명
button 필터	:button	모든 버튼 선택
checkbox 필터	:checkbox	체크박스 엘리먼트 선택
체크상태 필터	:checked	선택된 체크박스
비활성화 필터	:disabled	비활성화 엘리먼트 선택
활성화 필터	:enabled	활성화 엘리먼트 선택
파일 필터	:file	파일 엘리먼트 선택

HTML 구조

```
<h2>jQuery필터 셀렉터</h2>
<p>jQuery필터 셀렉터의 종류를 나열하였습니다.</p>
<div>
  <table>
    <tr><th>이름</th><th>표현식</th><th>설명</th></tr>
    <tr><td>button 필터</td>
      <td>:button</td>
      <td>모든 버튼 선택</td>
    </tr>
    <tr><td>checkbox 필터</td>
      <td>:checkbox</td>
      <td>체크박스 엘리먼트 선택</td>
    </tr>
    <tr><td>체크상태 필터</td>
      <td>:checked</td>
      <td>선택된 체크박스</td>
    </tr>
    <tr><td>비활성화 필터</td>
      <td>:disabled</td>
      <td>비활성화 엘리먼트 선택</td>
    </tr>
    <tr><td>활성화 필터</td>
      <td>:enabled</td>
      <td>활성화 엘리먼트 선택</td>
    </tr>
    <tr><td>파일 필터</td>
      <td>:file</td>
      <td>파일 엘리먼트 선택</td>
    </tr>
  </table>
</div>
```

11-function-filter.html

함수 기반 필터 선택자 적용

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

3.5 DOM 요소 필터링 [6/6] – 함수기반 필터 선택자 예제 [2/2]

실행결과

jQuery 필터 셀렉터

jQuery 필터 셀렉터의 종류를 나열하였습니다.

이름	표현식	설명
button 필터	:button	모든 버튼 선택
checkbox 필터	:checkbox	체크박스 엘리먼트 선택
체크상태 필터	:checked	선택된 체크박스
비활성화 필터	:disabled	비활성화 엘리먼트 선택
활성화 필터	:enabled	활성화 엘리먼트 선택
파일 필터	:file	파일 엘리먼트 선택

함수 기반 필터 선택자 적용

```
$(function(){
    $("td:not(.myClass)").css("background", "#A9A9A9");
});
```

HTML 구조

```
<h2>jQuery필터 셀렉터</h2>
<p>jQuery필터 셀렉터의 종류를 나열하였습니다.</p>
<div>
    <table>
        <tr><th>이름</th><th>표현식</th><th>설명</th></tr>
        <tr><td class="myClass">button 필터</td>
            <td>:button</td>
            <td>모든 버튼 선택</td>
        </tr>
        <tr><td>checkbox 필터</td>
            <td class="myClass">:checkbox</td>
            <td>체크박스 엘리먼트 선택</td>
        </tr>
        <tr><td>체크상태 필터</td>
            <td>:checked</td>
            <td class="myClass">선택된 체크박스</td>
        </tr>
        <tr><td>비활성화 필터</td>
            <td class="myClass">:disabled</td>
            <td>비활성화 엘리먼트 선택</td>
        </tr>
        <tr><td class="myClass">활성화 필터</td>
            <td>:enabled</td>
            <td>활성화 엘리먼트 선택</td>
        </tr>
        <tr><td>파일 필터</td>
            <td class="myClass">:file</td>
            <td>파일 엘리먼트 선택</td>
        </tr>
    </table>
</div>
```

12-function-filter.html

3.6 jQuery메소드 (1/10) – 래퍼세트와 메소드

- ✓ jQuery는 선택자를 통해 탐색한 DOM 객체들을 래퍼세트이라는 특별한 배열 객체에 담아 반환합니다.
- ✓ jQuery는 선택된 DOM 객체가 없는 경우에도, 비어있는 래퍼세트 객체를 반환합니다.
- ✓ 래퍼세트 객체에는 내포된 DOM 객체들을 처리하는 다양한 메소드가 있습니다.
- ✓ 이러한 jQuery 메소드는 플러그-인 확장을 통해 추가할 수 있습니다.

메소드	반환 값	설 명
size()	요소 개수	래퍼세트의 요소 개수 반환
get(index)	DOM 요소	래퍼세트에서 인덱스 번호에 위치하는 DOM 객체 반환
index(element)	인덱스 번호	래퍼세트에서 해당 요소의 인덱스 번호 반환
add(expression)	래퍼세트	expression으로 명시한 요소를 래퍼세트에 추가
not(expression)	래퍼세트	expression으로 명시한 요소를 래퍼세트에서 제거
each(function(index, element))	이전 래퍼세트	래퍼세트의 각 요소마다 function을 수행
filter(expression)	래퍼세트	expression에 명시한 요소를 필터링
slice(begin, end)	래퍼세트	현재 래퍼세트의 일부분으로 새로운 래퍼세트를 생성하여 반환

3.6 jQuery메소드 (2/10) – 계층 구조 탐색

- ✓ jQuery에는 DOM 요소간의 관계를 이용하여 새로운 래퍼세트를 획득하는 함수가 있습니다.
- ✓ 메소드 호출 시 인자로 선택자를 제공하면, 해당 조건을 만족하는 새로운 래퍼세트를 생성할 수 있습니다.
- ✓ filter() 함수가 래퍼세트 요소들을 걸러내는 반면,
계층 구조를 탐색하는 메소드들은 DOM 계층구조에 있는 요소들로 새로운 래퍼세트를 생성합니다.

메소드	설명
parent([selector])	래퍼세트에 포함된 각 요소의 부모요소로 구성된 래퍼세트 반환
parents([selector])	래퍼세트에 포함된 각 요소의 조상요소로 구성된 래퍼세트 반환
children([selector])	래퍼세트에 포함된 각 요소의 자식요소로 구성된 래퍼세트 반환
prev([selector])	래퍼세트에 포함된 각 요소의 바로 이전 형제요소로 구성된 래퍼세트 반환
prevAll([selector])	래퍼세트에 포함된 각 요소의 이전에 나오는 형제요소들로 구성된 래퍼세트 반환
next([selector])	래퍼세트에 포함된 각 요소의 바로 이후 형제요소로 구성된 래퍼세트 반환
nextAll([selector])	래퍼세트에 포함된 각 요소의 이후에 나오는 형제요소들로 구성된 래퍼세트 반환
siblings([selector])	래퍼세트의 각 요소를 제외한 모든 형제요소들로 구성된 래퍼세트 반환

3.6 jQuery메소드 (3/10) – 요소 반복 (1/2)

- ✓ jQuery.each() 함수는 배열이나 객체를 반복적으로 처리할 때 사용합니다.
- ✓ 첫 번째 인자에는 자바스크립트 배열이나 래퍼세트 객체가 올 수 있습니다.
- ✓ 두 번째 인자에는 각 요소를 반복하면서 처리할 콜백(call back)함수를 정의합니다.
- ✓ 콜백 함수는 두 개의 매개변수(index:배열 인덱스, item:반복하는 요소객체)를 갖습니다.

배열 반복처리 함수

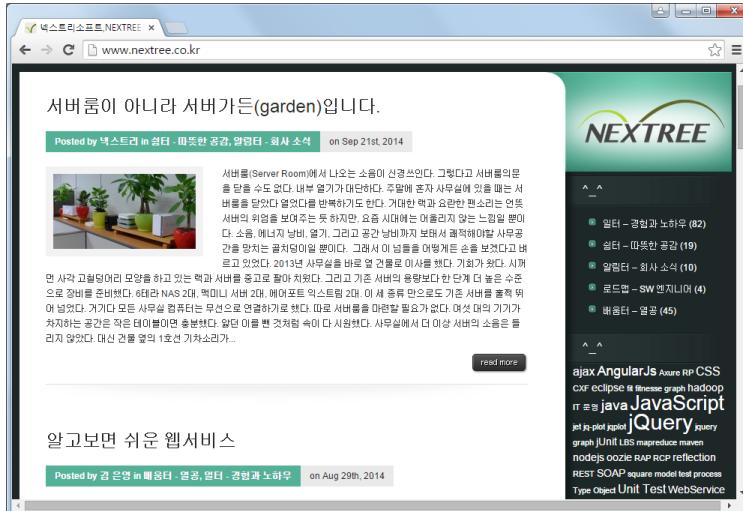
```
$.each(array, function(index, item){});
```

실행결과

[Nextree](#)

[KOSTA](#)

[Google](#)



```
<script>
$(function () {
    var array = [
        { name: 'NEXTREE',
            link: 'http://www.nextree.co.kr' },
        { name: 'KOSTA',
            link: 'http://edu.kosta.or.kr' },
        { name: 'Google',
            link: 'http://www.google.com' }
    ];
    var outHtml = '';

    // array 배열을 탐색하여 outHtml 생성
    $.each(array, function (index, item) {
        outHtml += '<a href="' + item.link + '">';
        outHtml += '    <h2>' + item.name + '</h2>';
        outHtml += '</a>';
    });
    document.body.innerHTML = outHtml;
});
```

21-each-array.html

3.6 jQuery메소드 [4/10] – 요소 반복 [2/2]

- ✓ jQuery선택자의 결과인 래퍼세트는 탐색한 DOM 요소들을 반복하는 each() 메소드를 제공합니다.
- ✓ \$.each() 함수와 달리 반복대상을 의미하는 첫 번째 인자가 없으며, 래퍼세트의 DOM 요소들이 반복 대상입니다.
- ✓ 래퍼세트의 each() 메소드는 유일한 인자인 콜백 함수가 있습니다.
- ✓ 콜백 함수는 두 개의 매개변수(index:배열 인덱스, item:반복하는 요소객체)를 갖습니다.

래퍼세트의 each() 메소드

```
$(selector).each(function(index, item){});
```

실행결과

첫번째

두번째

세번째

네번째

다섯번째

```
<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>
    $(function() {
        // h2 태그마다 high-light-0~4 클래스로 CSS 적용
        $("h2").each(function (index, item) {
            // item 대신 this 사용 가능
            $(item).addClass("high-light-" + index);
        });
    });
</script>
</head>
<body>
    <h2>첫번째</h2>
    <h2>두번째</h2>
    <h2>세번째</h2>
    <h2>네번째</h2>
    <h2>다섯번째</h2>
</body>
</html>
```

22-each-dom.html

3.6 jQuery 메소드 (5/10) – 필터 함수 (1/2)

- ✓ filter() 는 래퍼세트에 포함된 DOM 요소를 주어진 조건으로 걸러내는 메소드입니다.
- ✓ filter() 메소드는 기존 래퍼세트에서 DOM 요소를 축소한 새로운 래퍼세트를 반환합니다.
- ✓ filter() 메소드 인자에는 필터링 조건을 나타내는 선택자나 함수가 올 수 있습니다.

래퍼세트의 filter() 메소드

```
$(selector).filter(selector);
$(selector).filter(function(index, item){});
```

실행결과

첫번째

두번째

세번째

네번째

다섯번째

여섯번째

```
<script>
$(function() {
/*
 $("h2:even").css({
    background: "black",
    color: "white"
});
*/
// h2 태그의 짝수 또는 짝수에 CSS 색상 적용
$("h2").filter(":even").css({
    background: "black",
    color: "white"
});
</script>
</head>
<body>
<h2>첫번째</h2>
<h2>두번째</h2>
<h2>세번째</h2>
<h2>네번째</h2>
<h2>다섯번째</h2>
<h2>여섯번째</h2>
</body>
</html>
```

23-filter-selector.html

3.6 jQuery메소드 (6/10) – 필터 함수 (2/2)

- ✓ 필터 선택자를 사용하여 필터링 조건을 나타내기 어려운 경우, 함수를 사용할 수 있습니다.
- ✓ 필터링 조건을 함수로 사용한 경우, filter() 메소드는 함수 반환 값이 true인 요소만 새로운 래퍼세트에 포함합니다.
- ✓ filter() 메소드를 호출하여 필터링한 래퍼세트에서 end() 메소드를 호출하면 이전 래퍼세트로 돌아갑니다.

```
1 $(function() {  
2     // 필터 Selector 적용이 어려운 경우  
3     // 3n번째 h2 엘리먼트에 CSS 색상 적용  
4     $("h2").filter(function (index) {  
5         return index % 3 == 0;  
6     }).css({  
7         background: "black",  
8         color: "white"  
9     });  
10});
```

24-filter-function.html

실행결과

첫번째
두번째
세번째
네번째
다섯번째
여섯번째

코드설명

[Line4~6] 요소가 3n 번째인 경우 true를 반환합니다. true가 반환되면 filter() 함수는 새로운 래퍼세트에 해당 요소를 포함합니다.
[Line6~9] css() 메소드에 객체를 전달하여 하나 이상의 스타일 속성을 한번에 적용합니다.

```
1 $(function() {  
2     // 필터 Selector 적용한 경우  
3     /*  
4         $("h2").css("background", "silver");  
5         $("h2:even").css("color", "white");  
6         $("h2:odd").css("color", "red");  
7     */  
8     // 메소드 체인 적용  
9     // 위의 내용을 filter 메소드를 사용하여 구현  
10    $("h2").css("background", "silver")  
11        .filter(":even").css("color", "white").end()  
12        .filter(":odd").css("color", "red");  
13});
```

25-method-chain.html

실행결과

첫번째
두번째
세번째
네번째
다섯번째
여섯번째

코드설명

[Line10] h2 요소를 선택한 래퍼세트에 CSS를 적용합니다.
[Line11] 확장 접합에서 짹수 번째 요소만 남기고 필터링 합니다. CSS를 적용하고 end() 메소드를 호출하여 필터링 이전 상태로 래퍼세트를 돌립니다.
[Line12] 홀수 번째 요소로 필터링하여 CSS를 적용합니다.

3.6 jQuery메소드 (7/10) – 위치기반 함수

- ✓ 위치 기반 함수는 HTML DOM 객체의 특정 위치를 선택할 수 있는 함수입니다.
- ✓ eq() 메소드는 래퍼세트에서 주어진 인덱스번호에 해당하는 DOM 요소를 감싼 새로운 래퍼세트를 반환합니다.
- ✓ first() 메소드는 래퍼세트의 첫 번째 DOM 요소를 감싼 새로운 래퍼세트를 반환합니다.
- ✓ last() 메소드는 래퍼세트의 마지막 DOM 요소를 감싼 새로운 래퍼세트를 반환합니다.

위치 기반 함수

```
$(selector).eq(index);  
$(selector).first();  
$(selector).last();
```

실행결과

첫번째

두번째

세번째

네번째

다섯번째

여섯번째

```
$($function() {  
  
    // first() 함수를 사용하여 첫번째 h2 태그 색상 변경  
    // last() 함수를 사용하여 마지막 h2 태그 색상 변경  
    // eq() 함수를 사용하여 세번째 h2 태그 색상 변경  
    $("h2").first().css("background", "blue")  
        .css("color", "white");  
    $("h2").last().css("background", "black");  
    $("h2").eq(2).css("background", "silver");  
    $("h2").eq(-1).css("color", "white");  
  
});
```



eq() 메소드에 -1과 같이 음수 값(-n)을 제공하는 경우,
확장 집합 요소의 맨 뒤에서부터 n번째 요소를 선택합니다.

[26-location-function.html](#)

3.6 jQuery메소드 [8/10] – 래퍼세트에 요소 추가/삭제

- ✓ 앞서 살펴본, 필터 함수와 위치 기반 함수는 래퍼세트에 포함된 요소를 축소하기 위해 사용합니다.
- ✓ jQuery는 래퍼세트에 요소를 추가하거나 특정 요소를 삭제하는 메소드를 제공합니다.
- ✓ add() 메소드는 래퍼세트에 주어진 조건을 만족하는 요소를 추가한 새로운 래퍼세트를 반환합니다.
- ✓ not() 메소드는 래퍼세트에서 주어진 조건에 해당하는 요소를 제거한 새로운 래퍼세트를 반환합니다.

\$("p")	.add("h2")	.not("p")
<p>jQuery 래퍼세트 p</p>	<p>jQuery 래퍼세트 p h2</p>	<p>jQuery 래퍼세트 h2</p>
p 요소가 선택된 래퍼세트	h2 요소가 추가된 래퍼세트	p 요소가 제거된 래퍼세트

HTML 구조

```
<h2>래퍼세트 요소 추가 및 삭제</h2>
<p>add()는 래퍼세트에 요소를 추가하고, not()은 제거합니다.</p>
```

27-add-not.html

실행결과

확장집합 요소 추가 및 삭제

add()는 확장집합에 요소를 추가하고, not()은 제거합니다.

\$("p")

[<p>add()는 확장집합에 요소를 추가하고, not()은 제거합니다.</p>]

\$("p").add("h2")

[<h2>확장집합 요소 추가 및 삭제</h2>,

<p>add()는 확장집합에 요소를 추가하고, not()은 제거합니다.</p>]

\$("p").add("h2").not("p")

[<h2>확장집합 요소 추가 및 삭제</h2>]

3.6 jQuery 메소드 (9/10) – DOM 요소 판별

- ✓ `is()` 메소드는 기존 래퍼세트가 주어진 선택자와 일치하는지 여부를 반환합니다.
- ✓ 다른 메소드들이 새로운 래퍼세트 객체를 반환하는 반면, `is()` 메소드는 boolean 값을 반환합니다.
- ✓ 비교할 조건을 선택자로 표현하기 어려운 경우, 함수를 사용할 수 있습니다.
- ✓ 비교 조건을 나타내는 함수 호출 결과가 `true`인 경우, `is()` 메소드는 `true`를 반환합니다.

래퍼세트의 `is()` 메소드

```
$(selector).is(selector);
$(selector).is(function(index, item){});
```

실행결과

체크박스1

체크박스2

체크박스3

```
<script>
$(document).ready(function () {
    // 체크박스 중 name0이 checkedFalse이고
    // 체크되어 있으면 체크를 해제함
    $("input[type='checkBox']").each(function () {
        if ($(this).is("[name='checkedFalse']")
            && $(this).is(":checked")) {

            $(this).attr("checked", false);
        }
    });
}</script>
</head>

<body>
<h3><input type="checkBox" name="checkedFalse"
checked="checked"/>체크박스1</h3>
<h3><input type="checkBox" checked="checked"/>체크박스2</h3>
<h3><input type="checkBox" checked="checked"/>체크박스3</h3>
</body>
</html>
```

28-is.html

3.6 jQuery 메소드 (10/10) – 자손 요소 탐색

- ✓ `find()` 메소드는 래퍼세트의 모든 요소들에 대하여 주어진 선택자를 만족하는 모든 자손 요소를 선택합니다.
- ✓ `find()` 메소드는 선택자를 통해 탐색한 DOM 요소들을 새로운 래퍼세트로 반환합니다.
- ✓ 래퍼세트의 `filter()` 메소드는 래퍼세트 요소를 걸러내기 위해 사용합니다.
- ✓ 그러나, `find()` 메소드는 래퍼세트 요소들의 하위 자손들을 탐색하기 위하여 사용합니다.

래퍼세트의 `find()` 메소드

```
$(selector).find(selector);
```

실행결과

첫번째

두번째

세번째

네번째

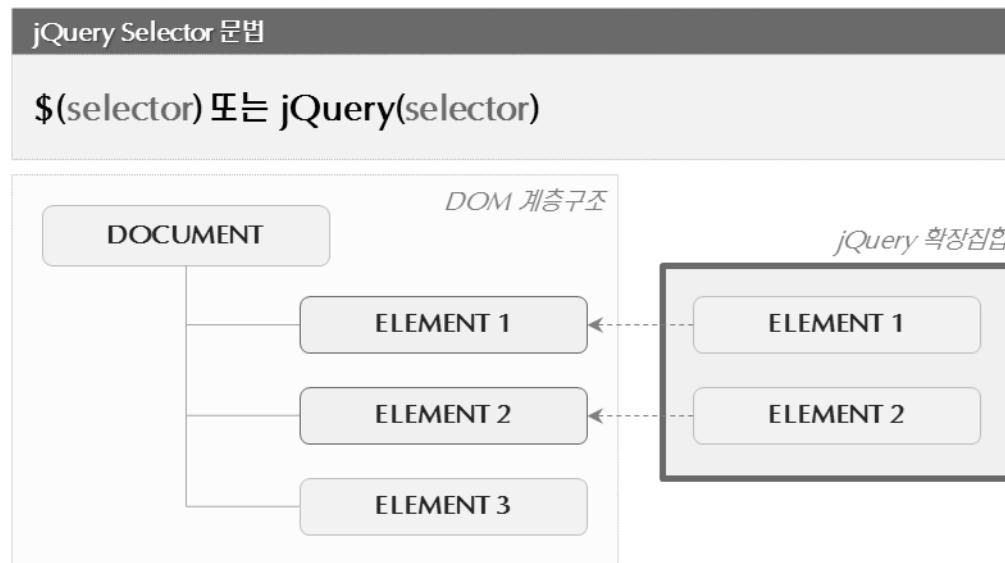
div 태그안

```
<script>
// findArea 하위 요소 중
// class가 myClass인 요소들의 배경색을 silver로 변경
$(function() {
    $("#findArea").find(".myClass")
        .css("background", "silver");
});
</script>
</head>
<body>
    <div id="findArea">
        <h1>첫번째</h1>
        <h2 class="myClass">두번째</h2>
        <h3 class="myClass">세번째</h3>
        <h4>네번째</h4>
        <div><h4 class="myClass">div 태그안</h4></div>
    </div>
</body>
</html>
```

29-find.html

3.7 요약

- ✓ jQuery는 DOM 탐색을 위하여 CSS에서 사용하는 Selector 표현 방식을 사용합니다.
- ✓ jQuery는 선택자를 통해 HTML DOM 계층 구조에 접근하고 제어하는 쉬운 방법을 제공합니다.
- ✓ 필터 선택자는 DOM 요소를 탐색한 결과에서 원하는 요소를 걸러내기 위하여 사용합니다
- ✓ jQuery 래퍼세트 객체에는 내포된 DOM 객체들을 처리하는 다양한 메소드가 있습니다.





4. DOM 객체 다루기

-
- 4.1 DOM 객체 제어 개요
 - 4.2 DOM 특성 제어
 - 4.3 DOM 내부 제어
 - 4.4 DOM 추가 및 삭제
 - 4.5 DOM 객체 삽입
 - 4.6 jQuery안티 패턴
 - 4.7 jQueryEffect
 - 4.8 요약

4.1 DOM 객체 제어 개요

- ✓ 순수 자바스크립트만을 이용하여 DOM 객체 구조를 처리하는 것은 어렵고 성가신 작업입니다.
- ✓ jQuery 메소드를 사용하면, DOM 객체를 보다 쉽게 다룰 수 있습니다.
- ✓ jQuery는 DOM 요소의 속성이나 class, style 을 제어하는 DOM 특성제어 메소드를 제공합니다.
- ✓ 이 밖에도, DOM 내부제어 메소드, DOM 추가/삭제 메소드와 DOM 객체를 삽입하는 메소드가 있습니다.

DOM 특성제어	DOM 내부제어	DOM 추가 및 삭제	DOM 객체삽입
attr() removeAttr() addClass() removeClass() toggleClass() css()	html() text()	\$() remove() empty() clone()	append() appendTo() prepend() prependTo() after() insertAfter() before() insertBefore()

4.2 DOM 특성 제어 (1/4) – 속성 제어 메소드

- ✓ attr(name) 메소드는 래퍼세트의 첫 번째 요소에 대한 속성(attribute) 값을 반환합니다.
- ✓ attr(name) 메소드는 해당 속성이 없으면 undefined를 반환합니다.
- ✓ attr(name, value), attr(object), attr(name, function) 메소드는 확장 집합의 모든 요소에 속성 값을 설정합니다.
- ✓ removeAttr(name) 메소드는 DOM 요소에서 해당 속성을 제거합니다.

속성 값 조회 메소드	속성 값 설정 메소드
<pre>\$(selector).attr(name);</pre>	<pre>\$(selector).attr(name, value); \$(selector).attr(name, function(index, attr){}); \$(selector).attr(object);</pre>
속성 삭제 메소드	<pre>\$(selector).removeAttr(name);</pre>

4.2 DOM 특성 제어 (2/4) – 속성 제어 예제

```
<script>
$(function() {
    // img 요소를 선택하여 src 속성 값을 메시지 창으로 띄움
    var srcValue = $("img").attr("src");
    alert("방법1:"+srcValue);

    $("img").each(function () {
        var value = $(this).attr("src");
        alert("방법2:"+value);
    });
    // 모든 img 요소의 폭과 높이를 200으로 변경
    $("img").attr({ "width": 200, "height" : 200});
    alert("폭과 높이를 200으로 변경함");

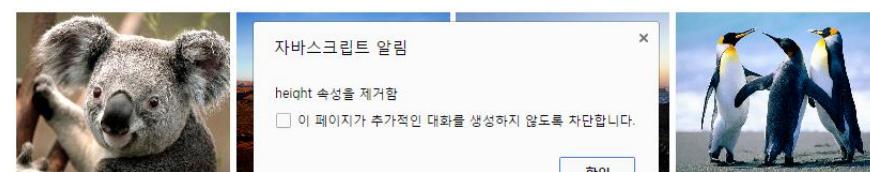
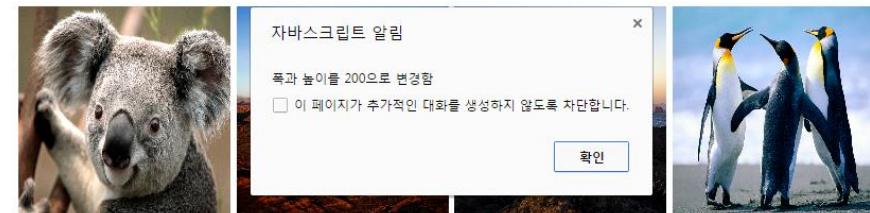
    // 모든 img 요소의 높이(height) 속성을 제거함
    $("img").removeAttr("height");
    alert("height 속성을 제거함");
    // 각 img 요소의 폭과 높이를 100, 200, 300, 400으로 변경
    $("img").attr("width", function (index) {
        return (index + 1) * 100;
    });
}
</script>
</head>
<body>
    
    
    
    
</body>
</html>
```

01-attr.html

실행결과

[방법1] 알림창은 첫 번째 img 요소에 대해서만 팝업됨

[방법2] 알림창은 모든 img 요소에 대해서 팝업됨



4.2 DOM 특성 제어 [3/4] – Class 속성 제어

class 속성 값 추가 및 삭제 메소드

```
$(selector).addClass('className');  
$(selector).removeClass('className');  
$(selector).toggleClass('className');
```

실행결과

첫번째	클릭 시 빨간색으로 변경	첫번째
두번째		두번째
세번째		세번째
네번째		네번째

```
<style>
    .orange { color: orange; }
    .blue { color: blue; }
    .red { color: red; }
</style>
<script>
$(function() {
    // '두번째' 헤더의 글자색을 orange로 변경
    $("#orange").addClass("orange");
    // '세번째' 헤더의 글자색을 blue로 변경
    $("h3[name='blue']").addClass("blue");
    // '네번째' 헤더의 red 클래스를 삭제
    $(".red").removeClass("red");
    // '첫번째' 헤더를 클릭할 때마다 글자색이 검은색과 빨간색으로 변경
    $("h1").click(function () {
        $(this).toggleClass('red');
    });
});
</script>
</head>
<body>
    <h1>첫번째</h1>
    <h2 id="orange">두번째</h2>
    <h3 name="blue">세번째</h3>
    <h4 class="red">네번째</h4>
</body>
</html>
```

02-class.html

4.2 DOM 특성 제어 (4/4) – Style 속성 제어

- ✓ `css(name)` 메소드는 래퍼세트의 첫 번째 요소에 대한 `style` 속성 값을 반환합니다.
- ✓ `css(name, value)` 메소드는 래퍼세트 내 모든 DOM 요소에 `style`을 설정합니다.
- ✓ `css(name, function)` 메소드는 `style` 속성 값을 바로 지정하는 대신, 함수를 통해 `style` 속성 값을 설정합니다.
- ✓ 여러 `style` 속성 값을 한번에 설정하려면 `css(object)`와 같이 객체를 인자로 사용합니다.

[사용법] style 속성 값 조회 메소드

```
$(selector).css(styleName);
```

[사용법] style 속성 값 설정 메소드

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

[CSS] 글자색을 클래스로 설정

```
<style type="text/css">
    .class1 { color: red; }
    .class2 { color: blue; }
    .class3 { color: green; }
    .class4 { color: orange; }
</style>
```

[HTML] 클래스를 다르게 설정한 h2 요소들

```
<h2 class="class1">첫번째</h2>
<h2 class="class2">두번째</h2>
<h2 class="class3">세번째</h2>
<h2 class="class4">네번째</h2>
```

[jQuery] 선택한 h2 요소들 중 첫 번째 요소의 style 속성 값 출력

```
var color = $("h2").css("color");
alert(color);
```

[jQuery] 선택한 모든 h2 요소들에 대하여 style 속성 값 설정

```
$("h2").css("color", "blue");
```

[jQuery] 함수를 사용한 style 속성 값 설정

```
var colors = ["silver", "lime"];
$("h2").css("color", function (index) {
    return colors[index % 2];
});
```

[jQuery] 객체를 사용하여 한번에 여러 속성 값 설정

```
var colors = ["silver", "lime"];
$("h2").css({
    color: function (index) {
        return colors[index % 2];
    },
    backgroundcolor: "black"
});
```

4.3 DOM 내부 제어 (1/2) – HTML과 텍스트 조회

- ✓ `html()` 과 `text()` 는 DOM 객체 내부를 조회하는 메소드입니다.
- ✓ 자바스크립트의 `innerHTML`이나 `textContent` 속성과 관련이 있습니다.
- ✓ `html()` 함수는 HTML 태그를 인식하지만, `text()` 함수는 HTML 태그를 인식하지 않습니다.
- ✓ 래퍼세트에서 `text()` 를 호출하면 태그를 인식하지 못하므로 태그를 제외한 텍스트만 반환합니다.

[사용법] 객체내부 조회 메소드

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

[jQuery] html()과 text() 메소드 사용

```
var html = $("#divArea").html(),  
    text = $("#divArea").text();  
  
$("#divArea").append(html).append(text);
```

[HTML] 태그와 텍스트가 섞여 있는 HTML 문서

03-html-get.html

```
<div id="divArea">  
  텍스트박스 : <input type="text" value="textBox"/><br/>  
  <span>spanTag</span><br/>  
</div>
```

[실행결과]

텍스트박스 :
spanTag
텍스트박스 :

4.3 DOM 내부 제어 (2/2) – HTML과 텍스트 설정

- ✓ `html(value)`와 `text(value)` 는 DOM 객체 내부를 설정하는 메소드입니다.
- ✓ `html(value)` 메소드는 HTML 태그를 인식하므로, DOM 객체 형태로 지정한 내용을 설정합니다.
- ✓ 그러나, `text(value)` 는 태그를 인식하지 못하므로, 텍스트 형태로 지정한 내용을 설정합니다.
- ✓ `text(value)` 로 태그가 포함된 내용을 설정하면, 이 내용은 단순 텍스트로 인식되어 태그 내용이 고스란히 보입니다.

[사용법] 객체내부 설정 메소드	[HTML] 태그와 텍스트가 섞여 있는 HTML 문서	04-html-set.html
<pre><code>\$(selector).html(value); \$(selector).text(value);</code></pre>	<pre><code><div class="htmlClass"></div> <div class="textClass"></div></code></pre>	
[실행결과]	jQuery] <code>html()</code> 과 <code>text()</code> 메소드 사용	
<p>첫번째</p> <p>두번째</p> <p><h2>첫 번째 </h2><h2>두 번째 </h2></p>	<pre><code>var html = "<h2>첫번째</h2>"; html += "<h2>두번째</h2>"; // 위의 html을 htmlClass div 태그에는 html 형태로, // textClass div 태그에는 text 형태로 삽입 \$("div.htmlClass").html(html); \$("div.textClass").text(html);</code></pre>	

4.4 DOM 추가 및 삭제 (1/3) – DOM 객체 추가

- ✓ `html()` 메소드는 문자열로 된 HTML을 DOM 객체로 추가합니다.
- ✓ 그러나, 문자열을 이어 붙여서 HTML을 구성하는 작업은 복잡하고 지저분한 코드를 유발합니다.
- ✓ `$()` 함수는 선택자를 수행하는 기능 뿐만 아니라, DOM 객체를 바로 생성하는 기능을 제공합니다.
- ✓ 메소드 체인을 사용하면, `$()` 함수로 DOM 객체를 생성하여 객체 특성을 설정하는 작업을 간편하게 할 수 있습니다.

[사용법] DOM 객체 생성 함수

```
$( );
```

[실행결과]

즐겨찾기

[넥스트리 흘블로그](#)

[jQuery] 메소드 체인을 통한 DOM 객체 생성과 특성 적용

05-create-dom.html

```
$( "<h2>즐겨찾기</h2>" )
  .css("color", "blue")
  .appendTo("body");
```

```
$( "<a/>" )
  .attr("href", "http://nextree.co.kr")
  .text("넥스트리 흘블로그")
  .appendTo("body");
```

4.4 DOM 추가 및 삭제 (2/3) – DOM 객체 삭제

- ✓ jQuery는 DOM 객체를 삭제하거나, DOM 객체 내부를 비우는 메소드를 제공합니다.
- ✓ remove() 메소드는 래퍼세트의 모든 요소를 HTML 문서에서 삭제하고, 삭제한 내용을 반환합니다.
- ✓ empty() 메소드는 래퍼세트의 모든 요소에 대해 하위 자식요소들을 삭제합니다.
- ✓ empty() 메소드는 하위 자식요소들을 삭제한 결과를 반환합니다.

[사용법] DOM 객체 삭제 메소드

```
$(selector).remove();  
$(selector).empty();
```

[HTML]

06-remove-empty.html

```
<div>  
  <h2>첫번째</h2>  
  <h2>두번째</h2>  
  <h2>세번째</h2>  
</div>  
  
<input type="button" id="btn1" value="두번째 H2 삭제"/>  
<input type="button" id="btn2" value="div 비움"/>
```

[jQuery] DOM 객체를 삭제하는 메소드 사용

```
$("#btn1").click(function () {  
  // h2 두번째 태그를 삭제  
  $("#seoncd").remove();  
});  
  
$("#btn2").click(function () {  
  // div 태그의 모든 하위 태그들을 삭제  
  $("div").empty();  
});
```

[실행결과]

첫번째

두번째

세번째

첫번째

세번째

4.4 DOM 추가 및 삭제 (3/3) – DOM 객체 복제

- ✓ `clone()` 은 기존에 존재하는 DOM 객체를 복제하여 사본을 생성하는 메소드입니다.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<script src="http://code.jquery.com/jquery-3.0.0.js">
</script>
<script>
$(document).ready(function () {
    $("body").append($(".h2").clone());
});
</script>
</head>

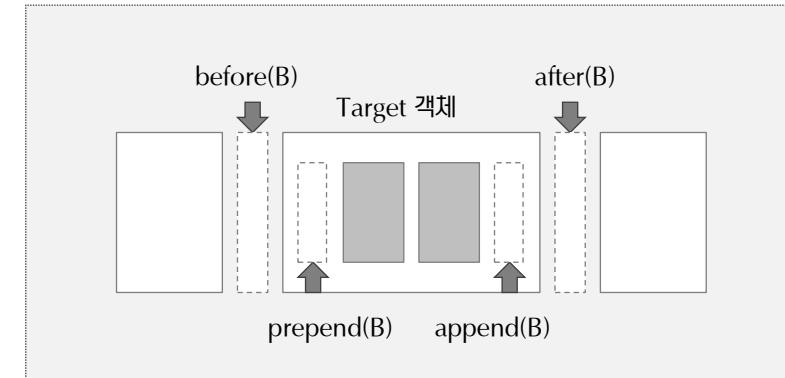
<body>
    <h2>첫번째</h2>
</body>
</html>
```

07-clone.html

4.5 DOM 객체 삽입 [1/3] – 개요

- ✓ DOM 객체 삽입은 이미 존재하는 DOM 객체에 다른 DOM 객체를 삽입하는 것입니다.
- ✓ Target 객체를 기준으로 어느 위치에 삽입하는지에 따라 다양한 메소드가 있습니다.
- ✓ `append()`, `prepend()` 메소드는 Target 객체의 자식요소로 DOM 객체를 삽입합니다.
- ✓ `before()`, `after()` 메소드는 Target 객체의 인접한 형제요소로 DOM 객체를 삽입합니다.

메소드	설명
<code>\$(Target).append(B)</code>	Target 객체의 하위 자식요소로 맨 뒤에 B 객체를 추가
<code>\$(Target).prepend(B)</code>	Target 객체의 하위 자식요소로 맨 앞에 B 객체를 추가
<code>\$(Target).after(B)</code>	Target 객체 바로 이전 요소로 B 객체를 추가
<code>\$(Target).before(B)</code>	Target 객체 바로 이후 요소로 B 객체를 추가



4.5 DOM 객체 삽입 [2/3] – jQuery메소드

- ✓ DOM 객체 삽입 메소드는 파라미터에 어떤 객체를 사용하는지에 따라 두 가지 유형으로 구분합니다.
- ✓ `appendTo()`, `prependTo()`, `insertAfter()`, `insertBefore()` 메소드는 삽입 대상 객체를 인자로 받습니다.
- ✓ `append()`, `prepend()`, `after()`, `before()` 메소드는 삽입하려는 내용(또는 객체)을 인자로 받습니다.
- ✓ 삽입하려는 내용이 DOM 객체인 경우, 해당 객체는 이전 위치에서 새로운 위치로 이동합니다.

실행결과

▶ 고객 목록

총 5명의 고객이 검색되었습니다.

이름 ▾ 검색

이름(이메일)	주소	전화번호	고객등급
하영화(yhha@namoosori.com)	전남 장성	010-2222-2826	우수고객
김민철(mckim@namoosori.com)	충남 아산	010-1234-8888	일반고객
이화진(hjlee@namoosori.com)	서울 관악구	017-134-6211	일반고객
최윤호(yhchoi@namoosori.com)	서울 강남구	016-111-2222	우수고객
강현석(hskang@namoosori.com)	충남 천안	019-400-9877	일반고객
나신규(nkna@namoosori.com)	서울 금천구…	010-9999-1234	우수고객

신규 고객을 테이블에 행으로 추가합니다. © namoo sori. All rights reserved.

[jQuery] 신규 고객정보(JSON)를 테이블 행으로 추가하기 *08-append.html*

```
// 신규 고객 목록에 추가하기
var customer = {
    "name" : "나신규",
    "email" : "new@namoosori.com",
    "address" : "서울 금천구",
    "phone" : "010-9999-1234",
    "vip" : true
};

var newTr = '<tr data-name="'+customer.name+'>';
newTr += '<td class="l"><input type="checkbox"></td>';
newTr += '<td class="l">' + customer.name + '(' + customer.email +')</td>';
newTr += '<td>' + customer.address + '</td>';
newTr += '<td>' + customer.phone + '</td>';
if (customer.vip) {
    newTr += '<td><strong>우수고객</strong></td>';
} else {
    newTr += '<td><strong>일반고객</strong></td>';
}
newTr += '</tr>';

$("tbody").append(newTr);
```

4.5 DOM 객체 삽입 [3/3] – 예제

- ✓ 다음은 4개의 이미지를 2초에 한번씩 순서를 바꾸는 예제입니다.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<script src="http://code.jquery.com/jquery-3.0.0.js">
</script>
<script>
$(function () {
    $("img").css("width", 200);

    setInterval(function () {
        $("img").first().appendTo("body");
    }, 2000);
});
</script>
</head>

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

09-image-move.html

4.6 jQuery안티 패턴 (1/6) – 개요

- ✓ jQuery는 다양한 방식으로 DOM을 탐색합니다. 동일한 결과를 보여 주는 코드를 서로 다르게 작성할 수 있습니다.
- ✓ 하지만, 이러한 코드들은 어떻게 구성하느냐에 따라 성능에 많은 차이가 있습니다.
- ✓ 특히, HTML 문서 크기가 클수록 성능은 큰 차이가 나타납니다.
- ✓ 성능에 좋지 않은 코딩 패턴을 살펴보고 성능을 향상시킬 수 있는 jQuery 사용 방법을 알아봅니다.



4.6 jQuery анти 패턴 [2/6] – Requery

- ✓ 이미 선택하였거나, 생성한 DOM 객체를 다시 조회하지 않도록 합니다.
- ✓ jQuery 메소드 체인을 사용하여 DOM 객체를 다시 조회하지 않고 처리합니다.



```
var anti = function() {
    $(document.body).append("<div class='baaron' />");
    // requery to bind stuff
    $("div.baaron:last").click(function() {
        // Something to do
    });
};
```

```
var preferred = function() {
    "<div class='baaron' />")
        .appendTo(document.body)
        .click(function() {
            // Something to do
        });
};
```

```
// test anti
var start = new Date();
for (var i = 0; i < 1000; i++) {
    anti();
}
console.log("anti => " + (new Date() - start));
```

```
// test preferred
start = new Date();
for (var i = 0; i < 1000; i++) {
    preferred();
}
console.log("preferred => " + (new Date() - start));
```

```
anti => 531
preferred => 122
```

4.6 jQuery 앤티 패턴 (3/6) – Append

- ✓ HTML을 추가할 때 전체 내용을 한꺼번에 만들어서 추가하는 것이 효율적입니다.



```
var anti = function (reallyLongArray) {
    $.each(reallyLongArray, function (count, item) {
        var newLI = '<li>' + item + '</li>';
        $('#ballers1').append(newLI);
    });
};
```

```
var preferred = function (reallyLongArray) {
    var frag = document.createDocumentFragment();
    $.each(reallyLongArray, function (count, item) {
        var newLI = $('<li>' + item + '</li>');
        frag.appendChild(newLI[0]);
    });
    $('#ballers2').append(frag);
};
```

```
var preferred2 = function (reallyLongArray) {
    var myhtml = '';
    $.each(reallyLongArray, function (count, item) {
        myhtml += '<li>' + item + '</li>';
    });
    $('#ballers3').html(myhtml);
};
```

```
// test data
var reallyLongArray = [];
for (var i = 0; i < 1000; i++) {
    reallyLongArray.push(i);
}

// test anti
var start = new Date();
anti(reallyLongArray);
console.log("anti => " + (new Date() - start));

// test preferred
start = new Date();
preferred(reallyLongArray);
console.log("preferred => " + (new Date() - start));

// test preferred2
start = new Date();
preferred2(reallyLongArray);
console.log("preferred2 => " + (new Date() - start));
```

anti => 68
preferred => 50
preferred2 => 3

4.6 jQuery 앤티 패턴 (4/6) – Selector cache

- ✓ Selector는 매번 조회하지 않고 필요할 때 한번만 조회하여 속도를 개선합니다.
- ✓ 그러나, 이 방법은 이후 DOM 객체가 변경 되었을 때 문제가 발생할 수 있으므로 주의하여 사용합니다.



```
// antipattern
$('#list-item1').click(function () {
    $('.photo1').hide();
});
```

```
// test anti
var listItem = $("#list-item1");
var start = new Date();
for (var i = 0; i < 1000; i++) {
    listItem.click();
}
console.log("anti => " + (new Date() - start));
```

```
// preferred
var photo;
$('#list-item2').click(function () {
    photo = photo || $('.photo2');
    photo.hide();
});
```

```
// test preferred
listItem = $("#list-item2");
start = new Date();
for (var i = 0; i < 1000; i++) {
    listItem.click();
}
console.log("preferred => " + (new Date() - start));
```

```
anti => 396
preferred => 341
```

4.6 jQuery안티 패턴 (5/6) – Always descend from an #id

- ✓ Selector를 사용할 때, 찾으려는 DOM 요소에 ID가 부여되어 있다면 ID 선택자를 우선 사용합니다.
- ✓ ID 선택자는 내부적으로 DOM API의 getElementById() 를 사용하므로 탐색 성능이 가장 좋습니다.
- ✓ HTML 문서가 클수록 ID 선택자를 사용한 탐색 성능은 큰 차이를 보입니다.



```
// antipattern
var anti = function(){
    var courses = $('.container div.course');
    // something to do
};
```

```
// better
var better = function(){
    var courses = $('#container div.course');
    // something to do
};
```

```
// preferred
var preferred = function(){
    var courses = $('#container').find('div.course');
    // something to do
};
```

```
// test anti
var start = new Date();
for (var i = 0; i < 10000; i++) {
    anti();
}
console.log("anti => " + (new Date() - start));
```

```
// test better
start = new Date();
for (var i = 0; i < 10000; i++) {
    better();
}
console.log("better => " + (new Date() - start));
```

```
// test preferred
start = new Date();
for (var i = 0; i < 10000; i++) {
    preferred();
}
console.log("preferred => " + (new Date() - start));
```

anti => 261
better => 249
preferred => 104

4.6 jQuery 앗티 패턴 [6/6] – Universal selector

- ✓ 전체 선택자(*)는 성능을 급격하게 떨어뜨릴 수 있습니다. 전체 선택자 사용은 자제합니다.



```
// antipattern 1
var anti1 = function(){
  var buttons = $('.buttons >');
  // something to do
};
```

```
// preferred 1
var preferred1 = function(){
  var buttons = $('.buttons').children();
  // something to do
};
```



```
// antipattern 2
var anti2 = function(){
  $('.gender :radio');
};

// antipattern 3
var anti3 = function(){
  $('.gender *:radio');
};
```

```
// preferred 2
var preferred2 = function(){
  // preferred 2
  $('.gender input:radio');
};
```

```
// test anti1
var start = new Date();
for (var i = 0; i < 1000; i++) {
  anti1();
}
console.log("anti => " + (new Date() - start));
// test preferred1
start = new Date();
for (var i = 0; i < 1000; i++) {
  preferred1();
}
```

```
console.log("preferred => " + (new Date() - start));
// test anti2
var start = new Date();
for (var i = 0; i < 1000; i++) {
  anti2();
}
console.log("anti2 => " + (new Date() - start));
// test anti3
var start = new Date();
for (var i = 0; i < 1000; i++) {
  anti3();
}
console.log("anti3 => " + (new Date() - start));
// test preferred2
start = new Date();
for (var i = 0; i < 1000; i++) {
  preferred2();
}
console.log("preferred2 => " + (new Date() - start));
```

anti => 45
preferred => 14
anti2 => 1500
anti3 => 1692
preferred2 => 125

4.7 jQueryEffect (1/4) – Effect 메소드

- ✓ jQueryEffect는 화면에서 보여주는 시각 효과를 구현하는 방법입니다.
- ✓ 사용자가 직접 애니메이션 효과를 만들 수 있습니다.
- ✓ jQuery플러그-인으로 시각 효과를 내는 메소드를 구현할 수 있습니다.
- ✓ jQuery에서 기본으로 제공하는 Effect 메소드는 아래와 같습니다.

메소드	설명
show()	DOM 요소를 화면에 보이도록 함
hide()	DOM 요소를 화면에서 숨김
toggle()	show()와 hide()를 번갈아 실행
slideDown()	DOM 요소를 슬라이드 효과와 함께 나타나게 함
slideUp()	DOM 요소를 슬라이드 효과와 함께 사라지게 함
slideToggle()	slideDown()과 slideUp() 을 번갈아 실행
fadeIn()	DOM 요소가 서서히 나타남
fadeOut()	DOM 요소가 서서히 사라짐
fadeToggle()	fadeIn()과 fadeOut() 을 번갈아 실행
animate()	css 속성 값을 이용하여 위치, 크기, 속성 값을 서서히 변경하는 애니메이션 효과를 줌

출처 : https://www.w3schools.com/jquery/jquery_ref_effects.asp

4.7 jQueryEffect (2/4) – Effect 매개변수

- ✓ jQueryEffect 모든 메소드는 공통적으로 세 개의 매개변수를 제공합니다.
- ✓ speed 는 효과를 진행하는 속도를 지정합니다.
- ✓ callback 은 효과를 완료한 후 실행할 함수를 지정합니다.
- ✓ easing은 애니메이션 easing 형태 지정합니다. 플러그-인을 사용하지 않으면 linear와 swing만 입력 가능합니다.

[사용법] Effect 메소드의 매개변수

```
$(selector).method();
$(selector).method(speed);
$(selector).method(speed, callback);
$(selector).method(speed, easing, callback);
```

[HTML]

10-hide-show_exmples.html

```
<button>토글 보여주기</button>
<div class="div1">
  <h2>토글 효과</h2>
  <p>토글 함수를 이용하여 간단한 jQuery효과를 볼 수 있습니다.</p>
</div>
```

[jQuery] button 을 클릭하면 div1 영역이 사라짐과 보임을 반복함

```
$("#button").click(function () {
  $(".div1").toggle("slow");
});
```

실행결과

토글 보여주기

토글 효과

토글 함수를 이용하여 간단한 jQuery 효과를 볼 수 있습니다.

4.7 jQueryEffect (3/4) – 사용자 정의 효과 (1/2)

- ✓ jQuery에서 기본으로 제공하는 간단한 효과만으로도 시각적인 효과를 얻을 수 있습니다.
- ✓ 조금 더 수준 높은 효과를 만들려면 효과를 개별적으로 정의해야 합니다.
- ✓ animate() 함수는 사용자 정의 효과를 만들 수 있는 방법을 제공합니다.
- ✓ 첫 번째 인자인 객체에 속성 값을 설정하여 세밀하게 효과를 조정할 수 있습니다.

[사용법] animate() 메소드

```
$(selector).animate(object);
$(selector).animate(object, speed);
$(selector).animate(object, speed, callback);
$(selector).animate(object, speed, easing, callback);
```

[CSS]

11-animate-example.html

```
<style>
  div {
    width: 50px; height: 50px;
    background-color: orange;
    position: relative;
  }
</style>
```

object 속성의 종류

opacity	top
height	bottom
width	margin
left	padding
right	

[jQuery] 마우스를 올리면 좌측 500px 이동하는 사각형

```
$("#div").hover(function() {
  $(this).animate({ left: 500 }, "slow");
}, function() {
  $(this).animate({ left: 0 }, "slow");
});
```

[HTML] 효과를 적용할 6개의 사각형

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

4.7 jQueryEffect (4/4) – 사용자 정의 효과 [2/2]

- ✓ 다음은 클릭하면 커지는 사각형 효과를 보여 줍니다.

```
<style>
  div {
    width: 50px; height: 50px;
    background-color: orange;
  }
</style>
</script>
<script>
$(function () {
  $("div").click(function () {
    $(this).animate({
      width: "+=50",
      height: "+=50"
    }, "slow");
  });
});
</script>
</head>

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

12-rectangle-example.html

4.8 요약

- ✓ 순수 자바스크립트만을 이용하여 DOM 객체 구조를 처리하는 것은 쉽지 않은 작업입니다.
- ✓ jQuery 메소드를 사용하면, DOM 객체를 다루는 작업을 간단하게 처리할 수 있습니다.
- ✓ jQuery 앤티패턴은 성능에 좋지 않는 코딩 패턴을 말합니다. 이러한 코딩을 지양하여 성능을 개선합니다.
- ✓ jQuery Effect를 사용하여 화면에서 발생하는 시각 효과를 구현할 수 있습니다.

DOM 특성제어	DOM 내부제어	DOM 추가 및 삭제	DOM 객체삽입
attr() removeAttr() addClass() removeClass() toggleClass() css()	html() text()	\$() remove() empty() clone()	append() appendTo() prepend() prependTo() after() insertAfter() before() insertBefore()



5. 이벤트 바인딩과 처리

-
- 5.1 JavaScript DOM 이벤트 처리
 - 5.2 jQueryDOM 이벤트 처리
 - 5.3 jQuery이벤트 바인딩
 - 5.4 다양한 이벤트 바인딩
 - 5.5 이벤트 발생과 처리
 - 5.6 마우스 이벤트
 - 5.7 키보드 이벤트
 - 5.8 윈도우 이벤트
 - 5.9 입력 이벤트
 - 5.10 요약

5.1 JavaScript DOM 이벤트 처리 (1/2)

- ✓ Event란 사용자가 키보드를 사용하거나 마우스 클릭과 같이 다른 것에 영향을 미치는 것을 의미합니다.
- ✓ JavaScript는 DOM에서 발생되는 이벤트에 반응하여 다양한 작업을 할 수 있습니다.
- ✓ 다음과 같은 상황에서 이벤트가 발생합니다.
 - 사용자가 마우스를 클릭 했을 때
 - 웹 페이지가 로딩 되었을 때
 - 이미지가 로딩 되었을 때
 - 마우스가 Element 사이로 움직일 때
 - Input 필드가 변경 되었을 때
 - Form이 submit 되었을 때
 - 키보드가 눌러 졌을 때

5.1 JavaScript DOM 이벤트 처리 (2/2)

- ✓ 다음은 JavaScript Event 예제입니다.

```
<html>
<head>
    <script>
        function changetext(id) {
            id.innerHTML = "Ooops!";
        }
    </script>
</head>
<body>
    <h1 onclick="changetext(this)">Click on this text!</h1>
</body>
</html>
```

```
<html>
<body>
    <input onkeydown="mOver(this)" onkeyup="mOut(this)" value="key down"/>

    <script>
        function mOver(obj) {
            obj.value="Thank You";
        }

        function mOut(obj) {
            obj.value="key down";
        }
    </script>
</body>
</html>
```

5.2 jQuery DOM 이벤트 처리

✓ jQueryEvent 처리

- jQueryEvent로 기존 JavaScript DOM Event를 간편하게 처리, 연결 가능
- 이벤트 핸들러를 할당, 해제할 수 있는 통합 메소드 제공
- DOM Element의 이벤트 타입마다 여러 핸들러 할당 가능
- click, mouseover 등과 같은 표준 이벤트 태입명을 사용
- 핸들러의 매개변수로 이벤트 인스턴스를 사용 가능
- 자주 사용하는 이벤트 인스턴스의 프로퍼티 등에 일관된 이름 제공
- 하나의 API로 표준 호환 브라우저와 IE 동시 지원

✓ jQueryEvent 기능

- click 함수로 클릭 이벤트 처리
- bind, on 함수를 이용한 모든 이벤트 처리
- unbind, off 함수를 이용하여 이벤트 제거

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

jQuery이벤트의 예

jQueryEvents

<http://api.jquery.com/category/events>

5.3 jQuery이벤트 바인딩 (1/3) – bind() 함수

✓ 선택된 DOM 객체의 이벤트에 지정한 핸들러를 연결하는 함수입니다.

✓ bind(eventType, data, listener)

- eventType : 핸들러를 할당할 이벤트 타입의 이름
- data : 핸들러 함수에서 사용할 데이터, 이벤트 인스턴스에 data라는 프로퍼티로 제공됨, 생략 시 두 번째 인자는 listener
- listener : 이벤트 발생시 수행되는 핸들러 함수

```
$(function(){
    $('#vstar').bind('click', function(event){say("What's up, first");})
        .bind('click', function(event){say("What's up, second");})
        .bind('click', function(event){say("What's up, third");});
});

function say(text){
    alert(text);
}
```

bind()

5.3 jQuery 이벤트 바인딩 [2/3] – unbind() 함수

✓ unbind() 함수는 객체 이벤트에서 지정한 핸들러를 지웁니다.

- unbind(eventType, handler)
- unbind(eventType)
 - 확장 집합의 모든 엘리먼트에서 매개변수에 따라 이벤트 핸들러를 삭제
 - 매개변수가 없을 경우 모든 핸들러를 제거
- 파라미터 종류
 - eventType : 핸들러를 할당할 이벤트 타입의 이름
 - handler : 이벤트 발생시 수행되는 핸들러 함수

5.3 jQuery 이벤트 바인딩 (3/3) – 예제

✓ 예제

```
<!DOCTYPE html>
<html>
<head>
<script src="http://code.jquery.com/jquery-3.0.0.js">
</script>
<script>
$(document).ready(function () {
    $("#bound").bind("click", function () {
        $("#show").bind("click", function () {
            $("p").show();
        });
        $("#hide").bind("click", function () {
            $("p").hide();
        });
    });

    $("#unbound").bind("click", function () {
        $("#show,#hide").unbind("click");
    });
});
</script>
</head>

<body>
    <h2>이 부분은 타이틀입니다.</h2>
    <p>이 부분은 단락입니다.</p>
    <p>이 부분은 또 다른 단락입니다.</p>
    <button id="show">show</button><br/>
    <button id="hide">hide</button><br/>
    <button id="bound">bound</button><br/>
    <button id="unbound">unbound</button><br/>
</body>
</html>
```

5.4 다양한 이벤트 바인딩 (1/8) – bind, delegate, live, on (1/3)

- ✓ bind(), delegate(), live(), on() 함수 모두 이벤트 핸들러를 DOM Element에 적용하는 함수입니다.
- ✓ 그런데 bind() 함수는 DOM 객체가 동적으로 생성된 경우 적용되지 않는 문제점이 있습니다.
- ✓ 이 문제를 해결하기 위해 live(), delegate() 등을 사용하기도 합니다.
- ✓ 한편, live()는 성능이나 이벤트 중복바인딩 등 문제가 많아서 1.4.x 버전 이후 delegate()로 개선되었습니다.
- ✓ 1.7.X 버전 이후에서는 bind(), live(), delegate() 를 통합한 on() 함수를 사용 가능합니다.
- ✓ 1.9 버전에서는 bind(), delegate(), live() 함수가 잠깐 사라졌다가 bind()와 delegate()는 다시 등장했습니다.
- ✓ bind(), delegate()도 결국 내부적으로 on()을 사용하므로 on()을 사용하기를 권장합니다.

5.4 다양한 이벤트 바인딩 (2/8) – bind, delegate, live, on [2/3]

- ✓ on()과 기존 방식의 차이점은 다음과 같습니다.

```
$( "h1" ).bind("click", function(){});
$( "#wrap" ).delegate("h1", "click", function(){});
$( "h1" ).live("click", function(){});
```

기존 이벤트 연결 방식

```
$( "h1" ).on("click", function(){});
$( "#wrap" ).on("click", "h1", function(){});
$( "h1" ).on("click", "h1", function());
```

on() 메소드를 이용한 연결 방식

이 부분은 div1입니다.

이 부분은 단락입니다.

이 부분은 또다른 단락입니다.

새로운 단락

새로운 단락

이 부분은 div2입니다.

새로운 단락

```
<script>
$(document).ready(function () {
    $("#append").click(function () {
        $("<p>새로운 단락</p>").appendTo("#div1");
    });
    $("#append2").click(function () {
        $("<p>새로운 단락</p>").appendTo("#div2");
    });
    $("#on").click(function () {
        // div1에 delegate 방식의 on 메소드로 click 핸들러 등록
        // 모든 <p> 태그에 대해 클릭 시 글자 색상을 빨강으로
    });
    $("#bind").click(function () {
        // div1에 bind 메소드로 click 핸들러 등록
        // 모든 <p> 태그에 대해 클릭 시 글자 색상을 빨강으로
    });
    $("#off").click(function () {
        // 모든 <p> 태그의 클릭 핸들러 해제
    });
});
</script>
<body>
    <div id="div1">
        <h2>이 부분은 div1입니다.</h2>
        <button id="on">on</button><br/>
        <button id="bind">bind</button><br/>
        <button id="off">off</button><br/>
        <button id="append">append to div1</button><br/>
        <button id="append2">append to div2</button><br/>
        <p>이 부분은 단락입니다.</p>
        <p>이 부분은 또다른 단락입니다.</p>
    </div>
    <hr/>
    <div id="div2">
        <h2>이 부분은 div2입니다.</h2>
    </div>
</body>
```

01-bindOn.html

5.4 다양한 이벤트 바인딩 (3/8) – bind, delegate, live, on [3/3]

✓ 예제

```
<script>
$(document).ready(function () {
    $("#append").click(function () {
        $("<p>새로운 단락</p>").appendTo("#div1");
    });
    $("#append2").click(function () {
        $("<p>새로운 단락</p>").appendTo("#div2");
    });

    $("#on").click(function(){
        // div1에 delegate 방식의 on 메소드로 클릭 핸들러 등록
        // 모든 <p> 태그에 대해 클릭 시 글자 색상을 빨강으로
        $("#div1").on("click", "p", function () {
            $(this).css("color", "red");
        });
    });
    $("#bind").click(function () {
        // div1에 bind 메소드로 클릭 핸들러 등록
        // 모든 <p> 태그에 대해 클릭 시 글자 색상을 빨강으로
        $("p").bind("click", function () {
            $(this).css("color", "red");
        });
    });
    $("#off").click(function () {
        // 모든 <p> 태그의 클릭 핸들러 해제
        $("#div1").off("click", "p");
    });
});
</script>
```

```
<body>
    <div id="div1">
        <h2>이 부분은 div1입니다.</h2>
        <button id="on">on</button><br/>
        <button id="bind">bind</button><br/>
        <button id="off">off</button><br/>
        <button id="append">append to div1</button><br/>
        <button id="append2">append to div2</button><br/>
        <p>0이 부분은 단락입니다.</p>
        <p>0이 부분은 또 다른 단락입니다.</p>
    </div>
    <hr/>
    <div id="div2">
        <h2>이 부분은 div2입니다.</h2>
    </div>
</body>
```

01-bindOn.html

5.4 다양한 이벤트 바인딩 (4/8) – on() 함수 [1/2]

- ✓ on() 함수는 bind() 함수와 마찬가지로 DOM 객체에 이벤트 핸들러를 연결합니다.
- ✓ delegate 방식으로 사용할 경우 현재 존재하는 DOM 객체 뿐만 아니라 미래에 존재할 DOM 객체에도 적용 가능합니다.
- ✓ 1.7.X 버전부터 생겨났으며 이벤트 연결에 가장 기본이 되는 함수로 권장하고 있습니다.

```
$(selector).on(eventType, function(event){});  
$selector.on(object);
```

on() 함수

```
<body>  
  <h2>첫번째</h2>  
  <h2>두번째</h2>  
  <h2>세번째</h2>  
</body>  
</html>
```

```
<style>  
  .reverse { background: black; color: white; }  
  
<script>  
$(document).ready(function () {  
    $("h2").on("click", function () {  
        $(this).html(function (index, html) {  
            return html + "*";  
        });  
    });  
  
    $("h2").on({  
        mouseenter: function () {  
            $(this).addClass("reverse"); },  
        mouseleave: function () {  
            $(this).removeClass("reverse"); }  
    });  
}</script>  
</head>
```

5.4 다양한 이벤트 바인딩 [5/8] – on() 함수 [2/2]

- ✓ 이벤트 바인딩에 있어서 문제는 동적으로 생성된 DOM 객체입니다.
- ✓ 즉, 기존 이벤트 연결 방식은 동적으로 생성된 DOM 객체에는 적용되지 않은 문제점이 있습니다.
- ✓ Event Delegate 방식은 부모 DOM 객체에 이벤트를 연결한 후 이를 하위 DOM 객체에 전달하는 방식으로 동적으로 적용 가능 합니다.

```
$(selector).on(eventType, delegate selector, function(event){});
```

on() 함수의 Event Delegate

```
<script>
$(document).ready(function () {
    $("#div1 > h2").on("click", function () {
        var size = $("#div1 > h2").size(),
            origin = $(this).html();
        $("<h2>" + size + " - " + origin + "</h2>").appendTo("#div1");
    });

    $("#div2").on("click", "h2", function () {
        var size = $("#div2 > h2").size(),
            origin = $(this).html();
        $("<h2>" + size + " - " + origin + "</h2>").appendTo("#div2");
    });
});
</script>
</head>
```

```
<body>
    <div id="div1">
        <h2>코스타 강좌</h2>
    </div>

    <div id="div2">
        <h2>코스타 과목</h2>
    </div>
</body>
</html>
```

5.4 다양한 이벤트 바인딩 (6/8) – off() 함수

- ✓ off() 함수는 on()과 반대로 DOM 객체의 이벤트를 제거합니다.
- ✓ 선택된 DOM 객체의 특정 이벤트 또는 모든 이벤트를 제거할 수 있습니다.

```
$(selector).off();  
$(selector).off(eventType);  
$(selector).off(eventType, function(){});
```

off() 함수

```
<body>  
  <h2>첫번째</h2>  
  <h2>두번째</h2>  
  <h2>세번째</h2>  
  <input type="button" id="btn1" value="클릭제거"/>  
</body>  
</html>
```

첫번째***

두번째*

세번째*

```
<style>  
  .reverse { background: black; color: white; }  
</style>  
<script>  
$(document).ready(function () {  
  $("h2").on({  
    click: function () {  
      $(this).html(function (index, html) {  
        return html + "*";  
      });  
    },  
    mouseenter: function () {  
      $(this).addClass("reverse"); },  
    mouseleave: function () {  
      $(this).removeClass("reverse"); }  
    });  
  $("#btn1").click(function () {  
    $("h2").off("click");  
  });  
</script>  
</head>
```

5.4 다양한 이벤트 바인딩 (7/8) – jQuery Simple Event Bind

- ✓ jQuery는 DOM 객체에 이벤트를 간단하게 연결할 수 있는 다양한 함수를 제공합니다.
- ✓ 예를 들어, 클릭 이벤트를 연결할 때 `on()`을 사용하는 대신, `click()` 함수를 사용할 수 있습니다.
- ✓ Simple Event 함수의 종류는 다음과 같습니다.

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

Simple Event 함수

```
$(“p”).on(“click”, function(event){});
```

on() 사용시



```
$(“p”).click(function(event){});
```

click() 사용시

5.4 다양한 이벤트 바인딩 (8/8) – one() 함수

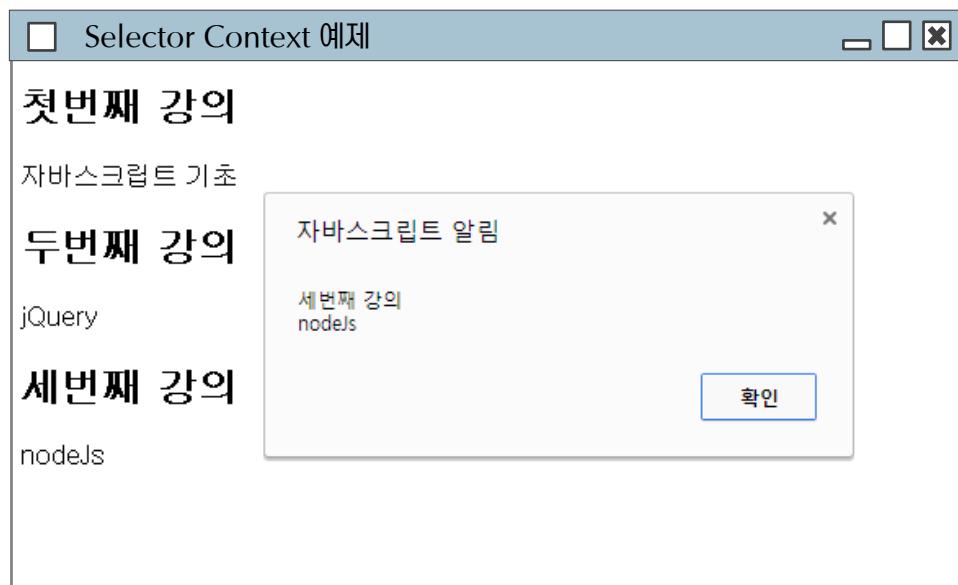
- ✓ one() 함수는 이벤트를 연결하고 한번 실행 후 삭제하는 기능입니다.
- ✓ on() -> 이벤트 실행 -> off() 의미와 동일합니다.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<script src="http://code.jquery.com/jquery-3.0.0.js">
</script>
<script>
$(document).ready(function () {
    $("h2").one("click", function () {
        $(this).html(function (index, html) {
            return html + "*";
        });
    });
});
</script>
</head>

<body>
<h2>첫번째</h2>
<h2>두번째</h2>
<h2>세번째</h2>
</body>
</html>
```

5.5 이벤트 발생과 처리 (1/7) – Selector Context

- ✓ Selector로 찾는 DOM 객체의 범위를 이벤트가 발생한 영역으로 한정하는 기법입니다.
- ✓ \$(selector) 대신 \$(selector, context)를 사용합니다.
- ✓ \$(context).find(selector)를 사용하여도 효과는 동일합니다.



```
<script>
  $(document).ready(function () {
    $("div").click(function () {
      var header = $("h2", this).text(),
          content = $("p", this).text();
      alert(header + "\n" + content);
    });
  });
</script>
</head>

<body>
  <div>
    <h2>첫번째 강의</h2>
    <p>자바스크립트 기초</p>
  </div>
  <div>
    <h2>두번째 강의</h2>
    <p>jQuery</p>
  </div>
  <div>
    <h2>세번째 강의</h2>
    <p>nodeJs</p>
  </div>
</body>
</html>
```

5.5 이벤트 발생과 처리 (2/7) – Event 객체 (1/2)

- ✓ Event 객체는 DOM 객체에서 이벤트가 발생시 jQuery에서 넘겨주는 여러가지 Event 정보를 담고 있습니다.
- ✓ 자바스크립트의 Event 객체와는 비슷하지만 조금 차이가 있습니다. (jQuery에서 브라우저 별 이벤트 정보를 정형화 함)

프로퍼티	설명
altKey, ctrlKey, shiftKey	각 키의 눌림여부 (true/false)
data	핸들러 할당시 bind() 커맨드의 두번째 인자로 전달된 값
keyCode	keyup, keydown 이벤트의 경우 눌린 키 반환
metaKey	pc에서 Ctrl, 맥에서는 Command 키 눌림여부(true/false)
pageX, pageY	마우스 이벤트시 페이지에서 이벤트가 발생한 x,y좌표
relatedTarget	이벤트가 발생했을 때 커서가 들어가거나 나온 엘리먼트
screenX, screenY	마우스 이벤트시 스크린에서 이벤트가 발생한 x,y좌표
target	이벤트가 발생한 엘리먼트
type	발생한 이벤트 타입
which	키보드 이벤트시 이벤트 발생시키기 키의 숫자코드, 마우스 이벤트시 눌려진 버튼값(좌 1, 중 2, 우 3)
preventDefault()	기본 이벤트를 제거함
stopPropagation()	이벤트 전달을 제거함

출처 : https://www.w3schools.com/jquery/jquery_ref_events.asp

5.5 이벤트 발생과 처리 (3/7) – Event 객체 (2/2)

✓ Event 객체를 응용한 canvas 예제



```
<script>
$(document).ready(function () {
    var canvas = document.getElementById("canvas"),
        context = canvas.getContext("2d");

    $(canvas).on({
        mousedown: function (event) {
            var pos = $(this).offset(),
                x = event.pageX - pos.left,
                y = event.pageY - pos.top;
            context.beginPath();
            context.moveTo(x, y);
        },
        mouseup: function (event) {
            var pos = $(this).offset(),
                x = event.pageX - pos.left,
                y = event.pageY - pos.top;
            context.lineTo(x, y);
            context.stroke();
        }
    });
</script>
</head>
<body>
<canvas id="canvas" width="700" height="400"
        style="border: 3px solid black">
</canvas>
</body>
</html>
```

02-canvas-example.html

5.5 이벤트 발생과 처리 (4/7) – 이벤트 강제 발생 (1/2)

- ✓ trigger()는 사용자의 입력 없이 이벤트를 강제로 발생하여 줍니다.
- ✓ 이벤트 핸들러를 직접 호출하지 않고 이벤트 발생을 통한 일관된 핸들러 호출을 유지하고 싶은 경우 사용됩니다.

```
$(selector).trigger(eventType);
$(selector).trigger(eventType, data);
```

trigger() 함수

첫번째****

두번째

세번째

```
<script>
$(document).ready(function () {
    // TODO: 모든 <h2> 태그에 click 핸들러 등록
    // 이벤트가 발생되면 해당 태그 뒤에 별표(*)를 붙임

    setInterval(function () {
        // TODO : 첫 번째 h2 태그의 click 이벤트 강제 발생
    }, 1000);
});
</script>
</head>

<body>
    <h2>첫번째</h2>
    <h2>두번째</h2>
    <h2>세번째</h2>
</body>

</html>
```

04-trigger.html

5.5 이벤트 발생과 처리 (5/7) – 이벤트 강제 발생 (2/2)

✓ 예제

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<script src="http://code.jquery.com/jquery-3.0.0.js"></script>
<script>
$(document).ready(function () {
    // TODO: 모든 <h2> 태그에 click 핸들러 등록
    // click 이벤트가 발생되면 해당 태그 뒤에 별표(*)를 붙임
    $("h2").on("click", function () {
        $(this).html(function (index, html) {
            return html + "*";
        });
    });

    setInterval(function () {
        // TODO : 첫번째 h2 태그의 click 이벤트 강제 발생
        $("h2").first().trigger("click");
    }, 1000);
});
</script>
</head>
<body>
    <h2>첫번째</h2>
    <h2>두번째</h2>
    <h2>세번째</h2>
</body>
</html>
```

04-trigger.html

5.5 이벤트 발생과 처리 (6/7) – 기본 이벤트 전달

- ✓ <a> 링크와 같이 이벤트를 정의하지 않아도 기본으로 동작하는 이벤트를 기본 이벤트라 합니다.
- ✓ 만일 부모 요소에 같은 타입의 이벤트 핸들러가 정의되어 있다면 이를 같이 실행하는데 이를 이벤트 전달이라 합니다.
- ✓ 가끔 이러한 기본 이벤트 및 이벤트 전달을 막아야 될 필요성이 생깁니다.
- ✓ jQuery에서는 이러한 기본 이벤트 및 이벤트 전달을 막을 수 있는 방법을 제공합니다.

```
event.preventDefault();
```

기본 이벤트 제거

```
event.stopPropagation();
```

이벤트 전달 제거

KOSTA

자바스크립트 알림

a click!

확인

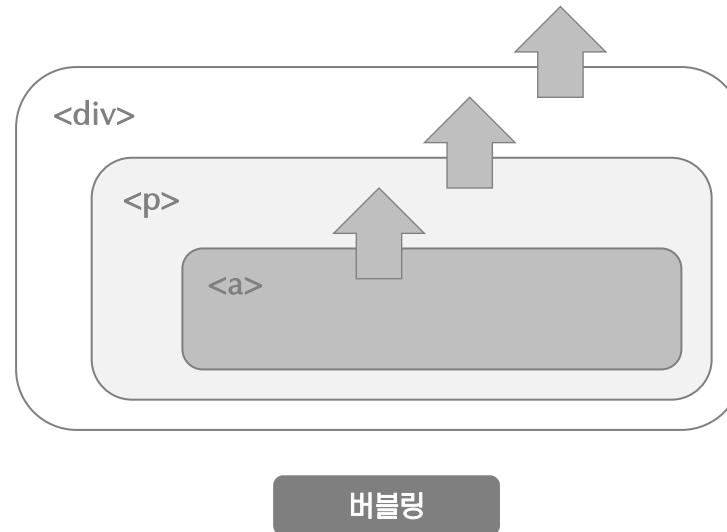
```
<script>
    $(document).ready(function () {
        $("a").click(function (event) {
            alert("a click!");
            // a 링크 고유기능이 사라짐
            event.preventDefault();
            // h2 이벤트 전달이 사라짐
            event.stopPropagation();
        });
        $("h2").click(function () {
            alert("h2 click!");
        });
    });
</script>
</head>

<body>
    <h2><a href="http://edu.kosta.or.kr">KOSTA</a></h2>
</body>
</html>
```

05-bubble.html

5.5 이벤트 발생과 처리 (7/7) – 이벤트 전파방식

- ✓ 이벤트 버블링은 자식요소에 발생한 이벤트가 부모요소까지 전파된다는 의미입니다.
- ✓ 이벤트가 발생하면 이벤트 요소부터 이벤트 요소를 포함하고 있는 부모 요소까지 올라가며 이벤트를 검사합니다.
- ✓ 이때 버블속성의 이벤트 핸들러가 있다면 실행하면서 올라갑니다.



5.6 마우스 이벤트 (1/3)

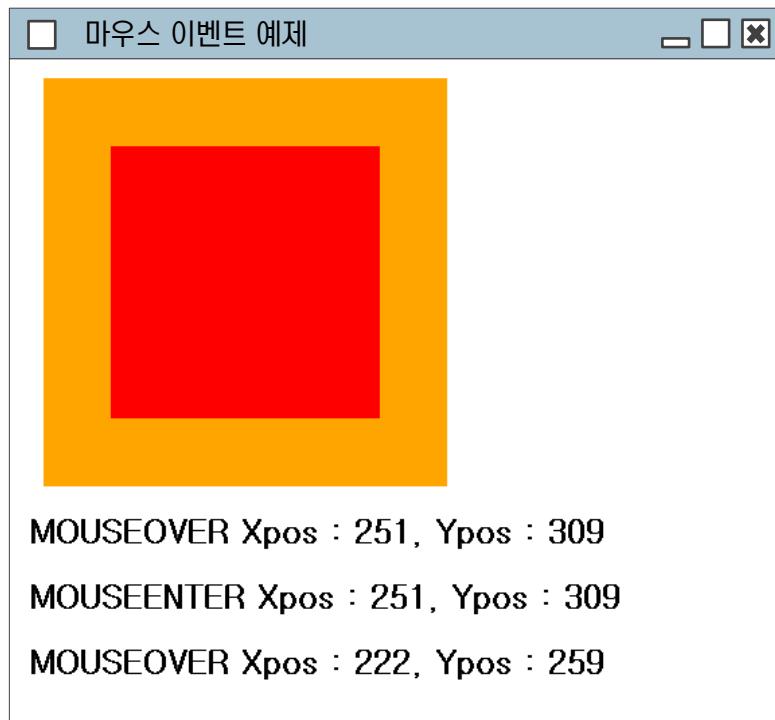
- ✓ jQuery에서는 마우스 클릭 및 움직임과 관련된 이벤트 처리가 가능합니다.
- ✓ mouseenter 이벤트는 경계 외부에서 내부로 접근시 발생합니다.
- ✓ mouseover 이벤트는 마우스가 요소 안에 들어 올 때 발생하며 이벤트 버블링이 적용됩니다.

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

출처 : https://www.w3schools.com/jquery/jquery_ref_events.asp

5.6 마우스 이벤트 (2/3)

- ✓ 다음은 마우스이벤트 예제입니다.



```
<style>
    .outer { width: 200px; height: 200px; background: orange;
              padding: 50px; margin: 10px; }
    .inner { width: 100%; height: 100%; background: red; }

<script>
$(document).ready(function () {
    // TODO: 바깥 div 태그에 mouseover와 mouseenter 핸들러를 등록 한다.
    // 핸들러에서는 이벤트가 발생한 마우스의 x,y 좌표를 출력한다.

    // 이벤트 전파를 막을 경우
    /*
    $(".inner").mouseover(function(event) {
        event.stopPropagation();
    });
    */
});

</script>
</head>

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

06-mouse.html

5.6 마우스 이벤트 (3/3)

✓ 예제

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style>
    .outer { width: 200px; height: 200px; background: orange; padding: 50px; margin: 10px; }
    .inner { width: 100%; height: 100%; background: red; }
</style>
<script src="http://code.jquery.com/jquery-1.11.1.js"></script>
<script>
$(document).ready(function () {
    // TODO: 바깥 div 태그에 mouseover와 mouseenter 핸들러를 등록 한다.
    // 핸들러 에서는 이벤트가 발생한 마우스의 x,y 좌표를 출력한다.
    $(".outer").mouseover(function (event) {
        $("body").append("<h2>MOUSEOVER Xpos : " + event.pageX + ", Ypos : " + event.pageY + "</h2>");
    }).mouseenter(function (event) {
        $("body").append("<h2>MOUSEENTER Xpos : " + event.pageX + ", Ypos : " + event.pageY + "</h2>");
    });
    // 이벤트 전파를 막을 경우
    /*
    $(".inner").mouseover(function(event) {
        event.stopPropagation();
    });
    */
});
</script>
</head>

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

06-mouse.html

5.7 키보드 이벤트 (1/2)

- ✓ 키보드에서 입력할 때, 키보드 이벤트 처리 가능합니다.

이벤트	설명
keydown	키보드를 누를 때 발생
keypress	글자가 입력될 때 발생
keyup	키보드를 뗄 때 발생

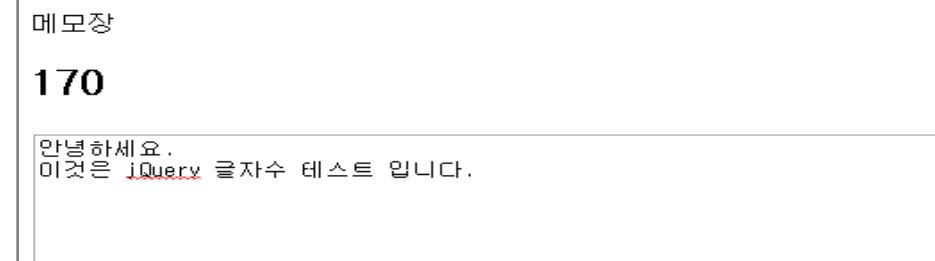


```
<script>
$(document).ready(function () {
    // TODO: textArea 태그에 키가 올라왔을 때 핸들러 등록
    // 핸들러 내용은 textArea 안의 글자수를 계산하여 출력
});

</script>
</head>
```

```
<body>
<div>
    <p>메모장</p>
    <h2>200</h2>
    <textarea rows="6" cols="70"></textarea>
</div>
</body>
</html>
```

07-keyboard.html



5.7 키보드 이벤트 [2/2]

✓ 예제

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<script src="http://code.jquery.com/jquery-1.11.1.js"></script>
<script>
$(document).ready(function () {
    // TODO: textArea 태그에 키가 올라왔을 때 핸들러 등록
    // 핸들러 내용은 textArea 안의 글자수를 계산하여 출력.
    $("textarea").keyup(function () {
        var inputSize = $(this).val().length,
            remain = 200 - inputSize;
        $("h2").html(remain);
    });
});
</script>
</head>

<body>
<div>
    <p>메모장</p>
    <h2>200</h2>
    <textarea rows="6" cols="70"></textarea>
</div>
</body>
</html>
```

07-keyboard.html

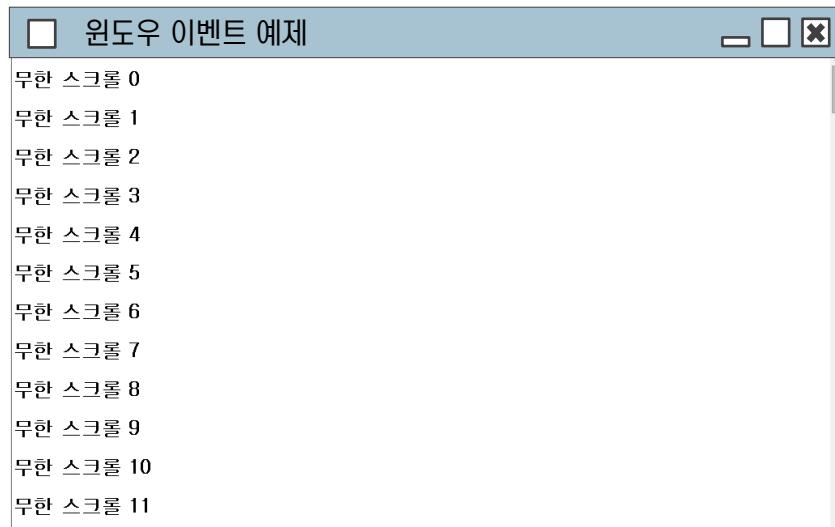
5.8 원도우 이벤트 (1/2)

- ✓ 원도우에서 발생한 변화를 이벤트로 처리 가능합니다.

이벤트	설명
ready	DOM 객체가 준비되면 발생
load	원도우를 불러들일 때 발생
unload	원도우를 닫을 때 발생
resize	원도우 크기가 변화될 때 발생
scroll	원도우 스크롤을 할 때 발생
error	에러가 있을 때 발생

5.8 원도우 이벤트 (2/2)

- ✓ 원도우 이벤트를 이용하여 무한 스크롤을 구현합니다.



```
<script>
$(document).ready(function () {
    var count = 0;

    $(window).scroll(function () {
        var scrollHeight = $(window).scrollTop() + $(window).height(),
            documentHeight = $(document).height();

        if (scrollHeight === documentHeight) {
            for (var i = 0; i < 10; i++) {
                $("<h2>무한 스크롤 " + count++ + "</h2>").appendTo("body");
            }
        }
    });
    for (var i = 0; i < 20; i++) {
        $("<h2>무한 스크롤 " + count++ + "</h2>").appendTo("body");
    }
});
</script>
</head>

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

08-scroll-example.html

5.9 입력 이벤트 (1/6)

- ✓ 일반적으로 HTML에서 입력 양식으로 데이터를 전송하는 작업을 많이 합니다.
- ✓ 이러한 입력 양식에서도 발생하는 이벤트가 존재하며 jQuery에서는 이를 지원합니다.
- ✓ 주로 <form>, <input> 등에서 발생하는 입력 이벤트를 처리합니다.

이벤트	설명
change	입력 양식의 내용을 변경할 때 발생
focus	입력 양식에 포커스가 가면 발생
focusin	입력 양식에 포커스가 가기 전에 발생
focusout	입력 양식에 포커스가 사라지기 전에 발생
blur	입력 양식에 포커스가 사라지면 발생
select	입력 양식을 선택할 때 발생(text, textarea 제외)
submit	submit 버튼 선택시 발생
reset	reset 버튼 선택시 발생

5.9 입력 이벤트 (2/6)

✓ submit 이벤트

- form submit 버튼 클릭시 수행되는 이벤트입니다.
- 아래 예제는 submit 버튼 클릭시 이름과 패스워드를 출력하는 예제입니다.

Submit 이벤트 예제

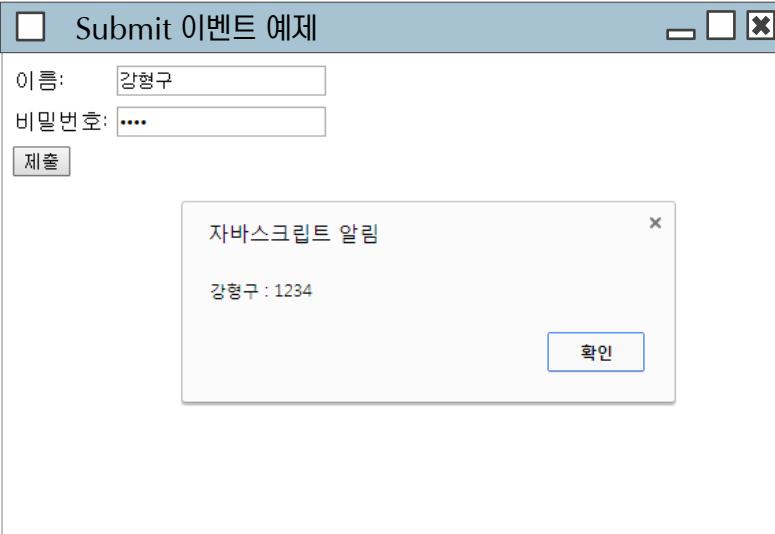
이름: 강혁구
비밀번호:

제출

자바스크립트 알림

강혁구 : 1234

확인

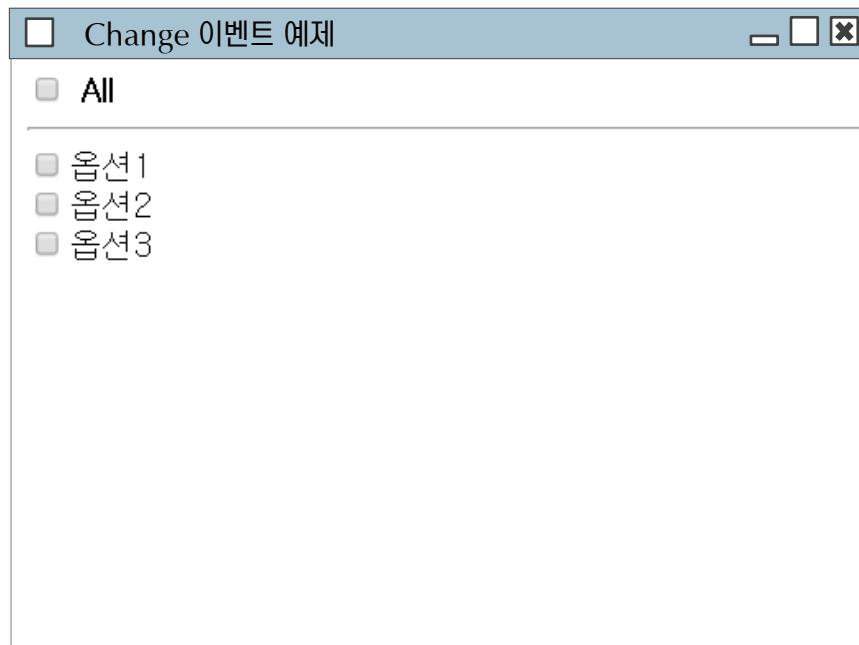


```
<script>
$(document).ready(function () {
    $("#myForm").submit(function (event) {
        var name = $("#name").val(), password = $("#password").val();
        alert(name + " : " + password);
        event.preventDefault();
    });
});
</script>
<body>
<form id="myForm">
    <table>
        <tr><td>이름:</td>
            <td><input type="text" name="name" id="name"/></td>
        </tr>
        <tr><td>비밀번호:</td>
            <td><input type="password" name="password" id="password"/></td>
        </tr>
    </table>
    <input type="submit" value="제출"/>
</form>
</body>
```

5.9 입력 이벤트 (3/6)

✓ change 이벤트 예제 1

- 아래 화면에서 All 체크박스를 체크/해제 하면 아래의 세 옵션 모두 동일하게 체크/해제 되도록 구현하세요.



```
<script>
$(document).ready(function () {
    // TODO : All 체크박스를 체크/해제 하면
    // 다른 모든 체크박스도 체크/해제

});
</script>
</head>

<body>
    <input type="checkbox" id="allCheck"/>
    <label><b>All</b></label>
    <hr/>
    <div id="checkItem">
        <input type="checkbox"/><label>옵션1</label><br/>
        <input type="checkbox"/><label>옵션2</label><br/>
        <input type="checkbox"/><label>옵션3</label><br/>
    </div>
</body>
</html>
```

09-change-checkbox.html

5.9 입력 이벤트 (4/6)

✓ change 이벤트 예제 1 구현

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<script src="http://code.jquery.com/jquery-1.11.1.js"></script>
<script>
$(document).ready(function () {
    // TODO : All 체크박스를 체크/해제 하면
    // 다른 모든 체크박스도 체크/해제
    $("#allCheck").change(function () {
        if ($(this).is(":checked")) {
            $("#checkItem").children().prop("checked", true);
        } else {
            $("#checkItem").children().prop("checked", false);
        }
    });
});
</script>
</head>
<body>
    <input type="checkbox" id="allCheck"/>
    <label><b>All</b></label>
    <hr/>
    <div id="checkItem">
        <input type="checkbox"/><label>옵션1</label><br/>
        <input type="checkbox"/><label>옵션2</label><br/>
        <input type="checkbox"/><label>옵션3</label><br/>
    </div>
</body>
</html>
```

09-change-checkbox.html

5.9 입력 이벤트 (5/6)

✓ change 이벤트 예제 2

- 아래 화면과 같이 입력한 내용을 아래 div 영역 안에 복사되도록 구현하세요.



```
<style>
div {
    width: 500px; height: 500px;
    background-color: orange;
}
</style>
<script>
$(document).ready(function () {
    // TODO: textArea의 텍스트가 변경되면 내용을 divArea에 복사
});
</script>
</head>
<body>
    <h5>입력 : </h5>
    <textarea id="textArea" rows="5" cols="68"></textarea>
    <hr/>
    <div id="divArea">
    </div>
</body>
</html>
```

10-change-textarea.html

5.9 입력 이벤트 (6/6)

✓ change 이벤트 예제 2 구현

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<style>
div { width: 500px; height: 500px; background-color: orange; }
</style>
<script src="http://code.jquery.com/jquery-1.11.1.js"></script>
<script>
$(document).ready(function () {
    // TODO: textArea의 텍스트가 변경되면 내용을 divArea에 복사
    $("#textArea").change(function () {
        $("#divArea").text($(this).val());
    });
});
</script>
</head>

<body>
<h5>입력 : </h5>
<textarea id="textArea" rows="5" cols="68"></textarea>
<hr/>
<div id="divArea"></div>
</body>
</html>
```

10-change-textarea.html

5.10 요약

- ✓ JavaScript는 DOM에서 발생하는 다양한 이벤트에 반응하여 원하는 작업을 수행할 수 있습니다.
- ✓ jQuery는 기존 JavaScript DOM Event를 간편하게 처리하고 연결할 수 있는 메소드를 제공합니다.
- ✓ bind, delegate, live, on과 같은 함수는 이벤트 핸들러를 DOM 요소에 적용하는 jQuery 함수입니다.
- ✓ 계층구조를 가진 DOM 객체에서 정확한 이벤트 처리를 하려면, 기본이벤트와 이벤트 전파방식 이해가 필요합니다.

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

Simple Event 함수

```
$(“p”).on(“click”, function(event){});
```

on() 사용시

```
$(“p”).click(function(event){});
```

click() 사용시

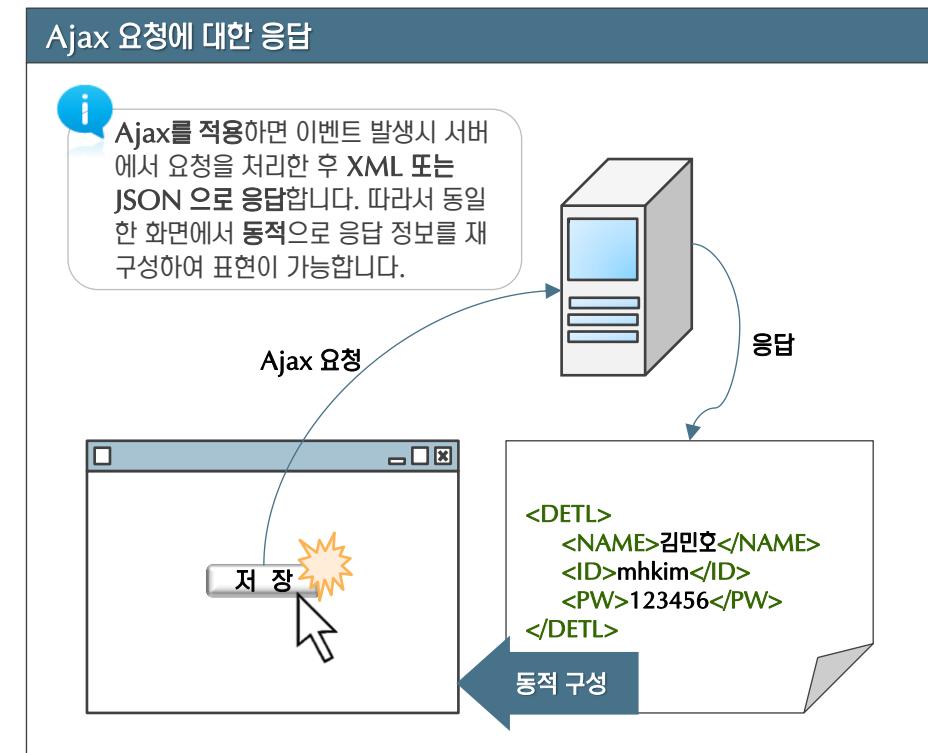
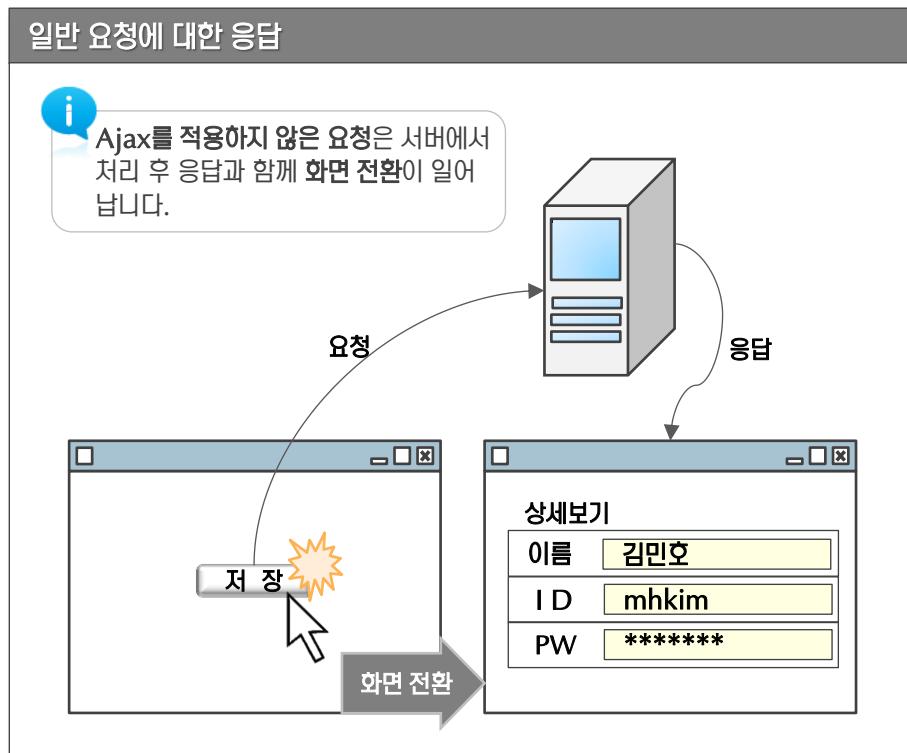


6. Ajax 프로그래밍

- 6.1 Ajax 소개
- 6.2 Ajax 프로그래밍
- 6.3 jQuery Ajax 함수
- 6.4 Ajax 전역 함수
- 6.5 요약

6.1 Ajax 소개 (1/3) – 개요

- ✓ Ajax (Asynchronous Javascript and XML)는 언어나 프레임워크가 아닌 구현하는 방식을 의미합니다.
- ✓ Ajax는 웹(Web)에서 화면을 갱신하지 않고 데이터를 서버로부터 가져와 처리하는 방법을 의미합니다.
- ✓ JavaScript의 XMLHttpRequest(XHR) 객체로 데이터를 전달하고 비동기 방식으로 결과를 조회합니다.
- ✓ 화면 갱신이 없으므로 사용자 입장에서는 편리하지만, 동적으로 DOM을 구성해야하므로 구현이 복잡합니다.



6.1 Ajax 소개 (2/3) – XMLHttpRequest 객체

- ✓ XMLHttpRequest 는 자바스크립트가 Ajax 방식으로 통신할 때 사용하는 객체입니다.
- ✓ XMLHttpRequest 객체는 Ajax 통신 시 전송방식, 경로 등 전송정보를 싣는 역할을 합니다.
- ✓ 실제 서버와의 통신은 브라우저의 Ajax 엔진에서 수행합니다.
- ✓ 직접 자바스크립트로 Ajax를 프로그래밍할 경우 브라우저 별로 통신방식이 달라 코드가 복잡해집니다.

```
<script>
  function getXMLHttpRequest() {
    if (window.ActiveXObject) // 마이크로 소프트 IE
    {
      try {
        return new ActiveXObject("Msxml2.XMLHTTP"); // IE 업 버전
      } catch (e) {
        try {
          return new ActiveXObject("Microsoft.XMLHTTP"); // IE 다운 버전
        } catch (e1) {
          return null;
        }
      }
    } else if (window.XMLHttpRequest) // 기타 웹 브라우저들
    {
      return new XMLHttpRequest();
    } else {
      return null;
    }
  }
  var httpRequest = null;
```

```
function sendRequest(url, params, callback, method) {
  httpRequest = getXMLHttpRequest();

  var httpMethod = method ? method : 'GET';
  if (httpMethod != 'GET' && httpMethod != 'POST') {
    httpMethod = 'GET';
  }

  var httpUrl = url;
  var httpParams = (params == null || params == '') ? null : params;
  if (httpMethod == 'GET' && httpParams != null) {
    httpUrl = httpUrl + "?" + httpParams;
  }

  // open(): 요청의 초기화, GET/POST 선택, 접속할 URL 입력
  httpRequest.open(httpMethod, httpUrl, true);
  httpRequest.setRequestHeader('Content-Type',
    'application/x-www-form-urlencoded');
  httpRequest.onreadystatechange = callback; // 콜백함수 등록

  // send(): 웹 서버에 요청 전송
  httpRequest.send(httpMethod == 'POST' ? httpParams : null);
}

function loadGroups() {
  sendRequest("/groups", null, loadedGroups, "GET");
}

function loadedGroups() {
  if (httpRequest.readyState == 4) {
    if (httpRequest.status == 200) {
      var groupList = document.getElementById("groupList");
      groupList.innerHTML = httpRequest.responseText;
    }
  }
}
```



6.1 Ajax 소개 (3/3) – jQueryAjax

- ✓ jQuery를 사용하면 자바스크립트 보다 간편하게 Ajax를 구현할 수 있습니다.
- ✓ XMLHttpRequest 객체를 직접 다루지 않고 jQuery API 만으로 Ajax를 처리합니다.
- ✓ 웹 브라우저 특성에 따른 차이를 고려하지 않아도 되므로 코드가 간결해집니다.
- ✓ jQueryAjax는 ajax(), get(), post(), load() 등 다양한 Ajax 함수를 제공합니다.

```
1 <script>
2 $(document).ready(function () {
3     $.ajax("/groups", {
4         success: function (data) {
5             console.log(data);
6         }
7     });
8 });
9 </script>
10 </head>
11
12 <body>
13     <div id="div1"></div>
14 </body>
15 </html>
```

코드설명

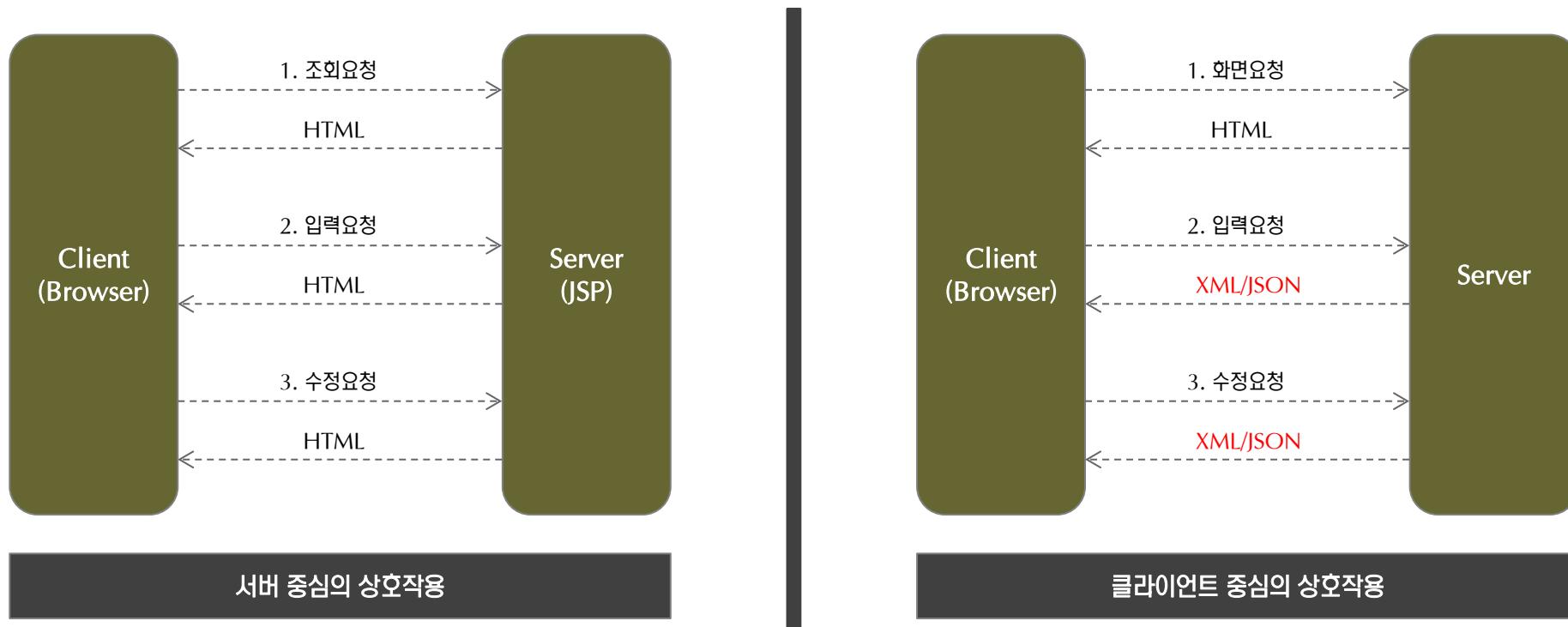
[Line3] \$.ajax() 함수를 통해 서버의 /groups URL로 요청을 보냅니다.
[Line5] 서버 응답결과가 성공인 경우, 서버에서 전달된 객체를 콘솔에 로그로 남깁니다.

실행결과

```
▼ Array[2]
  ▼ 0: Object
    code: "A0001"
    id: null
    ▶ members: Array[0]
    name: "넥스트리소프트"
    registDate: 1391481796233
    ▶ __proto__: Object
  ▼ 1: Object
    code: "A0002"
    id: null
    ▶ members: Array[0]
    name: "KOSTA"
    registDate: 1391481796233
    ▶ __proto__: Object
    length: 2
    ▶ __proto__: Array[0]
```

6.2 Ajax 프로그래밍 (1/2) – 서버/클라이언트 상호작용

- ✓ 웹 화면을 구성하는 방식은 서버 중심의 상호작용 방식과 클라이언트 중심의 상호작용 방식으로 구분합니다.
- ✓ 서버 중심의 개발방식은 화면 구성이 서버에서 이루어 집니다. (프리젠테이션 영역의 JSP나 PHP 등)
- ✓ 클라이언트 중심의 개발방식은 클라이언트(웹 브라우저)에서 화면을 구성합니다. (주로 JavaScript)
- ✓ Ajax는 클라이언트 중심의 개발방식이며 비동기 요청보다는 동적 화면구성이 관건입니다.



6.2 Ajax 프로그래밍 [2/2] – 간단한 입력/조회

✓ 서버로 입력요청 보내기

- Web에서의 입력작업은 서버에 어떤 내용을 저장한다는 의미입니다.
- 서버에 저장 기능이 존재해야 합니다. (저장 기능이 가능한 URL)
- 입력 요청을 보낼 때 주로 HTTP METHOD 중 POST 방식을 사용합니다.

✓ 입력 및 조회 예제 구현

- 간단한 입력 및 조회로 서버와 연동하는 화면을 구현합니다.
- 서버에 저장 요청을 하면 서버는 저장 후 저장된 결과를 넘겨줍니다.
- 화면에서는 저장된 결과 데이터로 목록 화면을 동적으로 구성합니다.

```
<body>
  <h2>등록 회사 리스트</h2>
  <p>회사을 등록하고 조회합니다.</p>
  <table>
    <tr>
      <th>ID</th>
      <th>회사코드</th>
      <th>회사명</th>
    </tr>
  </table>
  <p>
    회사코드: <input type="text" id="code"/>
    회사명: <input type="text" id="name"/>
    <button>입력</button>
  </p>
</body>
```

```
<script>
$(document).ready(function () {
  $.ajax({
    url: "/groups",
    type: "get",
    dataType: "json",
    success: function (data) {
      for (var i = 0; i < data.length; i++) {
        appendTr(data[i]);
      }
    }
  });

  $("button").click(function () {
    $.ajax({
      url: "/groups/create",
      type: "post",
      dataType: "json",
      data: { code: $("#code").val(),
              name: $("#name").val() },
      complete: clear(),
      success: function (data) {
        appendTr(data);
      }
    });
  });
});

function appendTr(data) {
  var tr = $(<tr/>).appendTo("table");
  $(<td/>).text(data.id).appendTo(tr);
  $(<td/>).text(data.code).appendTo(tr);
  $(<td/>).text(data.name).appendTo(tr);
}

function clear() {
  $("input:text").val("");
}
</script>
```

6.3 jQuery Ajax 함수 (1/5) – \$.ajax() 함수 [1/2]

- ✓ \$.ajax() 함수는 jQuery에서 Ajax 기능을 제공하는 가장 기본적인 함수입니다.
- ✓ 다른 함수들에 비하여 옵션을 다양하게 지정할 수 있으며 실무에서 가장 많이 사용합니다.
- ✓ 함수의 옵션은 다양하지만 대부분 자동으로 지정하므로 생략 가능합니다.

Ajax 함수

`$.ajax(options);`
`$.ajax(url, options);`

옵션	설명
async	true : 비동기 호출, false : 동기 호출
url	호출 대상 URL 지정
data	요청 매개변수 지정(Object)
type	http method 지정. (GET 또는 POST)
contentType	요청에 지시되는 contentType 기본값은 "application/x-www-form-urlencoded"
dataType	응답의 결과로 반환되는 데이터의 종류 <ul style="list-style-type: none">xml : 응답텍스트는 xml로 파싱, XML DOM을 전달html : callback함수로 텍스트 그대로 전송, html코드평가json : JSON 문자열로 평가, 생성된 객체 전달jsonp : json 타입과 유사, 원격스크립트 허용script : 콜백 호출보다 우선하여 스크립트 구문으로 처리text : 일반 텍스트로 전달

출처 : https://www.w3schools.com/jquery/ajax_ajax.asp

6.3 jQuery Ajax 함수 [2/5] – \$.ajax() 함수 [2/2]

- ✓ .ajax() 함수 option은 다음과 같습니다.

옵션	설명
timeout	Ajax 요청의 제한시간, 제한시간 초과시 error callback호출되거나 요청이 취소
success(data, textStatus, jqXHR)	응답이 성공시 호출되는 함수
error(jqXHR, textStatus, errorThrown)	응답이 에러 상태코드 반환하면 호출되는 함수
complete(jqXHR, textStatus)	요청이 완료되면 호출되는 함수 (success or error 이후 실행)
global	true : 전역함수 활성, false : 전역함수 비활성
beforeSend()	요청 전송이전 호출 함수
processData	false 시 URL 인코딩 형태로 처리되어 전달되는 데이터를 금지
ifModified	true 시 헤더 정보 확인 하여 미 변경시 요청 성공
jsonp	JSONP 매개변수 이름 지정
jsonpCallback	JSONP 콜백함수 이름 지정

출처 : https://www.w3schools.com/jquery/ajax_ajax.asp

6.3 jQuery Ajax 함수 [3/5] – \$.get(), \$.post() 함수 [1/2]

- ✓ \$.get(), \$.post() 함수는 \$.ajax() 의 옵션 속성 중 type 옵션이 미리 지정된 함수입니다.
- ✓ 지정된 HTTP Method로 Ajax 통신을 하며 get()은 GET 방식을, post()는 POST 방식을 사용합니다.

get 함수

```
$.get(url, function (result, textStatus, jqXHR){});
$.get(url, data, function (result, textStatus, jqXHR){});
```

get 함수를 이용한 Ajax 구현

```
<script>
$(document).ready(function () {
    $.get("/groups", function (data) {
        for (var i = 0; i < data.length; i++) {
            $("<h2>" + data[i].name + "(" + data[i].code + ")" + "</h2>")
                .appendTo("#div1");
        }
    });
});
</script>
</head>

<body>
    <div id="div1"></div>
</body>
</html>
```

6.3 jQuery Ajax 함수 (4/5) – \$.get(), \$.post() 함수 [2/2]

- ✓ \$.post() 함수는 POST 방식으로 Ajax 통신을 합니다.

post 함수

```
$.post(url, function (data, textStatus, jqXHR){});
$.post(url, data, function (data, textStatus, jqXHR){});
```

post 함수를 이용한 Ajax 구현

```
<script>
$(document).ready(function () {
    $.post("/groups/create", {
        code: "X0001",
        name: "우리회사"
    }, function (data){
        console.log(data);
    });
});
</script>
</head>

<body>
    <div id="div1"></div>
</body>
</html>
```

GET, POST 방식

HTTP 프로토콜의 요청 Method입니다.
GET방식은 요청파라미터가 URL의
쿼리스트링으로 전달되므로 그 길이가 제한이
되고 보안상 문제가 있습니다. 반면 POST
방식은 Request Body에 파라미터가
들어가므로 길이 제한이 없고 안전합니다.
이와 같은 HTTP Method는 GET, POST,
PUT, DELETE 등이 있습니다.

6.3 jQuery Ajax 함수 (5/5) – load() 함수

- ✓ load() 함수는 서버로부터 내용을 조회하여, 선택자를 통해 탐색한 DOM 객체에 동적으로 삽입합니다.
- ✓ 첫 번째 인자는 필수 값으로 HTML을 조회할 서버 URL을 지정합니다.
- ✓ 두 번째 인자는 요청 시 서버에 전달할 데이터를 지정합니다.
- ✓ 세 번째 인자는 서버와 통신을 완료한 후에 수행할 콜백함수를 지정합니다.

load 함수

```
$(selector).load(url);
$(selector).load(url, data, function (text, status, jqXHR){});
```

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
</head>
<body>
<h2>KOSTA</h2>
<pre>
    KOSTA는 국내 유일의 기업, 대학, 정부기관 및 연구소 등
    133개 기관 공동참여의 산학연 컨소시엄 형태의 SW 기술 보급 전문단체인
    한국소프트웨어기술진흥협회의 부설 교육원입니다.
</pre>
</body>
</html>
```

```
<script>
$(document).ready(function () {
    $("button").click(function () {
        $("#more").load(
            "/style/htm/info.html");
    });
});
</script>
</head>
<body>
    KOSTA 소개
    <button>더보기</button>
    <div id="more"></div>
</body>
</html>
```

6.4 Ajax 전역함수 (1/2)

- ✓ Ajax는 서버와 통신하는 과정이 웹 브라우저 내부에서 이루어지므로 사용자가 진행상황을 알기 어렵습니다.
- ✓ Ajax 전역함수를 사용하여 Ajax 처리 중에 진행 상태를 보여주는 기능을 구현할 수 있습니다.
- ✓ jQuery는 Ajax 처리가 이루어지는 각 단계 별로 전역함수를 호출합니다.
- ✓ 단, jQuery는 전역함수는 `$.ajaxSetup()` 함수에 global 프로퍼티 설정이 true인 경우에만 수행됩니다. (디폴트 : true)

전역함수	호출시점	매개변수
ajaxStart	Ajax함수실행, XHR객체생성 전	<ul style="list-style-type: none">• 전역 콜백정보 객체
ajaxSend	XHR 인스턴스 생성 뒤, 서버전송 전	<ul style="list-style-type: none">• 전역 콜백정보 객체• XHR 인스턴스• <code>\$.ajax()</code> 가 사용하는 프로퍼티
ajaxSuccess	서버요청 반환, 응답이 성공상태	<ul style="list-style-type: none">• 전역 콜백정보 객체• XHR 인스턴스• <code>\$.ajax()</code>가 사용하는 프로퍼티
ajaxError	서버요청 반환, 응답이 실패상태	<ul style="list-style-type: none">• 전역 콜백정보 객체• XHR 인스턴스• <code>\$.ajax()</code>가 사용하는 프로퍼티• XHR인스턴스가 반환한 예외객체
ajaxComplete	서버요청 반환, ajaxSuccess 또는 ajaxError 호출된 후	<ul style="list-style-type: none">• 전역 콜백정보 객체• XHR 인스턴스• <code>\$.ajax()</code>가 사용하는 프로퍼티
ajaxStop	모든 Ajax진행이 완료, 다른 전역콜백이 호출된 후	<ul style="list-style-type: none">• 전역 콜백정보 객체

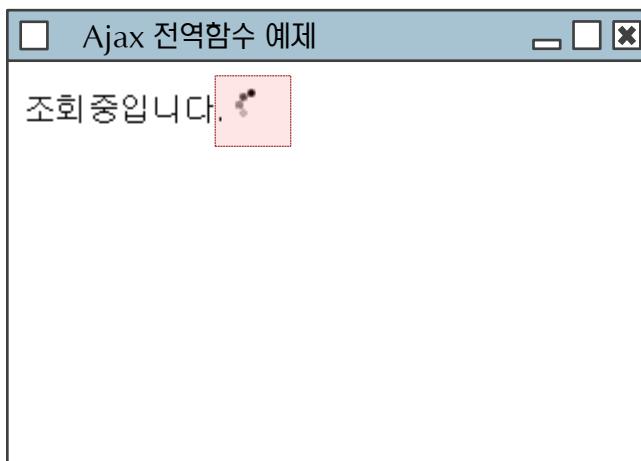


출처 : <http://www.ajaxload.info/>

출처 : https://www.w3schools.com/jquery/jquery_ref_ajax.asp

6.4 Ajax 전역함수 (2/2)

✓ Ajax 전역함수를 이용한 로딩화면 구현



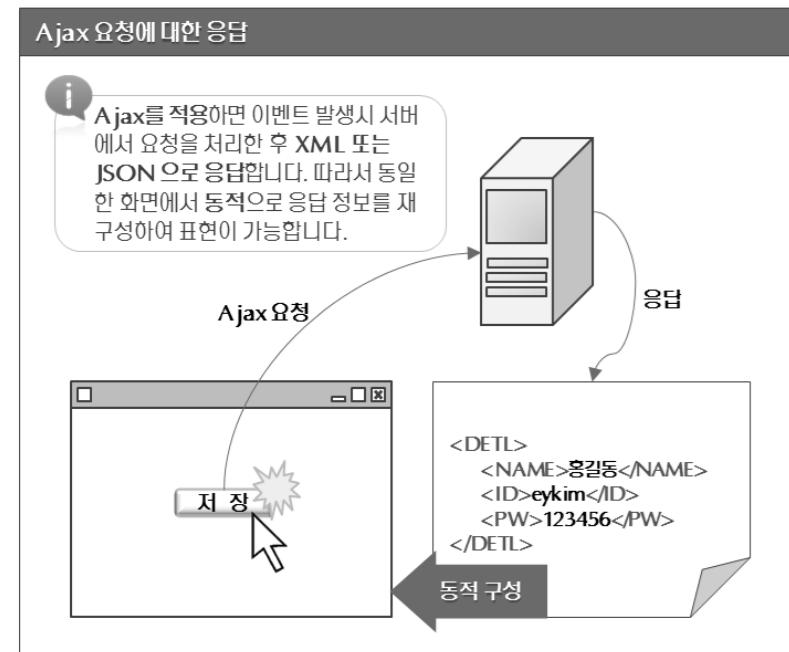
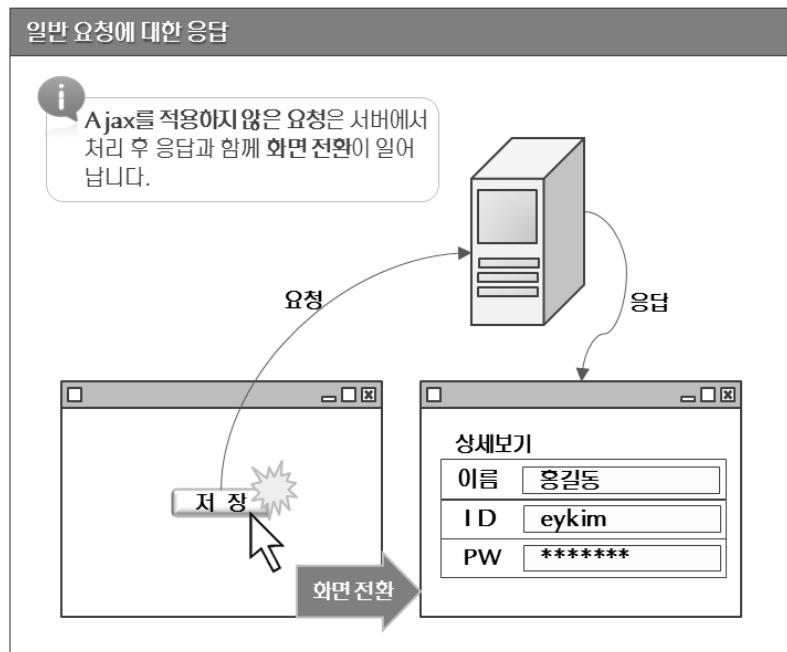
```
<script>
$(document).ready(function () {
    $.ajax({
        url: "/groups/long",
        type: "get",
        dataType: "json",
        success: function (data) {
            for (var i = 0; i < data.length; i++) {
                $("<h2/>")
                    .append(data[i].name + "(" + data[i].code + ")")
                    .appendTo("#div1");
            }
        });
    });

$(document).ajaxStart(function () {
    $("#loading").fadeIn();
}).ajaxStop(function () {
    $("#loading").hide();
});
</script>
```

```
<body>
    <div id="div1">
        <div id="loading" style="display:none;">
            조회중입니다. 
        </div>
    </div>
</body>
```

6.5 요약

- ✓ Ajax 방식을 사용하면 웹(Web)에서 화면을 갱신하지 않고 데이터를 서버로부터 가져와 처리할 수 있습니다.
- ✓ XMLHttpRequest(XHR) 객체는 JavaScript 가 비동기 방식으로 서버와 통신하는 객체입니다.
- ✓ jQuery는 Ajax를 구현할 때 발생하는 크로스 브라우징 문제를 일관된 방식으로 처리할 수 있게 도와줍니다.
- ✓ jQuery는 기본적인 `$.ajax` 함수 외에도 `get()`, `post()`, `load()` 등 다양한 Ajax 함수를 제공합니다.



토론

- ✓ 질문과 대답
- ✓ 토론

