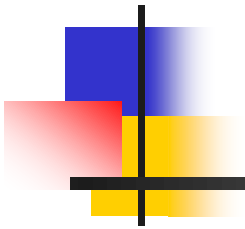


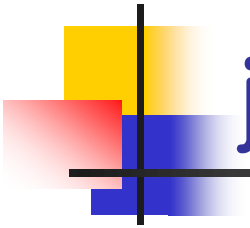
JQuery 1강 - 선택자,문서객체조작

양 명 숙

[now4ever7@gmail.com]

목차

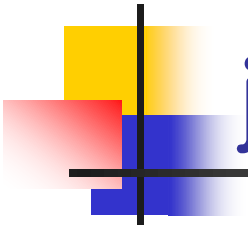
- JQuery란
- jQuery 선택자
  - 기본 선택자
  - 자식 선택자, 후손 선택자
  - 인접 선택자, 이웃 선택자
  - 속성 선택자
  - 필터 선택자
  - each() 메서드
- 문서 객체 조작
  - 조작 메서드
  - CSS 관련 메서드/어트리뷰트 관련 메서드



jQuery

- jQuery란
  - 가장 많이 사용되며, 잘 만들어진 오픈 소스 자바스크립트 라이브러리
  - 가볍고 빠르며, 간결한 오픈 소스 스크립트 라이브러리
  - 2006년 1월, 존 레식(John Resig)이 BarCamp NYC에서 발표

  - Rich 웹 UI를 개발하는 데 도움이 되는 다양한 기능들 지원
    - HTML 문서 트래버스, 이벤트 처리, 애니메이션, Ajax 상호 작용 등 지원
    - => 빠르고 견고하게 리치 웹 애플리케이션을 개발할 수 있도록 지원
  - CSS에서 사용되는 선택자 개념으로 DOM의 멤버에 접근 가능
  - 플러그인 개념을 도입해서 기본기능에서 쉽게 확장 기능으로 업그레이드 가능



jQuery

- jQuery의 기능적인 특징
  - 막강한 CSS 선택자
  - 크로스 브라우저 지원
  - 메서드 체이닝
  - Ajax 지원
  - 풍부한 플러그 인 지원

- **개발 준비사항**
  - jQuery 라이브러리 다운로드
- Jquery 공식 사이트 : http://jquery.com/
  - jQuery의 최신 버전 1.12.4, 2.24, 3.1.1, 3.2.1, 3.3.1, 3.4.1
  - 작업하는 어떠한 웹 애플리케이션에든지 복사해서 쓰면 됨

CDN (Content Delevery Network)
- 사용자에게 간편하게 콘텐츠를 제공하는 방식
- 구글, 마이크로소프트, jQuery측에서 제공

jQuery 사용

- 두 가지 방법으로 사용
- [1]CDN 호스트 사용
  - 다운로드 페이지에서 Jquery의 CDN 호스트 주소 이용

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
```

- [2]직접 다운로드 받아 사용

```
<script src="jquery/ jquery-3.3.1.js "  type="text/javascript"></script>
```

예제1-자바스크립트

1. 클릭하면 사라짐
2. 클릭하면 사라짐
3. 클릭하면 사라짐

```
<script type="text/javascript">
        function disappear(element) {
            element.style.display = "none";
        }
    </script>
</head>

<body>
    <div onclick="disappear(this)">1. 클릭하면 사라짐</div>

    <div onclick="disappear(this)">2. 클릭하면 사라짐</div>

    <div onclick="disappear(this)">3. 클릭하면 사라짐</div>
</body>
```

예제2

```
<script type="text/javascript">
        function init() {
            var divElement = document.getElementsByTagName("div");
            for (var i = 0; i < divElement.length; i++) {
                //var elem = divElement.item(i);
                var elem = divElement[i];
                elem.onclick = function() {
                    this.style.display = "none";
                }
            }//for
        }
</script>
</head>

<body onload="init()">
    <div>1. 클릭하면 사라짐</div>

    <div>2. 클릭하면 사라짐</div>

    <div>3. 클릭하면 사라짐</div>
</body>
```

예제3 - JQuery 이용

```
<script src="../../jquery/jquery-1.11.3.js" type="text/javascript"></script>

<script type="text/javascript">
    $(document).ready(function() {
        $("div").click(function() {
            $(this).hide();
        })
    });
</script>
</head>

<body>
    <div>1. 클릭하면 사라짐</div>

    <div>2. 클릭하면 사라짐</div>

    <div>3. 클릭하면 사라짐</div>
</body>
```

Html코드에 javascript 코드들이 뒤섞여있지 않고, 완전한 분리 가능
- 개발시, 유지보수시 큰 이점

$()

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

- jQuery의 진입점(엔트리 포인트)
- jQuery가 제공하는 페이지 로드 이벤트 함수
- 문서의 DOM 요소들을 조작 가능한 시점이 되면 자동으로 호출되는 이벤트 메서드
- window.onload 이벤트와 유사
- 메서드의 인자로는 델리게이트 함수명을 기입하거나, 익명 함수를 작성

- $(document).ready(function(){... }); 자바스크립트 익명 메서드 호출
- 익명 함수
  - function(){... }
- $()로 CSS의 선택자와 모든 DOM 요소 접근

```
$("선택자").메서드();
```

- $() == JQuery() 동일

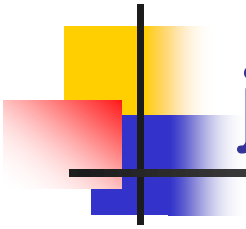
익명함수

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

function pageLoad()
{
      //.. 페이지 로드시에 할 일
}
```

- 일반적으로 pageLoad라는 함수를 별도로 작성하지 않고, 익명 함수로 작성

```
$(document).ready(function(){
        //.. 페이지 로드시에 할 일
  }
);
```



jQuery의 진입점(엔트리 포인트)

- [1] 첫 번째 방법
  - $(document).ready()
- [2] 두 번째 방법
  - $().ready()
- [3] 세 번째 방법
  - $()
    - 예) $(function(){... });

선택자(Selector)

- <body> 태그내의 모든 DOM 요소에 접근 가능
- $("selector")
  - Selector 로 읽어오는 개체의 반환값은 Jquery에서 사용할 수 있는 개체
- CSS 선택자를 그대로 사용

```
$("선택자").메서드();

$('h1').css('color','blue');
```

css() 메서드 - 대상 개체에 css 스타일을 적용하는 메서드

CSS 선택자

- CSS 선택자 – css 파일에서 사용했던 표현식

```
div p {
    font-color:red;
}

#tel {
    font-weight:bold;
    background:yellow;
}

.spring {
    padding:10px;
    background:white;
}
```

- div p 선택자 – 현재 문서 상에서 div 요소의 후손으로 존재하는 모든 p 요소들에 적용
- #tel – tel 이라는 id 값을 가진 요소에 적용
- .spring – class 어트리뷰트 값으로 spring를 갖는 모든 요소에 적용

선택자

$(selector) 또는 jQuery(selector)

- jQuery에서 원하는 DOM 요소의 그룹을 찾기 위해서 사용하는 표현식

```
$("div p") 또는 jQuery("div p")
$("#tel") 또는 jQuery("#tel")
$(".spring") 또는 jQuery(".spring")
```

- 각각의 표현식은 각 DOM 요소의 확장 개체인 jQuery 개체 집합을 반환함
  - DOM 요소를 직접 반환해주는 것이 아니라 그의 래퍼(Wrapper)인 jQuery 개체로 반환해 주기 때문에 직접 DOM 요소를 제어할 때보다 훨씬 편하고 쉽게 개체를 제어할 수 있음
- 예) $("div p").hide()
  - 현재 문서 상에서 div 요소의 모든 후손 p 태그집합들은 보이지 않게 됨
  - hide() – jQuery 개체가 지원하는 명령. jQuery 선택자에 의해 반환되는 개체가 일반 DOM 개체라면 이러한 명령을 사용할 수 없겠지만, jQuery 개체 집합으로 반환되기에 그러한 명령을 사용할 수 있음

기본 선택자

선택자의 종류	표현 방법
All Selector (전체 선택자)	$('*')
ID Selector (아이디 선택자)	$('#id')
Class Selector (클래스 선택자)	$('.className')
Tag Selector (태그 선택자)	$('tagName')

• 여러 개의 태그 선택
- 콤마로 선택자를 구분
$('span, h1, p').css('color','blue');

기본 선택자

- [1] 전체 선택자
  - html 페이지에 있는 모든 문서 객체를 선택하는 선택자

```
$('*').css('color', 'blue');
```

- [2] 태그 선택자
  - 특정한 태그를 선택하는 선택자
  - 태그의 이름을 사용

```
$('span').css('color', 'red');
```

- [3] 아이디 선택자
  - 특정한 id 속성이 있는 문서 객체를 선택하는 선택자
  - # 을 이용

```
<p id='ph1'>문단</p>

$('#ph1').css('color','blue');
```

기본 선택자

- [4] 클래스 선택자
  - 특정한 class 속성이 있는 문서 객체를 선택하는 선택자

```
<p class='ph'>문단1</p>
<p class='ph red'>문단2</p>
<span class='ph'>안녕</span>

$('.ph').css('color','gray');
$('span.ph').css('color','cyan');
$('.ph.red').css('color','red');
```

  - 두 클래스 속성을 모두 갖는 문서 객체를 선택 – 두 클래스 선택자를 붙여서 사용

자식 선택자, 후손 선택자

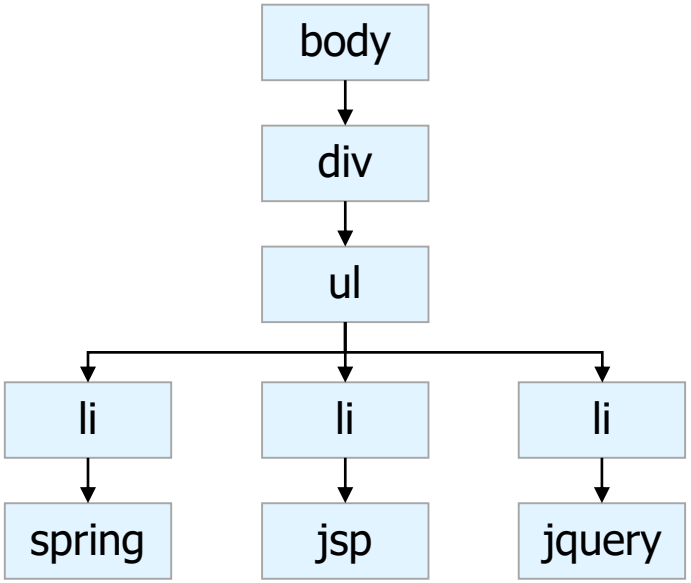
선택자의 종류	표현방법
Child Selector (자식 선택자)	$('parent > child')
Descendant Selector (후손 선택자)	$('ancestor descendant');

$("p a") : p요소 하위에 존재하는 모든 a 요소를 선택
$("p > a") : p 요소 바로 아래 자식인 a 요소 선택

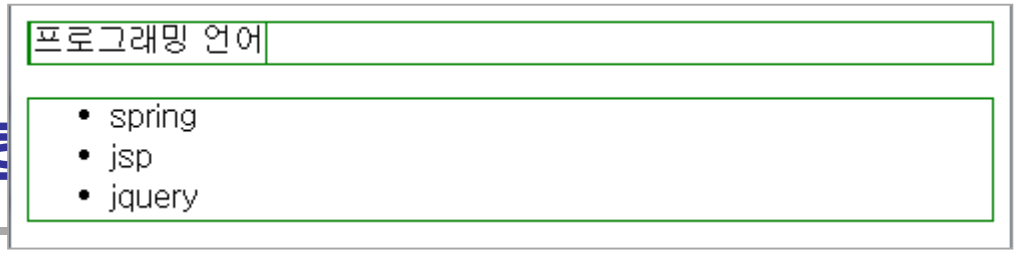
자식 선택자, 후손 선택자

- html 문서는 DOM(Document Object Model : 문서 객체 모델) 형식을 취하는 트리구조로 되어 있다
  - div 태그는 body 태그의 자식
  - body 태그는 div 태그의 부모
  - body 태그 밑에 있는 모든 것들을 body태그의 후손이라고 부름

```
<body>
        <div>
                <ul>
                        <li>spring</li>
                        <li>jsp</li>
                        <li>jquery</li>
                </ul>
        </div>
</body>
```



자식 선택자, ㅎ



- [1] 자식 선택자
  - 자식을 선택하는 선택자로 $('부모 > 자식') 의 형태로 사용

```
<body>
        <div>
                <div>
                        <span>프로그래밍 언어</span>
                </div>
                <ul>
                        <li>spring</li>
                        <li>jsp</li>
                        <li>jquery</li>
                </ul>
        </div>
</body>
```

```
$('div > *').css('border', '1px solid green');
```

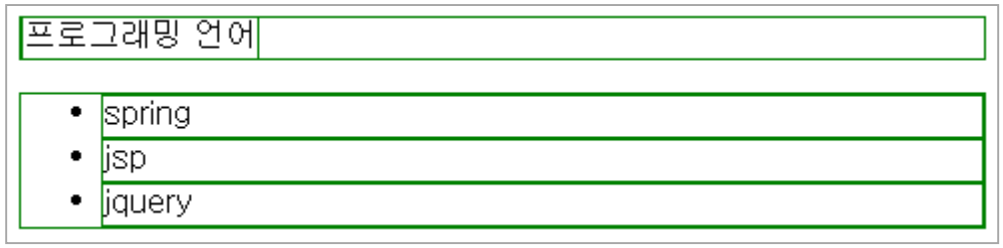
- div 태그 밑의 자식을 선택해서 테두리를 그려라

자식 선택자, 후손 선택자

- [2] 후손 선택자
  - 부모의 밑의 모든 것을 선택할 때 사용
  - $('부모 후손') 의 형태로 사용

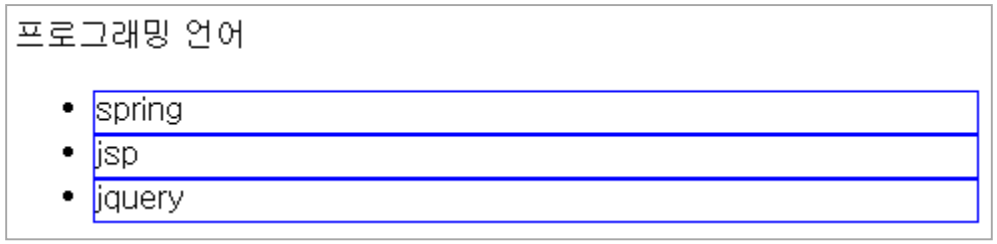
```
$('div *').css('border', '1px solid green');
```

- 문서내의 div 태그를 찾아서 그의 밑에 있는 모든 엘리먼트에 테두리를 그려라




```
$('div > li').css('border', '1px solid red');
$('div li').css('border', '1px solid blue');
```

- div의 자식인 li 는 없다
- div의 후손중에는 li 가 있기 때문에 li 태그들에 파란색 테두리가 그려짐



인접 선택자, 이웃 선택자

선택자의 종류	표현 방법
Next Adjacent Selector (인접 선택자)	$('prev + next');
Next Siblings Selector (이웃 선택자)	$('prev ~ siblings');

$("div + p") : div 요소의 바로 다음에 나오는 형제 p 요소 선택
$("div ~ p") : div 요소의 다음에 나오는 모든 형제 p 요소 선택

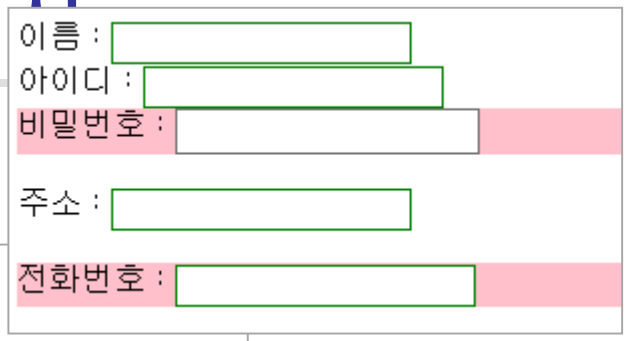
인접 선택자, 이웃 선택자

- 자식 선택자(Child Selector)와 후손 선택자(Descendant Selector)는 서로 상하 관계에 있는 요소를 선택할때 사용
- 인접 선택자와 이웃 선택자는 평등한 관계에 있는 요소를 선택할 때 사용
- 평등한 요소들을 형제요소라 부름

- 인접 선택자, + 기호 : 앞의 선택자 바로 뒤에 오는 형제 선택자를 선택
- 이웃 선택자, ~ 기호 : 앞의 선택자 **뒤에** 오는 형제 요소들 중에 뒤의 선택자를 선택

인접 선택자, 이웃 선택자

```
$('label + input').css('border', '1px solid green');
$('div.start ~ div').css('background', 'pink');
```



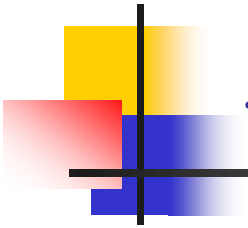

```
<body>
        <div>
                <label>이름 : </label><input type="text">
        </div>
        <div class="start">
                <label>아이디 : </label><input type="text">
        </div>
        <div>
                <span>비밀번호 : </span><input type="text">
        </div>
        <p>
                <label>주소 : </label><input type="text">
        </p>
        <div>
                <label>전화번호 : </label><input type="text">
        </div>
</body>
```

body 밑에 있는 div,div,div,p,div 는 서로 형제 요소
그 안에 있는 label과 input 은 서로 형제 요소

속성 선택자

- 기본 선택자 뒤에 붙여 사용함

형식(선택자)	설명
$(Selector[속성])	attr(속성)을 가지는 Selector
$(Selector[속성="value"])	attr(속성)의 값이 value와 동일한 Selector
$(Selector[속성!="value"])	attr(속성)의 값이 value와 다른 Selector
$(Selector[속성^="value"])	attr(속성)의 값이 value로 시작하는 Selector
$(Selector[속성$="value"])	attr(속성)의 값이 value로 끝나는 동일한 Selector
$(Selector[속성*="value"])	attr(속성)의 값이 value를 포함하는 Selector
$(Selector[속성~="value"])	attr(속성)의 값에 value가 단어로써 포함되어 있는 Selector



속성 선택자

- a[title]
  - title 어트리뷰트를 갖는 하이퍼링크 선택
- a[href^="mailto:"]
  - href 값이 mailto로 시작하는 하이퍼링크 선택
- a[href$=".pdf"]
  - .pdf로 끝나는 모든 하이퍼링크 선택
- a[href*="herb.com"]
  - herb.com이라는 값이 포함되어 있는 모든 하이퍼링크 선택
- input[type="text"]
  - Type이 text 형식인 입력 컨트롤 선택

속성 선택자

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

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

```
$('img[alt]').css('border', '3px solid blue');
```
img 태그 중에 alt 속성값이 있는 태그에 테두리 그린다

img 태그중에 alt 속성값이 "사과" 인것만 선택해서 파란 테두리주기
```
$('img[alt="사과"]').css('border', '3px solid blue');
```

img 태그중에 alt 속성값이 "사과" 가 아닌것만 선택해서 파란 테두리주기
```
$('img[alt!="사과"]').css('border', '3px solid blue');
```

img 태그중에 alt 속성값이 "이"로 시작하는 것만 선택해서 파란 테두리주기
```
$('img[alt^="이"]').css('border', '3px solid blue');
```

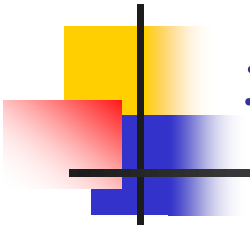
img 태그중에 alt 속성값이 "보영"으로 끝나는 것만 선택해서 파란 테두리주기
```
$('img[alt$="보영"]').css('border', '3px solid blue');
```

img 태그중에 alt 속성값에 "희"가 들어있는 것만 선택해서 파란 테두리주기
```
$('img[alt*="희"]').css('border', '3px solid blue');
```
- 이름에 희가 들어가면 선택됨

img 태그중에 alt 속성값이 단어로써 "보영"이 있는 것만 선택해서 파란 테두리주기
```
$('img[alt~="보영"]').css('border', '3px solid blue');
```
박보영은 붙여 썼고 이보영은 이 보영 이라고 썼다면, 단어로써 보영이 들어가있는 이 보영이 선택됨

jQuery가 지원하는 선택자들

선택자	설명
*	모든 요소와 일치
E1	E1(태그명)인 모든 요소와 일치
E1.class	E1(태그명) 요소의 클래스가 class와 동일한 요소와 일치
E1.#id	E1(태그명) 요소의 id 어트리뷰트 값이 id와 동일한 요소와 일치
E1 E2	E1 요소의 후손인 **모든** E2(태그명) 요소와 일치
E1 > E2	E1 요소 바로 아래 자식인 E2 요소와 일치
E1 + E2	E1 요소의 바로 다음에 나오는 형제요소 E2와 일치
E1 ~ E2	E1 요소의 다음에 나오는 **모든** 형제 E2와 일치
E1[*attr*]	*attr* 어트리뷰트를 갖는 E1 요소와 일치
E1[*attr*=*val*]	*attr* 어트리뷰트의 값이 *val*을 갖는 E1 요소와 일치
E1[*attr*^=*val*]	*attr* 어트리뷰트의 값이 *val* 값으로 시작하는 E1 요소와 일치
E1[*attr*$=*val*]	*attr* 어트리뷰트의 값이 *val* 값으로 끝나는 E1 요소와 일치
E1[*attr**=*val*]	*attr* 어트리뷰트의 값이 *val* 값을 포함하는 E1 요소와 일치



필터 선택자

- 선택자 중에 : 기호를 포함하는 선택자
- 기본 선택자 뒤에 사용
- 필터
  - 걸러내는 역할
  - 다양한 방식으로 원하는 요소들을 필터링하여 선택할 수 있게 함

필터 선택자

:first	선택된 개체들 중 첫 번째 요소와 일치함
:last	선택된 개체들 중 마지막 요소와 일치
:even	짝수 요소들과 일치함 (0부터 시작)
:odd	홀수 요소들과 일치함 (0부터 시작)
:not(selector)	괄호에 주어진 선택자와 일치되는 모든 요소를 제외함
:eq(index)	인덱스에 해당하는 단일 요소와 일치
:gt(index)	주어진 인덱스보다 높은 인덱스를 갖는 모든 요소와 일치
:lt(index)	주어진 인덱스보다 낮은 인덱스를 갖는 모든 요소와 일치
:contains(문자열)	특정 문자열을 포함하는 문서 객체를 선택
:nth-child(3n+1)	3n+1 번째에 위치하는 문서 객체를 선택(1,4,7…)
:parent	부모인 모든 요소들과 일치됨 자식 요소를 갖는 요소 뿐만 아니라 텍스트를 갖는 요소들이 이에 해당됨

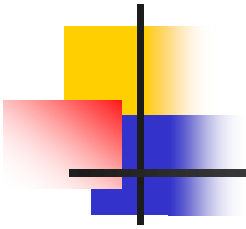
필터 선택자

- 예)

```
tr:odd
tr:gt(4)
tr:last
```

- tr:odd
  - 현재 문서에 존재하는 모든 tr 중 홀수 번째 tr 만을 선택
- tr:gt(4)
  - 문서에 존재하는 모든 tr 중 그 인덱스가 4 보다 큰 모든 tr들을 선택
- tr:last
  - 모든 tr 중 가장 마지막 tr을 선택하는 선택자

- 인덱스는 0부터 시작
  - 첫번째로 나오는 tr은 인덱스 0을 가지므로, 홀수가 아니라 짝수임



메서드 체이닝(chaining)

- 메서드 체이닝
  - 메서드를 연속으로 사용
  - jQuery가 제공하는 모든 메서드는 그 반환값(return value)이 효과가 반영된 jQuery 개체이기 때문에 **메서드를 이어서 사용**할 수가 있음

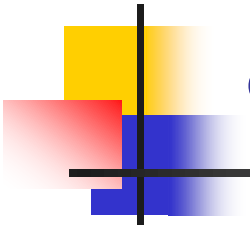
- $("tr:odd").css("font-weight", "bold").css("color", "blue");
  - 선택자 대상에 대해서 css()라는 메서드를 연달아 사용

입력 양식 필터 선택자

:input	모든 input, textarea, select, button요소들과 일치
:text	text 타입의 모든 input 요소들과 일치
:password	password타입의 모든 input 요소들과 일치
:radio	radio 타입의 모든 input 요소들과 일치
:checkbox	checkbox 타입의 모든 input 요소들과 일치
:submit	submit 타입의 모든 input 요소들과 일치
:image	image 타입의 모든 input 요소들과 일치
:reset	reset 타입의 모든 input 요소들과 일치
:button	모든 button 요소들 및 button 타입의 input 요소들과 일치
:file	file 타입의 모든 input 요소들과 일치
:hidden	hidden 상태인 모든 요소들과 일치

입력 양식 필터 선택자

:enabled	현재 enable 상태인 요소를 선택
:disabled	현재 disable 상태인 요소를 선택
:checked	체크된 요소를 선택
:selected	선택된 요소를 선택
:focus	초점이 맞추어져 있는 입력 양식을 선택

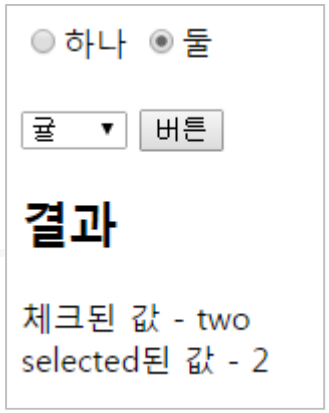


입력 양식 필터 선택자

- :checked
  - 체크박스 중 체크된 것을 선택
- :selected
  - select 컨트롤들 중에서 선택된 것을 선택

- => 자주 사용됨

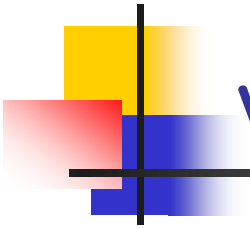
예제




```
<script type="text/javascript">
   $(document).ready(function() {
     $("#btn1").click(function() {
            var str="체크된 값 - " + $("input[type=radio]:checked").val() + "<br>";
              str += "selected된 값 - " + $("select > option:selected").val() + "<br>";
              $("#div1").html(str);
           //$("#div1").text(str);
       });
   });
</script>

<body>
   <input id="rd1" type="radio" name="num" value="one" >하나
   <input id="rd2" type="radio" name="num" value="two" checked>둘<br><br>
   <select id="sel1">
      <option value="1">사과</option>
      <option value="2">귤</option>
      <option value="3" selected>배</option>
   </select>
   <input id="btn1" type="button" value="버튼" />

   <h2>결과</h2>
   <div id="div1"></div>
</body>
```



val()

- val() 메서드
  - 엘리먼트(요소)의 value 속성 값을 문자열로 리턴함

```
예)<input type="text" value="hi" />
$("input").val() => hi
```

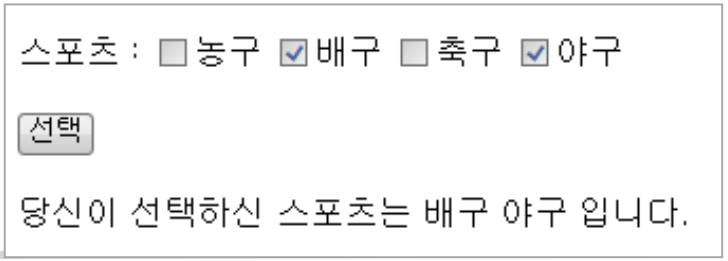
- val(값) 메서드
  - 대상 엘리먼트에 입력 값으로 받은 값을 적용함

```
$("#tx1").val('hello');
```

```
document.getElementById("tx1").value='hello';
```

위와 동일

예제




```
<script src="../../jquery/jquery-1.11.3.js" type="text/javascript"></script>
<script type="text/javascript">
      $(document).ready(function() {
         $("#btn1").click(function() {
            var sports = $("input:checked");
            var str = "";
            for (var i = 0; i < sports.length; i++) {
                  str += $(sports[i]).next().text() + " ";
            }
            $("#result").text(str);
         });
    });
</script>
  <p>
     스포츠 : <input id="c1" type="checkbox" /><span>농구</span>
     <input id="c2" type="checkbox" /><span>배구</span>
     <input id="c3" type="checkbox" /><span>축구</span>
     <input id="c4" type="checkbox" /><span>야구</span>
  </p>
  <p><input id="btn1" type="button" value="선택" /></p>
  <p>
     당신이 선택하신 스포츠는 <span id="result"></span> 입니다.
  </p>
```

```
$('input:checked').each(function(){
            str+=$(this).next().text() +" ";
});
$("#result").text(str);
```

$(선택자).click(*function*)

```
$("#btn1").click( someFunction );
```

- #btn1이라는 개체에게 동적으로 클릭 이벤트를 걸어주는 jQuery 코드

```
document.getElementById("btn1").click = someFunction; 와 동일
```

- 메서드 체이닝을 통해서 여러 작업을 이어서 지정할 수 있다
  - 예)

```
$("#btn1").click( someFunction ).mouseover( otherFunction ).css("color", "red");
```

- 클릭 이벤트와 마우스 오버 이벤트를 걸어줌과 동시에 color 스타일 값도 red로 변경하는 작업을 한번에 할 수 있음

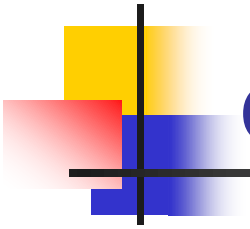
이벤트 연결 메서드 – on()

- 이벤트 연결 작업은 on 메서드를 통해서 설정해야 함

```
$("#btn1").on("click", someFunction );
```

- jQuery는 자주 사용하는 몇몇 이벤트에 대해서는 click과 같이 단축 메서드를 작성해 놓았기에, on 대신 click과 같은 메서드를 사용해서도 이벤트 핸들러를 지정할 수 있음

```
$("#btn1").click( someFunction );
```



click()

- click() 메서드도 인자로는 함수(클릭 시에 해야 할 작업을 작성하는 함수)를 지정할 수 있게 되어 있음
  - 주로 익명함수로 작성

```
$("#btn1").click(function() {
        //클릭 시 해야할 작업
});
```

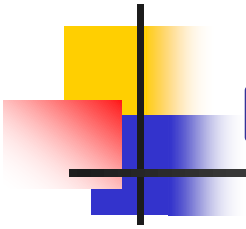
  - click 메서드의 인자로 지정된 익명함수 내에서는 원래의 개체에 대한 컨텍스트가 유지되기 때문에,
  - 함수 내에서 this라는 키워드를 사용할 수 있으며,
  - 이 this는 btn1이라는 html 요소를 의미
  - jQuery 개체로 다시 감싸고 싶다면, 함수 내부에서 **$(this)**와 같이 작성하면 됨

예제

```
var sports = $("input:checked");
var result = "";
for (var i = 0; i < sports.length; i++) {
        result += $(sports[i]).next().text() + " ";
}
$("#result").text(result);
```

- sports 변수는 선택자에 의해 필터링된 html 요소들의 배열
- 그들을 루프를 돌면서, 각 체크박스의 다음에 있는 html 요소로 접근한 뒤(next() 메서드), 그 요소의 텍스트값을 얻어와서 문자열로 구성

```
$('input:checked').each(function(){
        str+=$(this).next().text() +" ";
});
$("#result").text(str);
```



next()

- jQuery 메서드 – next(), text()
- **next()**
  - 현재의 개체 요소와 **형제** 수준인 요소 중 **바로 다음**에 나오는 개체를 얻어오는 메서드
  - 각각의 체크박스 바로 옆에 개체들이 놓여 있으므로, $(sports[i]).next() – 현재의 체크박스 바로 옆에 있는 span 개체를 찾아 감
  - sports[i].next() 가 아니라, $(sports[i]).next()
    - sports라는 배열에 들어있는 개체들은 jQuery 개체가 아닌 html 요소일 수 있기 때문에 $()를 사용하여 확실하게 jQuery 개체로 포장
    - jQuery 개체가 아니면 next() 메서드를 사용할 수 없으므로

text(), html()

text() 메서드
- 해당 개체가 가지고 있는 컨텐츠를 텍스트로 반환하는 메서드

html() 메서드
- 개체가 포함하고 있는 html 컨텐츠를 반환하는 메서드
이 메서드들은 인자를 사용하지 않으면 값을 가져오는 역할을 하고
인자로 값을 지정하면 그 값으로 원래의 값을 바꾸는 역할을 함

- text() 메서드 – 개체의 컨텐츠가 html 요소들을 포함하고 있다고 해도, 다 무시하고 텍스트만을 반환함
  - 예) test
    - $("span").text() => test
    - $("span").html() => test

each() 메서드

```
$('input:checked').each(function(){
        str+=$(this).next().text() +" ";
});
$("#result").text(str);
```

- 2개의 체크박스가 체크되었다면, each() 안의 함수는 그 2개의 체크박스에 대해 각각 호출될 것이고, 구문 내 this는 각각의 호출 시에 그에 해당하는 체크박스가 됨

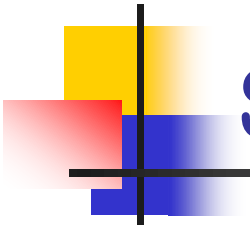
- 기존의 for 문으로 작성한 것보다 훨씬 간단하고 명료

each() 메서드

- each() 메서드를 사용하게 되면, 메서드 체이닝도 사용 가능

- ```
  $("#btn1").click(function() {
    var result = "";
    $("input:checked").each(function(idx, item) {
        result += $(this).next().text() + ",";
      })
      .css("background", "yellow");

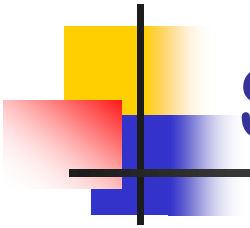
    $("#result").text(result);
  });
  ```
- each() 메서드에 이어서 css() 메서드를 체인으로 연결해서 작성
- 체크된 모든 체크박스에 대해서 each() 메서드를 사용한 기존 작업을 수행할 뿐 아니라, 배경색을 노란색으로 지정하는 것까지 수행하게 됨
  ```



$.each()

- $.each()
  - 배열과 객체를 순회할 목적의 메서드
  - 순차적으로 특정 함수를 수행할 수 있게 해주는 메서드
  - 매개변수로 입력한 함수로 for 반복문처럼 객체나 배열의 요소를 검사하는 메서드

  - 선택자에 의해 추출된 개체들에 대해서 각각 지정한 함수를 수행하게 함
  - 루프를 사용할 때와 마찬가지로 각 요소에 대해 어떤 작업을 하고 싶다면 바로 그 함수 안에다가 수행할 내용을 작성하면 됨
  - 이는 for 루프문을 대체할 수 있음
  - 객체를 적용하면 객체의 모든 속성과 값이 전달됨



$.each()

- each 메서드의 2가지 사용법

[1] $.each(배열객체, function(index, item){});

  - 자바스크립트 배열 관리
  - 배열 객체의 모든 요소에서 function을 수행
  - index에는 배열의 인덱스(또는 객체의 키)가 전달되고,
  - item에는 배열의 데이터(해당 인덱스나 키가 가진 값)가 전달됨
  - 배열이 아닌 일반 객체를 적용하면 객체의 모든 속성과 값이 전달됨

예제

[0]홍길동 - hong

[1]김길동 - kim

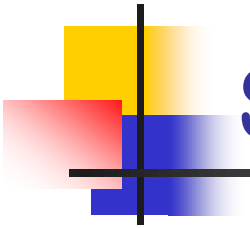
[2]이길동 - lee

```
<script src="../../jquery/jquery-1.11.3.js" type="text/javascript"></script>
    <script type="text/javascript">
        $(document).ready(function() {
            var arr = [{name:'홍길동', id:'hong'},
                       {name:'김길동', id:'kim'},
                       {name:'이길동', id:'lee'}];

            var str="";
            $.each(arr, function(index, item){
                str+="<p>["+index+"]"+item.name+" - " +item['id']+"</p>";
            });
            $("#result").html(str);
        });
    </script>
</head>
<body>
   <div id="result"></div>
</body>
```

```
var arr2 = ["java","jsp","oracle"];

str="";
$.each(arr2, function(index, item){
        str+="<p>"+index+" - "+item+"</p>";
});
$("#result2").html(str);
```



$(selector).each()

[2] $(selector).each(function(index, item){});

- jQuery 배열 관리
- jQuery 배열 객체는 따로 만드는 것이 아니라, 선택자로 여러 개의 문서 객체를 선택할 때 생성됨
- 모든 selector(jQuery의 배열)에게 function을 수행

예제

```
<p style="font-size: 10px; background: yellow;">문단1</p>
<p style="font-size: 20px; background: yellow;">문단2</p>
<p style="font-size: 30px; background: yellow;">문단3</p>
```

each - 선택자로 선택된 여러 개 요소들을 개별적으로 처리할 때 사용

```
<script src="../../jquery/jquery-1.11.3.js" type="text/javascript"></script>
    <script type="text/javascript">
        $(document).ready(function() {
            $('p').each(function(index, item){
                //$(item).css('font-size', (index+1)*10);  //아래와 동일
                $(this).css('font-size', (index+1)*10);
            });

            $('p').css('background','yellow');
        });
    </script>
</head>
<body>
    <p>문단1</p>
    <p>문단2</p>
    <p>문단3</p>
</body>
```

문단1

문단2

문단3

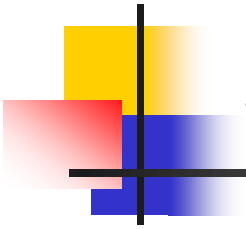
예제


```
<script type="text/javascript">
        $(document).ready(function() {
            var result = "";
            $("p").each(function(i) {
                result += i + "번째 p태그 : " + $(this).html() + "₩n";
            });

            alert(result);

            alert($('p').html());  //one
        });

    </script>
</head>
<body>
    <p>one</p>
    <div>
        <p>two</p>
    </div>
    <p>three</p>
</body>
```



실습-each

- ※ each 실습
  - 1. 모든 div태그의 글자색을 파랑색으로 지정
  - 2. 각 div 태그의 텍스트를 div100, div200, div300,div400으로 변경
    - 각 div 태그의 border를 1,2,3,4px로 지정
  - 3. span태그의 텍스트를 모두 읽어와서 메세지창에 출력하기
  - 4. 버튼 클릭시 select태그에서 선택된 모든 option의 텍스트를 읽어와서 id가 selInfo인 div에 출력하기

실습-each

```
<body>
<h1>each실습</h1>
<div>div태그1</div>
<div>div태그2</div>
<div>div태그3</div>
<div>div태그4</div>
<br>
<span>span태그1</span>
<span>span태그1</span>
<span>span태그1</span>
<br>
<hr>
<select name='sel1' multiple="multiple">
		<option>java</option>
		<option>jsp</option>
		<option>spring</option>
		<option>oracle</option>
</select><input type="button" value='선택'>
<br>
<div id="selInfo"></div>
</body>
```

예제-text()

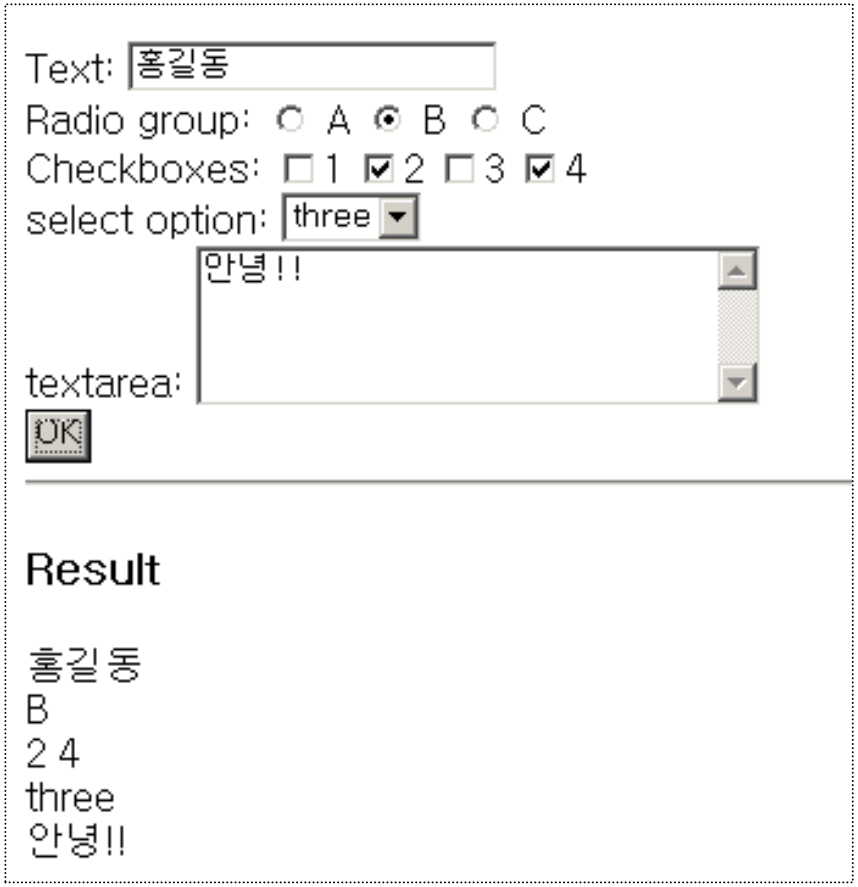

```
<script src="../../jquery/jquery-1.11.3.js" type="text/javascript"></script>
<script type="text/javascript">
    $(document).ready(function() {
        // 드롭다운 리스트가 변경될 때
        $('#selLang').change(function() {
            // 드롭다운리스트에서 선택된 값을 텍스트박스에 출력
            var selectedText = $(":selected").text();
                // $("#selLang option:selected").text();
                // $("option:selected").text();
            $('#txtLang').val(selectedText);
            $("#result").text(selectedText);
        });
    });
</script>

<select id="selLang">
    <option>jsp</option>
    <option>spring</option>
    <option>JQuery</option>
</select>
<br><br>

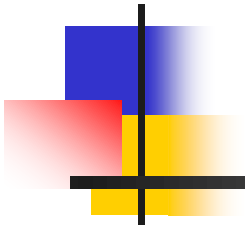
<h3>선택된 값</h3>
<input type="text" id="txtLang" />
<span id="result"></span>
```

실습-selectorEx1.jsp

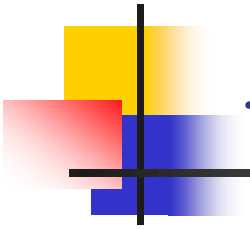
- 버튼 클릭시 입력한 값과 선택한 값을 div 태그에 출력하기



radio => value
checkbox, select => text

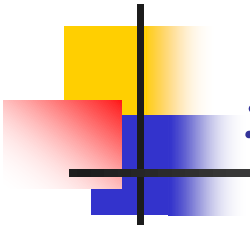


문서 객체 조작



목차

- 문서객체 조작
  - 조작(Manipulation) 메서드
    - 요소의 내용을 조작하는 메서드
    - 요소를 추가하는 메서드
      - 요소 내부에 추가하는 메서드
      - 요소 외부에 추가하는 메서드
    - 요소를 제거하거나 복사하는 메서드
  - CSS 관련 메서드
  - 어트리뷰트 관련 메서드



조작(Manipulation) 메서드

- 조작 관련 메서드
  - 요소에 값을 지정하거나, 특정 요소의 값을 읽어오거나, 동적으로 요소 자체를 생성, 삭제, 복사, 제거하는 기능들
  - 요소의 내용을 조작하는 메서드
  - 요소를 추가하는 메서드
  - 요소를 래핑하거나 바꿔치기 하는 메서드
  - 요소를 제거하거나 복사하는 메서드

[1] 요소의 내용을 다루는 메소드

내용 변경 메서드	
html()	일치된 요소의 html 내용을 가져옴 이는 요소의 innerHTML 값과 동일 만일, 일치된 요소가 여러 개라면 그 중 첫 번째 요소의 HTML을 가져옴
html(*val*)	일치된 요소의 html 본문을 *val* 값으로 설정함 만일, 일치된 요소가 여러 개라면 모든 요소에 이러한 작업을 수행함
text()	일치된 모든 요소의 텍스트를 합쳐서 가져옴
text(*val*)	모든 일치된 요소의 텍스트를 *val* 값으로 설정함

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

요소의 내용을 다루는 메소드

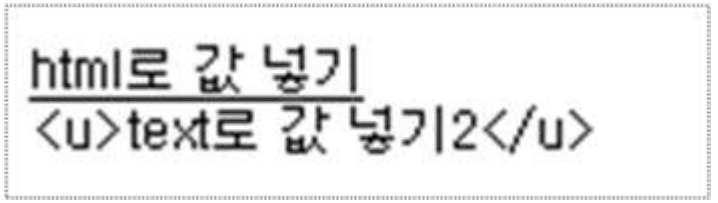
- text() 메서드
  - 해당 개체가 가지고 있는 컨텐트를 텍스트로 반환하는 메서드
  - 개체의 컨텐트가 html 요소들을 포함하고 있다고 해도, 텍스트만을 반환함
- html() 메서드
  - 개체가 포함하고 있는 html 컨텐트를 반환하는 메서드

```
예) <span><b>test</b></span>
    $("span").text() => 반환값은 test
    $("span").html() => 반환값은 <b>test</b>
```

- jQuery로 검색된 대상 요소가 여러 개일 경우, text() 메서드와 html() 메서드의 동작이 약간 다름
  - 일치된 요소가 여러 개인 경우,

```
text() 메서드 - 일치된 모든 대상의 텍스트를 결합해서 반환
html() 메서드 - 일치된 요소들 중 첫 번째 요소의 html 만을 반환함
```

예제




```
<script src="../../jquery/jquery-1.11.3.js" type="text/javascript"></script>
<script type="text/javascript">
    $(document).ready(function() {
        console.log("html() 결과=>" + $("#sp1").html());
        console.log("text()결과=>" + $("#sp1").text());
        console.log("span html() 결과=>" + $("span").html());
        console.log("span text()결과=>" + $("span").text());

        $("#sp1").html("<u>html로 값 넣기</u>");
        $("span:eq(1)").text("<u>text로 값 넣기2</u>");
    });
</script>
</head>

<body>
   <span id="sp1"><b>test중</b></span><br>
   <span><b>span 태그입니다</b></span><br>
</body>
```

```
html() 결과=><b>test중</b>
text()결과=>test중
span html() 결과=><b>test중</b>
span text()결과=>test중span 태그입니다
```

예제2

```
<script type="text/javascript">
    $(document).ready(function() {
        $('div').html(function (index) {
            return '<p>문단-' + index + '</p>';
        });

        $('h2').html(function (index, html) {
            return '[' + html + ']';   //두번째 매개변수에 원래 있던 html의 내용이 들어감
        });
    });
</script>
<body>
    <div></div>
    <div></div>
    <div></div>

    <h2>제목1</h2>
    <h2>제목2</h2>
    <h2>제목3</h2>
</body>
```

문단-0

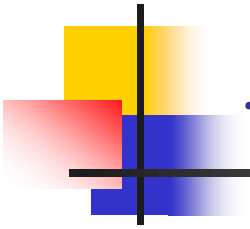
문단-1

문단-2

[제목1]

[제목2]

[제목3]



문서 객체 생성

- $(html태그)
  - 문서 객체를 생성한다
  - 예) $(' 안녕 ')
- $('선택자')
  - 문서 객체를 선택한다

예제

안녕

안녕하세요

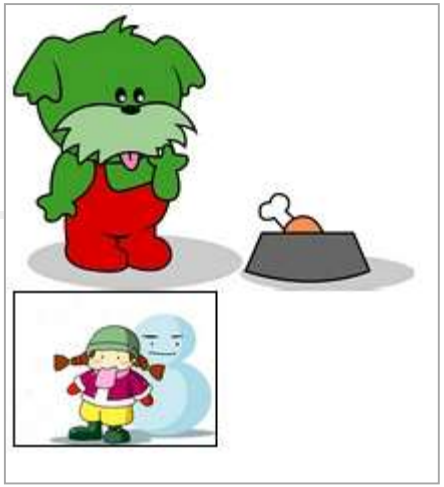
```
<script src="../../jquery/jquery-1.11.3.js" type="text/javascript"></script>
<script>
    $(document).ready(function() {
        $('<h1></h1>').html('안녕').appendTo('#div1');
        $('<h2>안녕하세요</h2>').appendTo('#div2');
    });
</script>

<body>
    <div id="div1">
    </div>
    <div id="div2">
    </div>
</body>
```

- 결과

```
<div id="div1">
    <h1>안녕</h1>
</div>
<div id="div2">
    <h2>안녕하세요</h2>
</div>
```

예제2



```
<script>
    $(document).ready(function () {
        $('<img>').attr('src', 'images/dog.jpg').appendTo('#div1');

        $('<img>', {
          src: 'images/snow.jpg',
          width: 100,
          border:1 }).appendTo('#div2');
    });
</script>

<body>
    <div id="div1">
    </div>
    <div id="div2">
    </div>
</body>
```

[2]요소를 추가하는 메서드

- 특정 요소나 개체 집합을 다른 요소의 "내부"에다가 앞, 뒤로 덧붙이는 기능을 제공하는 메서드들

추가(요소 내부에) 관련 메서드	
append(*content*)	일치된 요소의 내부 뒤쪽에 *content*를 덧붙임
appendTo(*selector*)	일치된 요소를 *selector*에 의해 일치된 모든 요소들 내부에 덧붙임 만일, 일치된 요소가 본문에 존재하는 개체(예, $("#some"))라면 그를 제거한 후 복사(즉, 이동)함
prepend(*content*)	append(*content*)와 동일 다만, 내부 앞쪽에 붙여 넣음
prependTo(*selector*)	appendTo(*selector*)와 동일 다만, 내부 앞쪽에 붙여 넣음

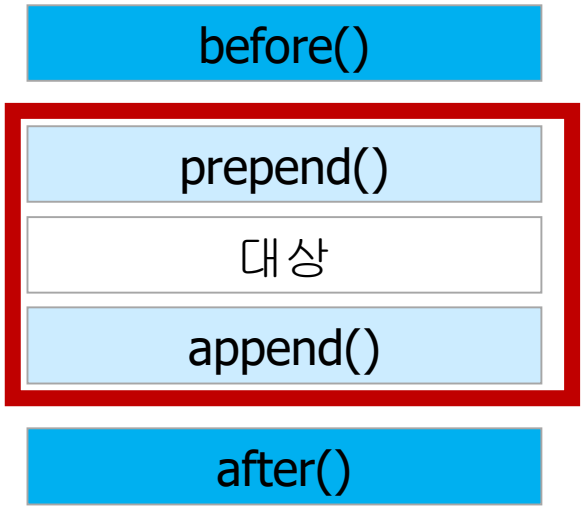
```
$(selector).append(content);
$(selector).append(function(index){});
```

append(*content*) 메서드

- append(*content*) 메서드
  - 인자로 지정된 content를 jQuery 개체 내부에 덧붙이는 메서드

```
예)  $("b.link").append("(클릭)");
        <b class="link">굵은 글자임</b>
     => <b class="link">굵은 글자임 (클릭)</b>
```

  - B 태그 내부 뒤쪽에 덧붙이는 것이기에 "(클릭)"이라는 단어도 굵게 표현됨



```
<script>
    $(document).ready(function () {
        var h1 = '<h1>제목1</h1>';
        var h2 = '<h2>제목2</h2>';

        $('#p1').append(h1);
        $('#p2').append(h1, h2);

        var arr = [
            { name: '홍길동', id: 'hong' },
            { name: '김길동', id: 'kim' },
            { name: '이길동', id: 'lee' }
        ];

        $('div').append(function (index) {
            var item = arr[index];
            var str = item.name + '- <b>' + item.id + '</b>';

            return str;
        });
    });//ready
</script>
<body>
    <p id="p1">p 태그입니다1.</p>
    <p id="p2">p 태그입니다2.</p>
    <div></div>    <div></div>    <div></div>  </body>
```

```
▼<p id="p1">
    "p 태그입니다1."
    <h1>제목1</h1>
  </p>
```

```
▼<p id="p2">
    "p 태그입니다2."
    <h1>제목1</h1>
    <h2>제목2</h2>
  </p>
```

p 태그입니다1.

제목1

p 태그입니다2.

제목1

제목2

홍길동- **hong**
김길동- **kim**
이길동- **lee**

append() 메서드

jQuery(b태그 뒤에 추가(내부))는 가볍고 빠르며, 간결한 오픈소스[1] 스크립트 라이브러리입니다. HTML 문서 트래버스[2], 이벤트 처리, 애니메이션, Ajax[3] 상호 작용 등

- 주석 표시

```
$(".annotation").each(function(i) {
  var idx = i + 1;
  $(this).append("<sup>" + idx + "</sup>");
});
```

  - .annotation라는 css 클래스가 지정된 html 요소들에 대해서 ¹, ²와 같은 주석 표현 태그를 동적으로 덧붙여주는 코드

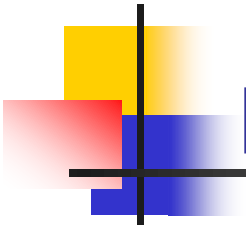
appendTo(*selector*) 메서드

- appendTo(*selector*) 메서드
  - append와는 덧붙이기의 대상이 반대
  - 선택자에 의해 일치된 대상 모두에게 덧붙여짐
-
```
$("#linkText").appendTo("a.link");
```
  - #linkText 라는 아이디를 갖는 요소를 얻어서, 그를 link 라는 css 클래스를 갖는 "모든" 하이퍼링크 뒤에 덧붙이게 됨
  - 원래의 #linkText 요소는 사라짐
  - 원본 개체는 복사되면서 제거됨

appendTo(*selector*) 메서드

- `$("<span>(클릭)</span>").appendTo("b");`
  - (클릭)라는 태그 문자열을 동적으로 생성한 뒤, 그를 모든 b 태그의 뒤에 덧붙임
- **$() 표현의 기능**
  - jQuery()라는 구문의 단축표현으로, 선택자를 수행하기 위한 구문
  - [1] 인자로 선택자를 지정한 경우에는 일치되는 모든 요소를 찾음
  - [2] 인자로 html 태그를 지정한 경우에는 동적으로 그 요소를 생성함
  - [3] 인자로 특정 DOM 요소를 지정하면, 그 요소의 jQuery 래퍼를 생성함.
  - [4] 인자로 function을 지정하면, 그는 $(document).ready() 메서드와 동일

`$("<span>(클릭)</span>")` - 동적으로 span 요소를 생성함
`$("#linkText")` - #linkText라는 선택자에 일치하는 요소들을 검색



prepend()

- prepend()
  - pre-append 라는 의미
  - append와 동일하지만, 요소를 뒤에 덧붙이는 것이 아니라 **앞에 덧붙임**

```
예)  $("b.link").prepend("(클릭)");
        <b class="link">굵은 글자임</b>
     => <b class="link">(클릭) 굵은 글자임 </b>
```

- prependTo() 메서드

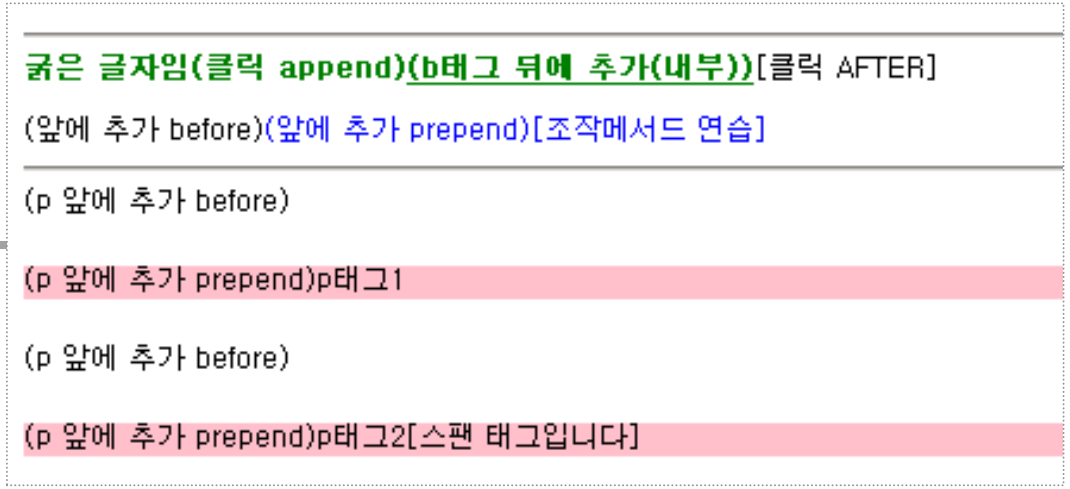
예제

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

        // 함수를 2초마다 실행
        setInterval(function () {
            // 첫 번째 이미지를 마지막으로 보낸다
            $('img').first().appendTo('div');
        }, 2000);
    });
</script>

<body>
    <div>
      <img src="images/dog.jpg" />
      <img src="images/magnum1.jpg" />
      <img src="images/snow.jpg" />
    </div>
```

예제




```
<style type="text/css">
    p{background:pink}
    b{color:green}
</style>
<script>
        $("b.link").append("(클릭 append)");
        $("b.link").after("[클릭 AFTER]");
        $("#sp2").prepend("(앞에 추가 prepend)");
        $("#sp2").before("(앞에 추가 before)");

        $("#linkText").appendTo("p.link");
        $("<span>(b태그 뒤에 추가(내부))</span>").appendTo("b");

        $("p").before("<span>(p 앞에 추가 before)</span>");
        $("p").prepend("<span>(p 앞에 추가 prepend)</span>");
</script>
<body>
    <b class="link">굵은 글자임</b> <br />
    <span id="linkText">[스팬 태그입니다]</span><br />
    <span id="sp2" style="color:Blue">[조작메서드 연습]</span><br />
    <p>p태그1</p>
    <p class="link">p태그2</p>
```

요소 외부에 추가하는 메서드

- 추가(내부) 관련 메서드와 유사하긴 하지만, 요소 내부에 추가를 하는 것이 아니라, 요소 외부에 추가하는 메서드 그룹

추가(요소 외부에) 관련 메서드	
after(*content*)	일치된 요소 뒤에 *content*를 삽입함 요소 내부가 아닌 **외부에** 삽입된다는 것을 제외하면 append와 동일
before(*content*)	일치된 요소 앞에 *content*를 삽입 요소 내부가 아닌 외부에 삽입된다는 것을 제외하면 prepend와 동일
insertAfter(*selector*)	일치된 요소를 *selector*에 의해 일치된 모든 요소들 뒤쪽에 삽입 요소 내부가 아닌 외부에 삽입된다는 것을 제외하면 appendTo와 동일
insertBefore(*selector*)	insertAfter(*selector*)와 유사하나, 요소 앞쪽에 삽입 요소 내부가 아닌 외부에 삽입된다는 것을 제외하면 prependTo와 동일

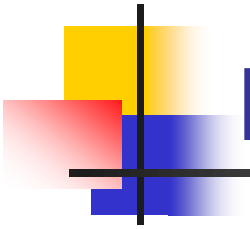
after(*content*)

- after() 메서드
  - append 메서드와 눈에 보이는 결과는 유사
  - 다만, 태그 내부에 덧붙이는 것이 아니라, 태그 바깥쪽에 덧붙인다는 차이

```
예)
$("a").after("<span>(클릭)</span>");
<a href="http://herb.com">허브사이트</a>
=> <a href="http://herb.com">허브사이트</a><span>(클릭)</span>
```

- append 구문 사용시

```
$("a").append("<span>(클릭)</span>");
=> <a href="http://herb.com">허브 사이트<span>(클릭)</span></a>
```



before(*content*)

- before 메서드 - prepend 메서드와 유사
- insertAfter() - appendTo() 메서드와 동일하게 동작,
- insertBefore() - prependTo()와 동일하게 동작
  - 자식 요소로서 태그 내부에 삽입되느냐, 형제 요소로서 외부에 삽입되느냐의 차이만 있음

[3] 삭제용 메서드와 복사용 메서드

- 삭제용 메서드와 복사용 메서드

삭제 메서드	
empty()	모든 일치된 요소들의 **자식 노드들을 제거**
remove()	모든 일치된 요소들을 DOM에서 제거
복사 메서드	
clone()	일치된 요소를 복사하고, 그를 선택함
clone(bool)	이벤트 처리기를 포함하여 DOM 요소를 복사하고 그를 선택함

예제

remove(), empty() 메서드 연습

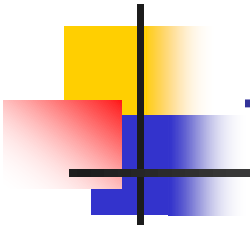
제목2

```
<script>
        $(document).ready(function () {
            $('h2').first().remove();
            $('#div1').empty();
        });
</script>

<body>
   <h1>remove(), empty() 메서드 연습</h1>
   <div>
        <h2>제목1</h2>
        <h2>제목2</h2>
    </div>
    <div id="div1">
        <h3>제목3</h3>
        <h3>제목4</h3>
    </div>
</body>
```

• 결과

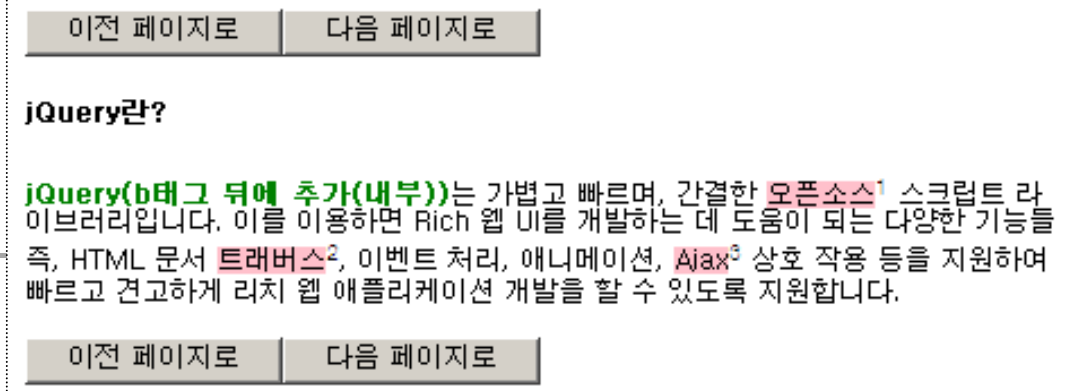
```
<h1>remove(), empty() 메서드 연습</h1>
<div>
                <h2>제목2</h2>
</div>
<div id="div1"></div>
```



복사용 메서드

- 복사 관련 메서드 clone()
  - 인자가 없는 기본 clone() 메서드 – 요소 자체만 복사될 뿐, 해당 요소에 달려있는 이벤트 처리기들은 복사가 되지 않음
- clone(true)
  - 인자로서 true를 지정하면, 단순히 요소만을 복사하는 것이 아니라, 그 요소에 달려있는 이벤트 처리기(예, click, mouseover)등도 복사가 됨

예제




```
<script>
$(document).ready(function() {
    $(".annotation").each(function(i) {
        var idx = i + 1;
        $(this).after("<sup>" + idx + "</sup>");
    });
//    $(".annotation").before("<sup>" + 1 + "</sup>");
//    $(".annotation").prepend("<sup>" + 1 + "</sup>");

    $("button").click(function() {
        alert($(this).text());
        $(this).remove();
        $("#div1").empty();
    });

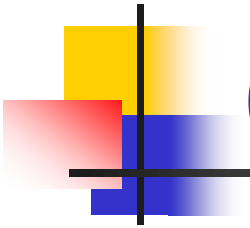
    var $prev = $("#prevBtn").clone();
    var $next = $("#nextBtn").clone(true);

    $("div#bottomDiv").prepend($prev);
    $("div#bottomDiv").append($next);
});
</script>
```

예제

```
<style type="text/css">
    * { font-size:12px; font-family:돋움; }
    .annotation { background: pink; }
    b{color:green}
</style>
<body>
  <button id="prevBtn">이전 페이지로</button>
  <button id="nextBtn">다음 페이지로</button>
  <h3>jQuery란?</h3>
  <div>
    <b>jQuery</b>는 가볍고 빠르며, 간결한 <span class="annotation">오픈소스</span>
    스크립트 라이브러리입니다. 이를 이용하면 Rich 웹 UI를 개발하는 데 도움이 되는 다양한 기능들 즉,
    HTML 문서  <span class="annotation">트래버스</span>, 이벤트 처리, 애니메이션,
    <span class="annotation">Ajax</span>
    상호 작용 등을 지원하여 빠르고 견고하게 리치 웹 애플리케이션 개발을 할 수 있도록 지원합니다.
  </div>
  <br />
  <div id="bottomDiv"></div>

<hr />
div태그 : <div id="div1"><b>empty 연습</b></div>
</body>
```



예제

- <button> 요소들에 대해서 click 이벤트 처리기를 지정, click 처리기에서는 그 버튼을 DOM에서 제거하도록 remove() 메서드를 사용
- 이전 페이지 버튼과 다음 페이지 버튼을 복사해서(clone), 하단의 div 영역에 append
- 이전 페이지 버튼은 clone() 메서드로 복사
  - 클릭 이벤트 처리기가 무시됨
- 다음 페이지 버튼은 clone(true) 메서드로 복사
  - click 이벤트 처리기까지 포함된 상태로 복사가 됨
  - 즉, 복사된 [이전 페이지] 버튼은 클릭해도 아무런 반응이 없지만, [다음 페이지] 버튼은 메시지박스가 뜬다

실습

Move and Copy

Source elements


Target elements



Copy elements

◉ appendTo ○ prepend

Execute | Copy | Empty | Remove

실습

1. Execute 버튼 클릭
[1] appendTo
- 마지막 p태그에 source elements의 flower를 잘라내서 붙인다(내부 뒤쪽에)
- 첫번째 p태그에 source elements의 car를 잘라내서 붙인다(내부 뒤쪽에)
[2] prepend
-마지막 p태그에 flower 이미지 태그를 만들어서 내부 앞쪽에 추가한다
-첫번째 p태그에 car 이미지 태그를 만들어서 내부 앞쪽에 추가한다

2. Copy 버튼 클릭
Target Element의 첫번째 이미지를 복사해서
Copy Element의 p태그 내부 뒤쪽에 추가한다

3. Empty 버튼 클릭
Copy Element의 p태그의 내부요소를 제거한다

4. Remove 버튼 클릭
Copy Element의 p태그의 마지막 이미지를 제거한다

Move and Copy

Source elements

Target elements



Copy elements

appendTo prepend

Execute Copy Empty Remove

실습

Move and Copy

Source elements


Target elements





Copy elements

○ appendTo ◉ prepend

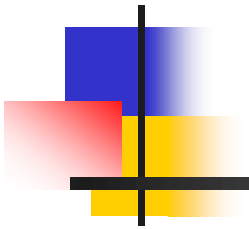
Execute | Copy | Empty | Remove

```
<script type="text/javascript">

</script>

<style type="text/css">
        img {  width: 108px;}
        fieldset {    width: 360px;}
</style>
</head>
<body>
        <h3>Move and Copy</h3>
        <form action="a.jsp" onsubmit="return false;">
                <fieldset id="source">
                        <legend>Source elements</legend>
                        <img id="flower" src="../images/flower.png" />
                        <img id="car" src="../images/car.png" />
                </fieldset>
                <br />
                <fieldset id="targets">
                        <legend>Target elements</legend>
                        <p><img src="../images/dragonfly.png" /></p>
                        <p><img src="../images/dragonfly.png" /></p>
                        <p><img src="../images/dragonfly.png" /></p>
                </fieldset>
                <fieldset id="copy1">
                        <legend>Copy elements</legend>
                        <p></p>
                </fieldset>
```

```
        <div>
            <div>
                <input type="radio" name="command" value="appendTo"  checked="checked" />appendTo
                <input type="radio" name="command" value="prepend" /> prepend
            </div>
            <button type="button" id="btn1">Execute</button> 
            <button type="button" id="btn2">Copy</button> 
            <button type="button" id="btn3">Empty</button> 
            <button type="button" id="btn4">Remove</button>
        </div>
    </form>
</body>
```



CSS 관련 메서드

CSS 관련 메서드

css(*name*)	매치되는 첫번째 요소의 스타일 속성을 반환한다. 예 : var color = $(this).css("color");
css(*name*, *value*)	매치된 모든 요소에 대해 단일 스타일 속성을 설정한다 예 : $(this).css('color', 'yellow');
css(*properties*)	매치된 모든 요소들의 스타일 속성에 키/값 객체를 설정한다. 예 : $(this).css({ 'color':'yellow','font-weight':'bold'});

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

CSS Class 관련 메서드

addClass(*class*)	매치된 요소들의 각 집합에 지정된 CSS 클래스를 추가한다.
addClass(fn)	함수를 매개변수로 입력 선택자로 선택한 문서 객체에 차례대로 속성을 지정할 수 있다
hasClass(*class*)	지정된 클래스가 매치된 요소 집합 중 최소 한 군데 이상 적용되어 있다면 true를 반환한다.
removeClass(*class*)	매치된 요소들의 각 집합에서 지정된 CSS 클래스 혹은 모든 클래스를 제거한다.
toggleClass(*class*)	지정된 클래스가 적용되지 않았다면 적용하고, 이미 적용되어 있다면 제거한다.

CSS 관련 메서드

- css(properties)
  - 여러 CSS 속성을 한번에 설정할 수 있는 오버로드 메서드

```
$("span").css({ 'background': 'pink', 'color': 'blue', "font-weight": "bold" });
```

- toggleClass()
  - CSS Class를 편하게 추가했다 제거했다 할 수 있게 하는 메서드

```
$("button:eq(0)").click(function() {
        $("#img1").toggleClass("summer");
});
```

예제

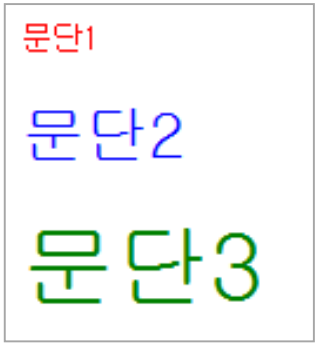
$("#img1").toggleClass("summer");
- 첫번째 클릭 시에는 addClass("summer")를 하고, 두번째 클릭시에는 removeClass("summer")를 반복해서 수행

```
<script type="text/javascript">
        $(function() {
            $("span").css({ 'background': 'pink', 'color': 'blue', "font-weight": "bold" });
            $("#img1")
              .addClass("winter")
              .css("width", "200");

            $("button:eq(0)").click(function() {
                $("#img1").toggleClass("summer");
            });
            $("button:eq(1)").click(function() {
                $("#img1").removeClass("winter");
            });
        });
    </script>
  <style>
      .winter { border:7px solid silver; }
      .summer { border:7px solid lightblue; }
  </style></head>
<body>
  <span>CSS 관련 메서드</span> <br /><br />
  <div id="Mt">
    <img id="img1" src="images/magnum2.jpg" />
  </div>
  <button>토글</button>
  <button>removeClass</button>
```

```
<head>
    <style type="text/css">
            .p1{color:red;font-size:10pt;}
            .p2{color:blue;font-size:20pt;}
            .p3{color:green;font-size:30pt;}
    </style>
  <script src="../../jquery/jquery-1.11.3.js" type="text/javascript"></script>

  <script>
     $(document).ready(function () {
        $('p').addClass(function (index) {
           return 'p' + (index+1);
        });
     });
  </script>
</head>
<body>
  <p>문단1</p>
  <p>문단2</p>
  <p>문단3</p>
</body>
```



Attribute 관련 메서드

attr(*name*)	매치된 첫 번째 요소의 특정 어트리뷰트에 접근하여 값을 가져온다. 만일, 지정된 어트리뷰트 명이 존재하지 않는다면 undefined가 반환된다. 어트리뷰트에는 title, alt, src, href, width, style와 같은 것들이 해당된다.
attr(*properties*)	모든 매치되는 요소들의 어트리뷰트를 키/값 개체로 설정한다.
attr(*key*, *value*)	모든 매치되는 요소들의 단일 어트리뷰트의 값을 지정한다.
attr(*key*, *fn*)	모든 매치되는 요소들의 단일 어트리뷰트에 대해 계산된 값을 지정한다.
removeAttr(*name*)	매치된 요소 각각으로부터 해당 어트리뷰트를 제거한다

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

Attribute 관련 메서드

- attr(*key*, *value*)

```
$("#photos img").attr("height", "100");
```

- attr(*properties*)

```
$("#photos img").attr({
                        border: "1px",
                        height : "100"
                    })
```

Attribute 관련 메서드

- 두 번째 인자로 함수를 지정하여 그 함수의 계산된 반환 값으로 어트리뷰트의 값을 지정할 수 있는 기능

```
예)
$("img").attr("title", function(i) {
                    return (i + 1) + "번째 이미지입니다";
 });
```

  - 모든 <img> 들을 가져와서 그들의 title 어트리뷰트의 값을 동적인 문자열로 설정하는 예
  - 두 번째 인자로 function이 사용
  - 그 함수의 반환 값이 결과적으로 img의 title 어트리뷰트 값이 됨
  - function(i)의 인자인 i 는 0부터 시작하는 요소의 인덱스 값
    - “1번째 이미지입니다”, “2번째 이미지입니다”와 같은 title을 갖게 됨

동일 ⟺

```
$("img").each(function(i) {
        $(this).attr("title", (i + 1) + "번째 이미지입니다");
 });
```

예제 1


```
<script type="text/javascript">
        $(document).ready(function() {
            alert($('a').attr('href'));     //[1] get
            $('img:first').mouseover(function() {
                $(this).attr("src", "images/magnum2.jpg");    //[2] set
            });
            $('img:first').mouseout(function() {
                $(this).attr("src", "images/magnum1.jpg");
            });
            $('#btnRemove').click(function() {
                $('img:eq(1)').removeAttr("src"); // src 속성 삭제
            });
        });
    </script>
    <style type="text/css">
        img{width:200px;}
    </style>
</head>
<body>
<a href="http://www.naver.com/">네이버</a><br /><br />

<img src="images/magnum1.jpg" />
<img src="images/dog.jpg" alt="강아지" />
<input id="btnRemove" type="button" value="src속성 제거" />
```

예제2


```
<script type="text/javascript">
        $(function() {
            $("#photos img")
              .attr({
                  border: "3px",
                  height : "100"
              })
              .attr("title", function(i) {
                   return (i + 1) + "번째 이미지입니다";
              });
        });
   </script>
</head>
<body>

  <div id="photos">
      <img src="images/dog.jpg" />
      <img src="images/magnum1.jpg" />
      <img src="images/snow.jpg" />
      <img src="images/wood.JPG" />
      <img src="images/magnum2.jpg" />
  </div>
```

```
$('#photos img').attr('width', function(idx){
        return (idx+1)*100;
});
```

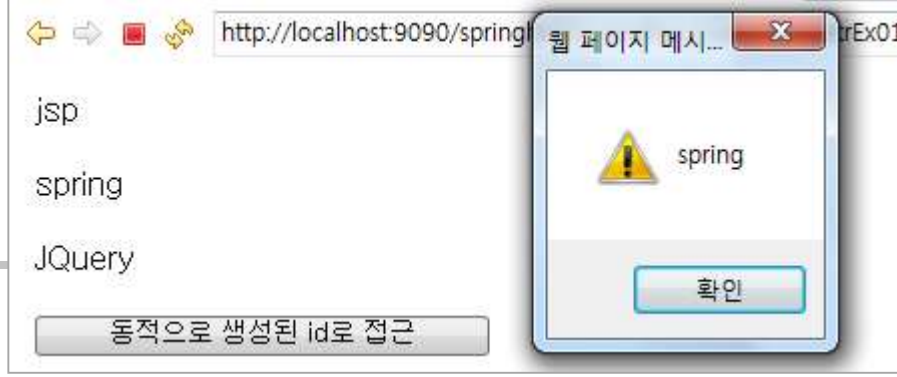
```
$('#photos img').attr({
      width: function(idx){
           return (idx+1)*100;
      },
      height:100
});
```

```
<script type="text/javascript">
	$(document).ready(function () {
		$('#product img').hover(function () {
			$("#showImage").show(); // 이미지 보여줄 레이어 보이기
			var imgSrc = "";
			imgSrc = $(this).attr("src");   //images/herb1.JPG
			imgSrc = imgSrc.substr(imgSrc.lastIndexOf("/") + 1); // 순수 파일명만 얻기 herb1.JPG

			$("#showImage img").attr('src', '../images/big/' + imgSrc);
		},
		function () {// 마우스 오버시 레이어 숨기기
			$("#showImage").hide(); // 레이어 숨기기
		});
	});
</script>
<style type="text/css">
	#showImage{ display:none;border:1px solid gray;width:300px;height:300px;
</style>
</head>
<body>
<div id="product">
	<img src="../images/herb1.JPG" />
	<img src="../images/herb2.JPG" />

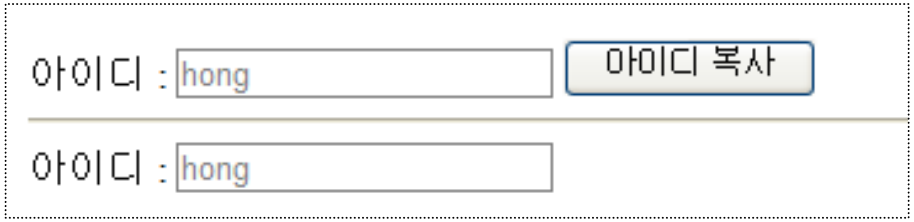
	<div id="showImage">
	  <img  />
	</div>
</div>
</body>
```

실습

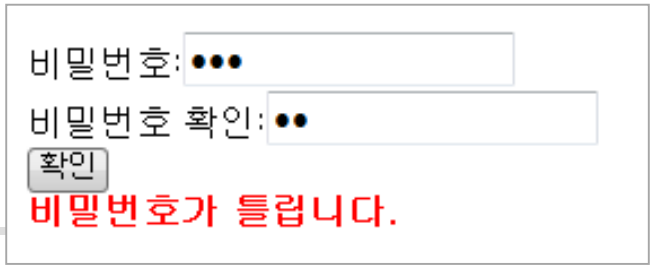


- 1. p 태그의 id 속성에 p_0, p_1, p_2 의 값을 동적으로 설정하기
  - 버튼 클릭시 id가 p_1 에 해당하는 text값을 메세지박스로 띄우기

- 2. 모든 텍스트박스의 테두리를 회색 실선으로, 글자색은 회색으로 설정(css(object) 이용)
  - 아이디 복사 버튼 클릭시 첫번째 텍스트박스의 값을 두번째 텍스트박스로 복사 (val() 이용)



실습



- 3. 확인버튼 클릭시 비밀번호 일치여부를 확인하여 "비밀번호가 틀립니다.", "비밀번호가 맞습니다 " 화면 출력하기
  - val(), html() 이용