# 자바스크립트 & 제이쿼리 (jpub) - 7장 jQuery

## 요소 탐색하기

- 계층 탐색

+ ancestor descendant

+ p>c: 다른 요소의 하위 요소들

+ p+n: 직접적인 자식 요소들(모든 자식요소 하고싶다면 p>\*)

+ p~s: 이전 요소의 모든 이웃 요소

- 기본 필터

+ :not(selector): 셀렉터에 부합하는 요소를 제외한 나머지 요소들(div:not('#summary'))

+ :first: 선택된 요소 중 첫 번째 요소

+ :last, even, odd

+ :eq(index): 선택된 요소 중 매개변수로 지정된 인덱스 번호를 가진 요소

+ :gt(index), :lt(index)

+ :header, :animated, :focus

- 콘텐츠 필터

+ :contains('text'): 매개변수로 지정된 텍스트를 가지고 있는 요소들

+ :empty: 자식 요소가 없는 모든 요소들

+ :parent: 위의 반대

+ :has(selector): 선택된 요소 중 매개변수에 지정된 셀렉터에 부합하는 요소를 최소한 하나 이상 가지고 있는 모든 요소 (ex. div:has(p) p요소를 가진 모든 div요소)

- 가시성 필터

+ :hidden: 화면에서 숨겨진 모든 요소들

+ :visible: 페이지의 레이아웃에서 공간을 차지하고 있는 모든 요소들.

- 자식 요소 필터

+ :nth-child(expr): 1부터 시작하는 해당 순번의 요소(ul li:nth-child(2)는 두번째 요소를 의미)

+ :first-child, :last-child, :only-child

- 특성 필터

+ [attribute]: 지정된 특성 갖고있는 요소들

+ [attribute='value']

+ [attribute!='value']

+ [attribute^='value']: 특성값이 지정된 값으로 시작하는 요소들

+ [attribute$='value']: 특성값이 지정된 값으로 끝나는 요소들

+ [attribute\*='value']: 특성값이 지정된 값으로 포함되는 요소들

+ `[attribute][attribute2]`: 지정된 특성 중 하나를 가진 모든 요소들

- 폼

+ :input, :text, :password, :radio...

+ :image, :button

+ :selected: 드롭다운 리스트에서 선택된 모든 요소들

+ :enabled, :disabled, :checked(체크된 요소들을 리턴)

## 선택된 요소에 필요한 작업 수행

- 콘텐츠 필터

+ 가져오기/수정하기

+ .html(): 첫 번째 요소들과 그 하위 노드들의 HTML코드를 얻음.

+ .text(): 객체집합 내 모든 요소의 텍스트와 그 하위 요소들의 텍스트 리턴.(input요소나 textarea요소에서 내용 가져오려면 .val()메서드 사용.)

+ .replaceWith(): 일치하는 모든 요소에 새로운 동일한 콘텐츠 추가하고 교체된 요소들을 모두 리턴

+ .remove(): 일치하는 모든 요소들을 제거

+ 요소

+ .before(): 선택한 요소 전에

+ .after(): 선택한 요소 다음에

+ prepend(): 선택한 요소의 여는 태그 다음에

\* a.prepend(b)는 a에 b를 추가한다 / a.prependTo(b)는 b에 a를 추가한다.

+ append(): 선택한 요소의 닫는 태그 다음에

+ remove

+ clone

+ unwrap

+ detach

+ empty

+ add: 매개변수로 지정된 셀렉터에 의해 선택된 요소들을 기존 객체집합에 추가

+ 특성

\* attr

- 가져오기: $('li#one').attr('id');

- 수정하기: $('li#one').attr('id', 'hot');

\* removeAttr

\* addClass

\* removeClass

\* css

- 가져오기: bg = $('li').class('background-color');

- 증감하기: $('li').class('padding', '+=20');

- 여러 속성: $('li').css({'color': '#272727', 'font-family': 'Courier'});

+ 폼 값

\* val: input, select, textarea요소에 주로 사용.

\* isNumeric

- 요소 탐색

+ 일반

\* find: 현재 객체집합에서 셀렉터와 일치하는 요소들 리턴

\* closest: 셀렉터와 일치하는 가장 근접한 상위요소(직계부모~최상위요소 모두)를 리턴

\* parent: 현재 객체집합의 직계부모요소 리턴

\* parents: 모든 부모요소

\* children: 모든 자식 요소

\* siblings: 모든 이웃 요소

\* next: 다음 이웃 요소

\* nextAll

\* prev

\* prevAll

+ 필터/테스트

\* filter: 두 번째 셀렉터를 이용하여 jQuery객체집합을 필터링

\* not

\* has

\* is: 폼요소가 선택 혹은 체크되어있는지 확인

\* :contains()

+ 객체집합 내 순서 평가

\* eq: 인덱스 번호에 해당하는 요소를 리턴

\* :lt

\* :gt

- 크기/위치

+ 크기

+ height: 영역의 크기

+ with innerHeight(영역의 높이에 안쪽 여백 더한 값) / innerWidth / outerHeight/ outerWidth

+ $(document).height()

+ $(document).width()

+ $(document).height()

+ $(window).height()

+ $(window).width()

+ 위치

\* offset: document객체의 좌측 상단 모서리에서부터 요소까지의 좌표를 가져오거나 지정(offset().top 이런식으로 쓴다)

\* position: 상위 요소 중 기본 흐름 값을 가진 요소로부터 해당 요소까지의 좌표를 가져오거나 지정

\* scrollLeft

\* scrollTop

- 효과/애니메이션

+ 기본

\* show: 선택된 아이템을 보이게

\* hide: 숨긴다

\* toggle

+ 흐림 효과

\* fadeIn

\* fadeOut

\* fadeTo: 선택된 요소의 불투명도 조절

\* fadeToggle

+ 슬라이딩 효과

\* slideDown

\* slideUp

\* slideToggle

+ 기타

\* delay: 큐 내의 다음 아이템의 실행을 잠시 지연

\* stop: 현재 실행중인 애니메이션 중단

\* animate: 새로운 임의의 애니메이션 생성

- 이벤트

+ 문서/파일

\* ready

\* laod

+ 사용자 상호작용

\* on

\* (.click(), .hover(), submit()같은 메서드들도 보게 될것인데, 이벤트 처리를 위한 on()메서드가 나타나며 사라짐)

### 효과 예제

```javascript

$('h2').hide().slideDown();

var $li = $('li');

$li.hide().each(function(index) { //일단 숨기고, 아이템들이 하나씩 나타나기.

$(this).delay(700\*index).fadeIn(700);

});

$li.on('click', function(){

$(this).fadeOut(700);

});

```

### CSS속성에 애니메이션 적용

- height, width, font-size처럼 값이 숫자로 표현될 수 있는 속성이라면 어떠한 CSS속성이라도 애니메이션에 사용가능.

```javascript

$(this).animate({

opacity: 0.0,

paddingLeft: '+=80'

}, 500, function() {

$(this).remove(); //애니메이션이 완료되면 콜백함수가 아이템을 제거

});

```

## tips

### 정보

- 획득

- jquery객체집합에 둘 이상 요소 저장되어 있으면 ($('li').html();) 첫번째 요소의 정보만 얻어옴.

- 다른 요소 참조하려면 탐색메서드나 필터 메서드, 혹은 셀렉터 이용.

- 모든 요소 콘텐츠 가져오려면 .each()메서드 사용.

- 수정

+ 둘이상 요소 저장되어있고 수정하면 모든 요소 수정됨.

### 변수에 캐싱

- 코드가 동일한 객체집합을 한번 이상 사용할 필요가 있다면 객체에 담아두는게 효과적.

- $listItems = $('li'); 와 같이 변수에 jQuery객체를 저장할 때 이렇게 하면 다른 변수들과 구분하는데 도움된다.

### 체이닝

- jQuery객체집합을 `수정`하기 위한 대부분의 메서드는 체이닝 가능

- 그러나 DOM/브라우저에서 정보를 `조회`하는 메서드들은 체이닝으로 연결 불가.

### 페이지가 준비되었는지 확인

```javascript

// 이 메서드 내부에 작성하면 코드가 페이지의 다른곳이나 다른 파일에 작성된 경우에도 의도대로 동작한다.

$(document).ready(function() {

//bla bla

});

// 위 함수의 약식 표현

$(function() {

//bla bla

});

```

- .load()

+ load이벤트가 발생할 때 호출.

+ .on()메서드로 대체되었다.

+ 페이지와 나머지 모든 리소스(이미지, CSS, 스크립트)가 로드된 이후에 발생

+ 스크립트가 반드시 로드되어야 하는 리소스에 의존적일 때 사용.

\* ex. 스크립트가 이미지의 크기를 알아야 할 때

- .ready()

+ 페이지가 빨리 로드된 것처럼 보이도록 브라우저에 DOM이 로드되자마자 실행.

+ 그러나 최신 브라우저들에서만 지원.(구버전: laod이벤트 되면 시작)

### 컨텐츠를 수정

현재 선택한 요소들의 콘텐츠를 사용하려는 동시에 수정하려면 함수에 매개변수를 전달하면 된다.

```javascript

$('li.hot').html(function(){

return '<em>' + $(this).text() + '</em>';

})

```

### .each()

```javascript

$(function() {

$('li').each(function() {

var ids = this.id; //$(this).attr('id')보다 이게 좋다.

$(this).append('<span class="order">' + ids + '</span>');

});

});

```

### this

`$(this)`처럼 this키워드를 이용해 새로운 jQeury 객체집합을 만드는 경우도 볼 수 있는데,

이렇게 하면 현재 요소에 대해서도 jQuery메서드들을 사용할 수 있다.

### .on()

- 매개변수를 2개 받는다. 첫번째는 대응할 이벤트, 두번째는 함수(기명/익명)

- (\*친것들은 onready처럼 작업을 더 쉽게 만들어주는 추가 메서드 제공.)

- focus, blur, change

- input, keydown keyup, keypress

- click, dblclick, mouseup, mousedown, mouseover, mousemove, mouseout, \*hover

- submit, select, change

- \*ready, load, \*unload

- error, resize, scroll

```javascript

//모바일은 마우스오버가 없으니 클릭도 같이 지정.

$listItems.on('mouseover click', functionn(){

//~~

});

```

### event객체

```javascript

$('li').on('click', function(e) {

eventType = e.type;

});

```

#### 속성

- type: 이벤트 종류

- which: 눌려진 버튼이나 키

- data

- target: 이벤트가 발생한 DOM요소

- pageX, pageY

- timeStamp

- .preventDefault(): 기본 동작을 취소(ex. 폼 데이터 전송)

- .stopPropagation(): 상위 객체로 이벤트가 버블링되는 것을 중단.

### .animate()

```javascript

.animate({

//변경할 스타일들 나열

}, [, speed][, easing][, complete]);

// speed: 진행될 시간의 길이 밀리초단위. low/fast도 가능

// easing: linear(일정속도) / swing(중간은 빨리 처음끝은 느리게)

// complete: 애니메이션이 종료시 호출될 함수(콜백함수)를 지정 위함

$(this).animate({

opacity: 0.0,

paddingLeft: '+=80'

}, 500, function() {

$(this).remove();

});

```

### 오타

303p attribute$='value'

322p $('li').clss('~')

322p 'background-color': '#272727',

**Listing 3. 간단한 문단을 생성하여 붙이기**

|  |
| --- |
| $('<p></p>')  .html('Hey World!')  .css('background', 'yellow')  .appendTo("body"); |

**웹 프런티어와 함께하는 jQuery 기초강좌**

**2nd - jQuery를 이용한 HTML DOM 접근(기본 셀렉터 첫 번째 이야기)**

**원하는 개체를 쉽고 편하게 선택하자 - 셀렉터**

이번 시간에는 jQuery의 가장 강력한 기능인 HTML DOM을 탐색하는 기능에 대해 알아 보도록 하겠습니다.

jQuery의 DOM 탐색은 CSS SELECTER를 사용하고 있어, CSS에 사용한 표현식을 알고 있다면 보다 쉽게 셀렉터를 이해 할 수 있습니다.

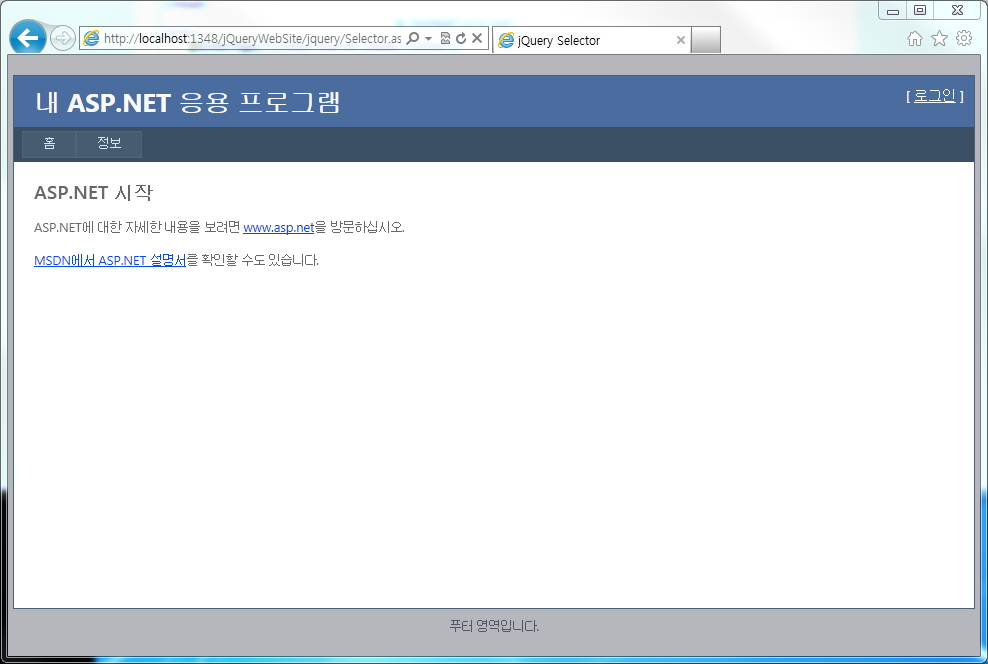
jQuery에서는 원하는 HTML의 DOM 요소를 찾기 위해 $(Selector), jQuery(Selector)와 같은 표현식을 사용합니다.

$는 jQuery의 축약어로 같은 역할을 하며, 다음과 같은 형태로 사용하여 원하는 DOM 요소를 선택 할 수 있습니다.

|  |  |
| --- | --- |
| **셀렉터의 종류** | **셀렉터 표현 방법** |
| **All Selector** | **$("\*")** |
| **ID Selector** | **$("#id")** |
| **Element Selector** | **$("elementName")** |
| **Class Selector** | **$(".className")** |
| **Multiple Selector** | **$("selector1, selector2, selector3, selectorN")** |

$(Selector), jQuery(Selector)를 사용하여 선택한 DOM의 요소는 “document.getElementByID”를 이용한 것과는 다르게 해당 객체를 jQuery 객체로 랩핑해서 반환해 주기 때문에 jQuery에서 지원하는 기능을 쉽게 적용 할 수 있다는 장점이 있습니다.

**All Selector : $(“\*”)**  
HTML DOM을 탐색하여 모든 요소를 배열형식의 jQuery 개체로 반환합니다.



<**script src="http://code.jquery.com/jquery-latest.js"**></**script**>

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

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

//HTML 문서내의 모든 엘리먼트를 찾아 붉은 선으로 표시합니다.

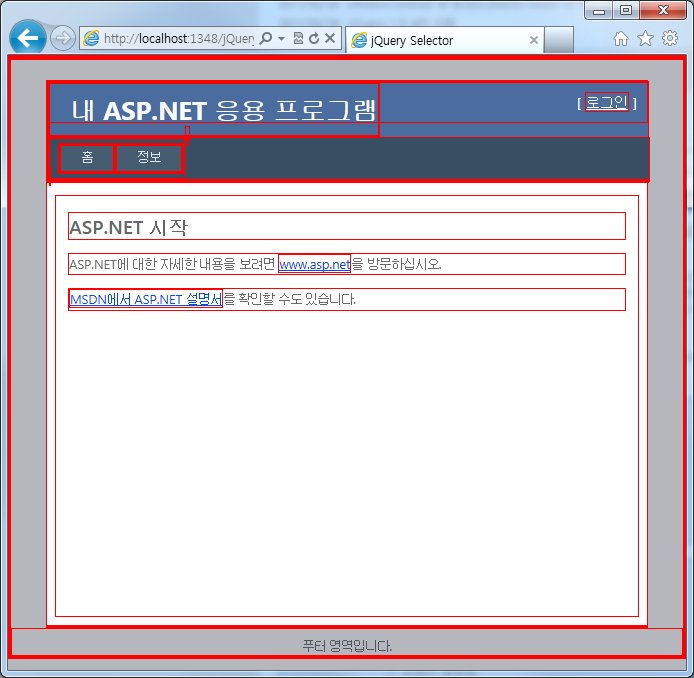
$("\*").css("border", "1px solid #ff0000");

});

</**script**>

jQuery를 페이지에 참조 시키고 $(document).ready() 함수를 통해 모든 객체를 선택한 후 선택된 jQuery 객체를 눈에 잘 보이게 붉은색의 1픽셀 테두리를 설정 하도록 해 보겠습니다.

$(document).ready() 또는 $(function() {})는 페이지의 HTML DOM이 모두 로드가 되면 실행이 되는 함수로 자바스크립트의 onload 메서드의 확장된 역할을 하고 있습니다. jQuery를 사용하실 경우 onload 이벤트의 사용보다 $(document).ready() 또는 $(function() {})를 사용하시길 적극 추천 드리며 관련 내용에 대해서는 마지막 부분에 다시 한번 정리를 하도록 하겠습니다.



화면 2와 같이 jQuery를 통해 선택된 모든 요소에 붉은색 테두리가 설정된 것을 확인 할 수 있습니다. 이러한 기능이 가능한 것은 일반적인 DOM 개체가 아닌 jQuery 개체로 반환이 되었기 때문입니다.

**ID Selector : $(“#ID”)**

문서 안에 있는 여러 엘리먼트중 ID값이 동일한 엘리먼트를 찾아 반환합니다.

동일한 값(동일ID)의 엘리먼트가 여러 개일 경우에는 최상위에 있는 엘리먼트를 선택 반환하며, 한 문서(HTML)에는 한 ID만 존재하는 것이 원칙으로 동일한 값을 통해 접근을 하고 싶을 경우에는 class 또는 attribute의 동일한 값을 설정 하여 사용하길 권장합니다.

<**html** xmlns="http://www.w3.org/1999/xhtml">

<**head**>

<**title**>jQuery Selector</**title**>

<**link** href="../Styles/Site.css" rel="stylesheet" type="text/css" />

<**style**>

**div** { background : #FFF; padding:10px; margin:10px; }

</**style**>

<**script src="http://code.jquery.com/jquery-latest.js"**></**script**>

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

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

//ID가 Content인 요소를 찾아 배경을 "yellow"로 변경합니다.

$("#content").css("background", "yellow");

});

</**script**>

</**head**>

<**body**>

<**h2**>jQuery 시작 Selector</**h2**>

<**p**>jQuery에 대한 자세한 내용을 보려면 jquery.com 을 방문하세요.</**p**>

<**div**>

<**div** id="content">

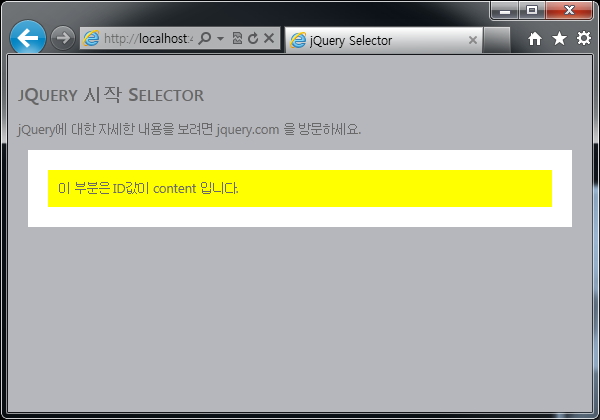
이 부분은 ID값이 content 입니다.

</**div**>

</**div**>

</**body**>

</**html**>



**Element Selector : $(“element”)**

자바스크립트의 getElementByTagName(“tagName”)과 비슷한 역할을 하며 DOM 개체를 구성하는 태그와 동일한 개체를 찾아 다수의 jQuery 개체를 반환합니다.

아래의 소스는 jQuery를 통해 선택된 “DIV” 개체의 테두리는 파란색으로 변경합니다.

<**html** xmlns="http://www.w3.org/1999/xhtml">

<**head**>

<**title**>jQuery Selector</**title**>

<**link** href="../Styles/Site.css" rel="stylesheet" type="text/css" />

<**style**>

**div**,**span** {

width:40px; height:40px; float:left; padding:10px; margin:10px;

background-color:#EEEEEE;

}

</**style**>

<**script src="http://code.jquery.com/jquery-latest.js"**></**script**>

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

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

//div 개체를 찾아 테두리를 "blue"로 변경합니다.

$("div").css("border", "2px solid blue");

});

</**script**>

</**head**>

<**body** style="padding:10px;">

<**h2**>jQuery 시작 Selector</**h2**>

<**p**>jQuery에 대한 자세한 내용을 보려면 jquery.com 을 방문하세요.</**p**>

<**span**>SPAN2</**span**>

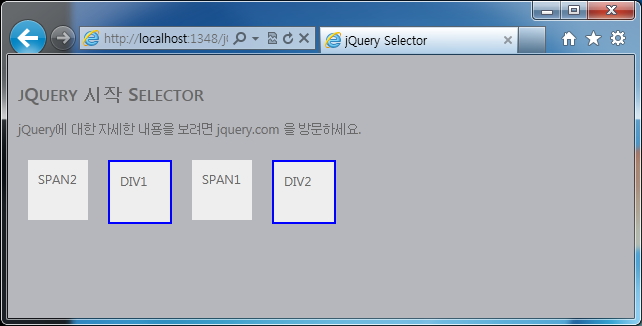
<**div**>DIV1</**div**>

<**span**>SPAN1</**span**>

<**div**>DIV2</**div**>

</**body**>

</**html**>



이번 시간에는 jQuery 가장 기본이 되는 셀렉터의 기능중 가장 많이 사용하는 id, elemnt를 이용한 방법에 대해 알아 보았습니다.

다음 시간에는 요소의 클래스(class)와 id, element, class를 조합하여 개체에 접근 하는 방법에 대해 알아 보기로 하겠습니다.

**웹 프런티어와 함께하는 jQuery 기초강좌**

**4th 이야기 - 셀렉터 Attribute를 활용한 요소 선택**

**개체의 속성값을 통해 요소를 선택하자 - Attribute Selector**

각각의 요소는 속성(attribute)을 가질 수 있습니다. 각 요소의 속성은 미리 정해진 이름이거나, 사용자의 필요에 의해서 만들어질 수 있으며 jQuery의 CSS 셀렉터와 필터의 조합으로 통해 관련된 요소에 접근 할 수 있습니다.  
다음은 jQuery에서 지원하고 있는 속성관련 셀렉터 입니다. (더 많은 종류의 속성관련 셀렉터를 지원하고 있으니 자세한 내용은 jQuery.com에서 확인을 하시기 바랍니다.)

|  |  |
| --- | --- |
| **형식(셀렉터)** | **설명** |
| **$(Selector[attr])** | **attr 속성(attribute)값을 가지는 Selector 요소와 일치** |
| **$(Selector[attr=”value”])** | **attr 속성의 값이 value와 동일한 값인 Selector 요소와 일치** |
| **$(Selector[attr!=”value”])** | **attr 속성의 값이 value와 같지 않은 값인 Selector 요소와 일치** |
| **$(Selector[attr^=”value”])** | **attr 속성의 값이 value 값으로 시작하는 Selector 요소와 일치** |
| **$(Selector[attr$=”value”])** | **attr 속성의 값이 value 값으로 끝나는 Selector 요소와 일치** |
| **$(Selector[attr\*=”value”])** | **attr 속성의 값이 value 값을 포함하는 Selector 요소와 일치** |
| **$(Selector[attr~=”value”])** | **attr 속성의 값이 공백과 함께 value 값을 포함하는 Selector 요소와 일치** |

**Has Attribute Selector : $(Selector[attr])**  
attr이라는 속성(Attribute)값을 가지는 셀렉터의 요소와 일치하는 요소를 반환합니다.  
input, checkbox, src, href등 “<”, “>” 사이에 정의된(사용자가 임의로 정의한 값 포함) 값에 대해 일치하는 항목이 존재 한다면 해당 요소를 반환합니다.

다음은 “a” 태그의 속성 “target”이 존재하는 요소를 선택하는 예제입니다.

<**html** xmlns="http://www.w3.org/1999/xhtml">

<**head**>

<**title**>jQuery Selector</**title**>

<**link** href="../Styles/Site.css" rel="stylesheet" type="text/css" />

<**style**>

**div** { background : #FFF; padding:10px; margin:10px; }

</**style**>

<**script src="http://code.jquery.com/jquery-latest.js"**></**script**>

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

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

$("div > a[target]").css("background", "#CCC");

});

</**script**>

</**head**>

<**body** style="padding:10px;">

<**h2**>jQuery 시작 Selector</**h2**>

<**p**>jQuery에 대한 자세한 내용을 보려면 jquery.com 을 방문하세요.</**p**>

<**div**>

<**a** href="http://www.slqer.com">SQLER.COM (No Target)</**a**><**br** />

<**a** href="http://www.slqer.com" target="\_blank">SQLER.COM (Target)</**a**><**br** />

<**a** href="http://www.hoons.kr">Hoons.NET (No Target)</**a**><**br** />

<**a** href="http://www.hoons.kr" target="\_blank">Hoons.NET (Target)</**a**><**br** />

<**a** href="http://www.hows.kr">HowsNET (No Target)</**a**><**br** />

<**a** href="http://www.hows.kr" target="\_blank">HowsNET (No Target)</**a**><**br** />

</**div**>

</**body**>

</**html**>



**Attribute Equals Selector : $(Selector[attr=”value”])**  
셀렉터의 요소중 attr 과 value의 값이 동일한 요소를 찾아 반환합니다.  
다음 예제는 HTML 문서에 존재하는 링크중에 [http://www.sqler.com](http://www.sqler.com/)인 요소를 찾아 표시 합니다.

<**html** xmlns="http://www.w3.org/1999/xhtml">

<**head**>

<**title**>jQuery Selector</**title**>

<**link** href="../Styles/Site.css" rel="stylesheet" type="text/css" />

<**style**>

**div** { background : #FFF; padding:10px; margin:10px; }

</**style**>

<**script src="http://code.jquery.com/jquery-latest.js"**></**script**>

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

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

$("div > a[href='http://www.sqler.com']").css("background", "#CCC");

});

</**script**>

</**head**>

<**body** style="padding:10px;">

<**h2**>jQuery 시작 Selector</**h2**>

<**p**>jQuery에 대한 자세한 내용을 보려면 jquery.com 을 방문하세요.</**p**>

<**div**>

<**p**>

다음 링크중에 href 값이 "http://www.sqler.com"인 요소를 표시합니다.

</**p**>

<**a** href="http://www.sqler.com">SQLER.COM (No Target)</**a**><**br** />

<**a** href="http://www.sqler.com" target="\_blank">SQLER.COM (Target)</**a**><**br** />

<**a** href="http://www.hoons.kr">Hoons.NET (No Target)</**a**><**br** />

<**a** href="http://www.hoons.kr" target="\_blank">Hoons.NET (Target)</**a**><**br** />

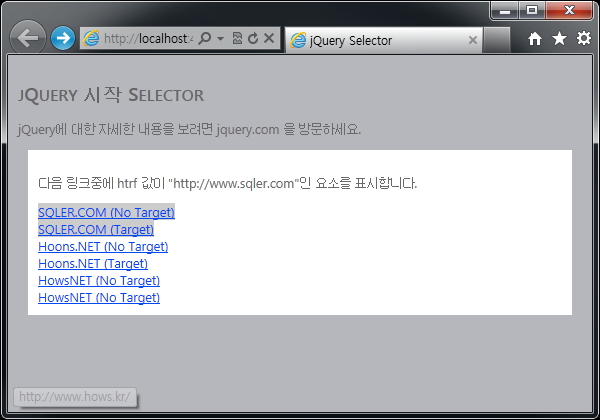
<**a** href="http://www.hows.kr">HowsNET (No Target)</**a**><**br** />

<**a** href="http://www.hows.kr" target="\_blank">HowsNET (No Target)</**a**><**br** />

</**div**>

</**body**>

</**html**>



**Attribute Not Equals Selector : $(Selector[attr!=”value”])**  
바로 앞에서 설명한 셀렉터와는 반대로 이번 셀렉터는 요소중 attr의 값이 value가 아닌 요소를 찾아 반환합니다. 앞의 예제를 활용하여 간단히 알아 보도록 하겠습니다.  
앞에서 설명한 예제 코드에서 다음 부분만 변경을 해 보겠습니다.

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

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

$("div > a[href='http://www.sqler.com']").css("background", "#DDD");

});

</**script**>

[변경전]

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

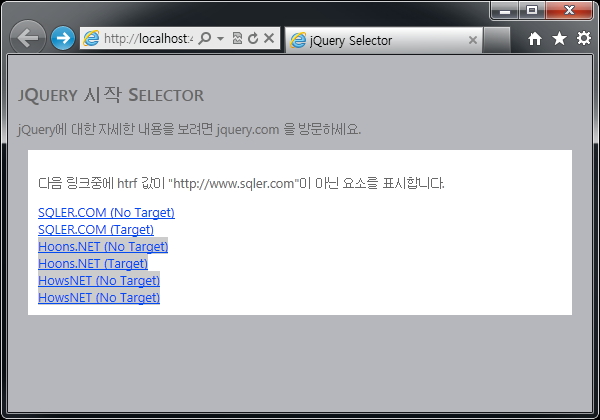
$(document).ready(**function** () {

$("div > a[href!='http://www.sqler.com']").css("background", "#CCC");

});

</**script**>

[변경후]



**Attribute Starts With Selector : $(Selector[attr^=”value”])**  
지정된 속성의 값으로 시작되는 요소와 일치하는 요소를 찾아 반환합니다.  
일정한 패턴으로 정의된 속성이 있을경우 해당 셀렉터를 사용하면 매우 유용합니다. 다음 예제를 통해 간단히 알아 보도록 하겠습니다.

<**html** xmlns="http://www.w3.org/1999/xhtml">

<**head**>

<**title**>jQuery Selector</**title**>

<**link** href="../Styles/Site.css" rel="stylesheet" type="text/css" />

<**style**>

**div** { background : #FFF; padding:10px; margin:10px; }

</**style**>

<**script src="http://code.jquery.com/jquery-latest.js"**></**script**>

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

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

$("div[id^='content-']").css("background", "#CCC");

});

</**script**>

</**head**>

<**body** style="padding:10px;">

<**h2**>jQuery 시작 Selector</**h2**>

<**p**>jQuery에 대한 자세한 내용을 보려면 jquery.com 을 방문하세요.</**p**>

<**div**>

<**p**>

"id"가 "content"로 시작하는 요소를 표시합니다.

</**p**>

<**div** id="content-1">DIV Content-1</**div**>

<**div** id="content-2">DIV Content-2</**div**>

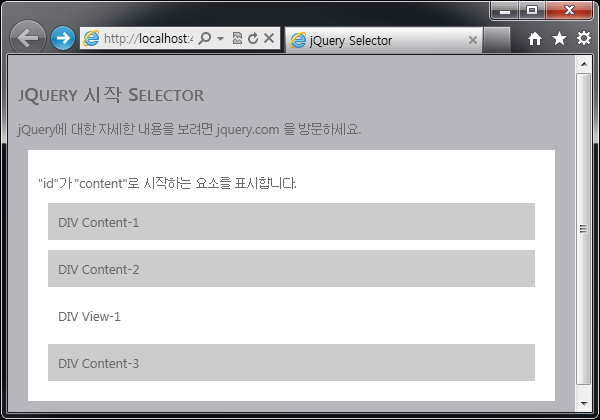
<**div** id="view-1">DIV View-1</**div**>

<**div** id="content-3">DIV Content-3</**div**>

</**div**>

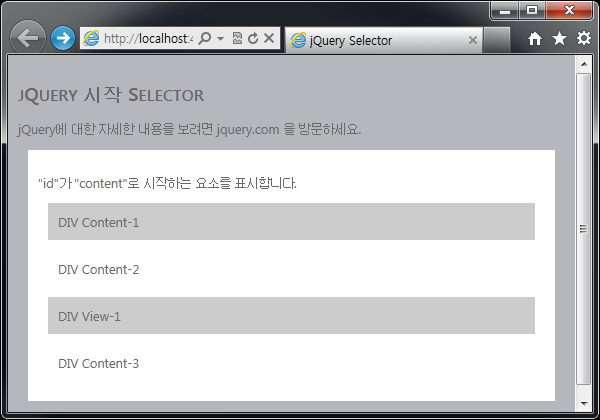
</**body**>

</**html**>



**Attribute Ends With Selector : $(Selector[attr$=”value”])**  
지정된 속성의 값으로 끝나는 요소와 일치하는 요소를 찾아 반환합니다.  
앞에서 설명한 예제 코드에서 다음 부분만 변경을 해 보겠습니다.

$("div[id$='1']").css("background", "#CCC");



**Attribute Contains Selector : $(Selector[attr\*=”value”])**  
지정된 속성의 값이 포함된 요소와 일치하는 요소를 찾아 반환합니다.

<**html** xmlns="http://www.w3.org/1999/xhtml">

<**head**>

<**title**>jQuery Selector</**title**>

<**link** href="../Styles/Site.css" rel="stylesheet" type="text/css" />

<**style**>

**div** { background : #FFF; padding:10px; margin:10px; }

</**style**>

<**script src="http://code.jquery.com/jquery-latest.js"**></**script**>

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

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

$("input[name\*='man']").css("background", "yellow");

});

</**script**>

</**head**>

<**body** style="padding:10px;">

<**h2**>jQuery 시작 Selector</**h2**>

<**p**>jQuery에 대한 자세한 내용을 보려면 jquery.com 을 방문하세요.</**p**>

<**div**>

<**p**>"name"값에 man이 포함된 요소를 표시합니다.</**p**>

<**input** name="man" value="name is man" /><**br** />

<**input** name="superman" value="name is superman" /><**br** />

<**input** name="iron man" value="name is iron man" /><**br** />

<**input** name="bat man" value="name is bat man" /><**br** />

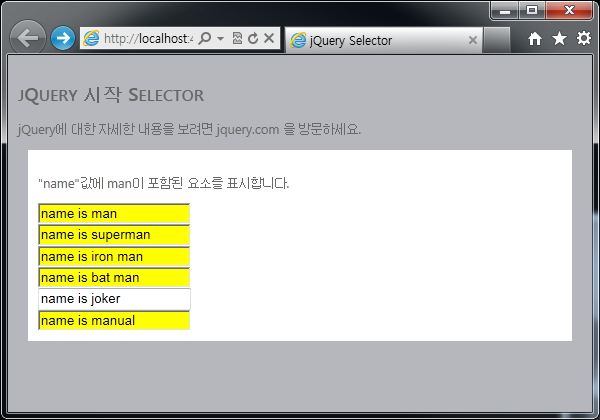
<**input** name="joker" value="name is joker"/><**br** />

<**input** name="manual" value="name is manual"/><**br** />

</**div**>

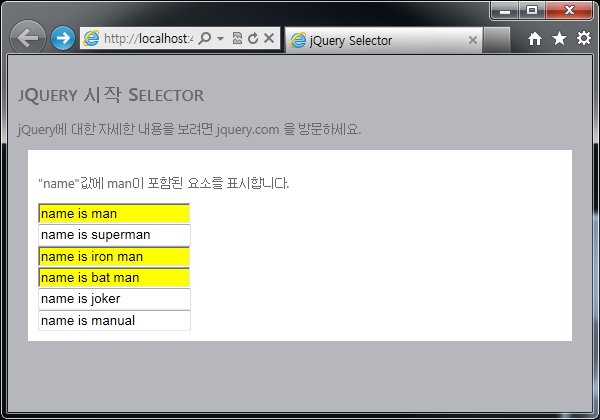
</**body**>

</**html**>



Name 값의 문자열에 “man”과 일치하는 항목이 있으면 무조건 해당 요소를 선택하며, 마지막 항목 “manual”도(man이 포함된 문자열도 일치하는 항목으로 선택) 함께 선택된 모습을 확인 할 수 있습니다.

**Attribute Contains Word Selector : $(Selector[attr~=”value”])**  
지정된 속성의 값이 공백과 함께 일치하는 요소를 찾아 반환합니다.  
$(input[name\*=”man”])의 경우 “man, superman, iron man, manual”… 에 대한 모든 항목에 대해 일치하지만 $(input[name~=”man”])의 경우 “man”과 정확히 일치하는 요소만 반환합니다. 앞의 셀렉터는 문자에 대해서 일치하는 항목을 찾았다면, 두번째의 경우는 단어의 단위로 일치하는 요소를 찾아 반환을 합니다.  
앞에서 설명한 예제 코드에서 $(input[name\*=”man”]) 부분을 $(input[name~=”man”])로 변경 후의 결과를 보면 다음과 같습니다.



앞의 결과와는 전혀 다르게 요소가 선택된 결과를 보이며, “man”이란 단어 단위로 정확하게 일치하는 요소에 대해서만 선택된 모습을 확인 할 수 있습니다. 성값을 이용한 요소의 선택은 기본 셀렉터를 이용하는 것 보다는 느리나, 보다 정확하게 원하는 요소에 접근 할 수 있다는 장점이 있습니다.

속성값을 이용하여 요소를 선택하기 앞서 클래스나 DOM을 이용한 요소의 선택 방법이 있다면 해당 방법을 사용 하길 권해드립니다.

다음 강좌에서는 HTML DOM의 구조를 셀렉터를 통해 접근하는 방법에 대해 알아 보기로 하겠습니다.

**웹 프런티어와 함께하는 jQuery 기초강좌**

**5th - DOM 계층(Hierarchy)을 이용한 요소 접근**

**- 첫 번째 이야기 DOM의 이해**

이번시간에는 jQuery에서 제공하는 HTML DOM의 계층을 이용한 요소의 접근에 대해 알아 보도록 하겠습니다.... 만...

우선 본론으로 들어가기 전에 HTML DOM에 대한 이야기를 먼저 하려고 합니다.

HTML DOM에 대해 정확하게 아시는 것도 중요 하지만, 기초를 다지기 위한 강좌이니 다음 강좌에서 꼭 알아야 하는 내용을 간단하게 정리해 보았습니다.

**DOM이란 ?**

HTML 트리 구조를 통해 클라이언트 영역에서 재조합 기능을 제공하여 사용자와 상호 작용을 하는 구조적인 모델로 HTML 트리 구조 내에서 개별 객체에 접근이 제공되는 방법을 말합니다.

HTML은 트리 구조로 이루어져 있어 자신의 부모, 조상, 자식, 후손에 대해 쉽게 접근이 가능합니다.

**HTML DOM의 구조**

<**html**>

<**head**>

<**title**>DOM Tree</**title**>

</**head**>

<**body**>

<**div**>Hello jQuery</**div**>

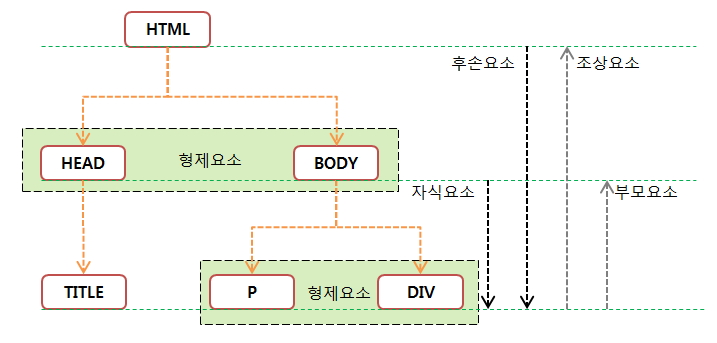
<**p**>안녕하세요!<**em**>승연아빠</**em**>입니다.</**p**>

</**body**>

</**html**>

위와 같은 코드를 트리 형태로 표현 하자면 다음과 같습니다.

그림에서 중요하게 보실 부분은 "자신의 부모, 조상, 자식, 후손, 형제"에 대한 부분입니다.



HTML DOM은 그림 1과 같은 관계를 형성하고 있습니다.

HEAD, BODY는 HTML의 자식 요소이면서 서로 HTML 부모에 묶여있는 형제 요소이며, HEAD는 TITLE의 부모요소가 BODY는 P, DIV의 부모요소가 됩니다. 또한 BODY를 부모로 두고 있는 P, DIV는 형제요소이면서 HTML의 후손 요소가 되며, 반대로 HTML은 P, DIV, TITLE에 대해 조상 요소가 됩니다.

각각 요소에 대한 상하관계와 평행관계를 정확히 이해를 하셔야 다음에 진행될 강좌를 이해 할 수 있습니다.

(디자이너의 피가 흐른다고 생각하고 최대한 열심히 그렸습니다.)

마치 가족의 족보를 옮겨 놓았다고 생각하면 이해가 쉽습니다.

DOM에 대한 설명은 여기서 간단히 마치며, 다음 강좌에서는 jQuery에서 지원하는 DOM 계층에 접근하는 방법에 대해 자세히 알아 보도록 하겠습니다.

**쉬어가는 페이지.....**

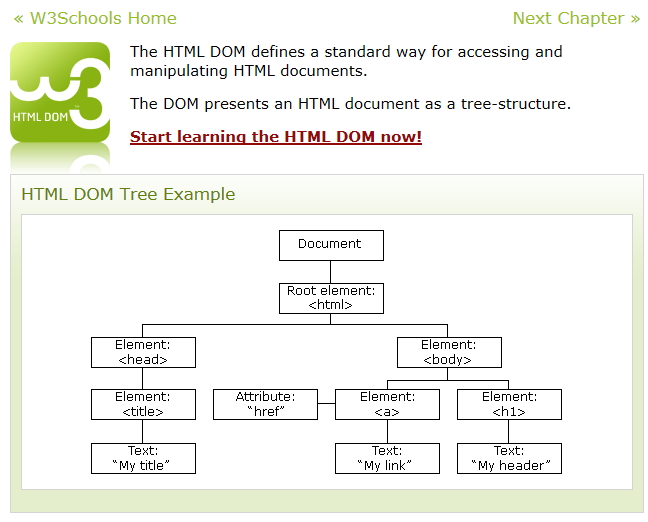
[**Webpages as Graphs**](http://www.aharef.info/)

웹 페이지의 DOM 구조를 이용해 그림을 그려주는 사이트입니다.

**HTML DOM Tutorial**

HTML DOM에 대해 자세히 알고 싶으시다면 이곳을 방문하세요.

[**W3School**](http://www.w3schools.com/htmldom/default.asp)



**웹 프런티어와 함께하는 jQuery 기초강좌**

**6th - DOM 계층(Hierarchy)을 이용한 요소 접근(2)**

이전 시간에 알아본 HTML DOM을 jQuery에서 지원하는 DOM 접근 방법에 대해 알아 보도록 하겠습니다.

**jQuery 계층 접근 셀렉터의 종류**

jQuery에서는 다음과 같이 4개의 계층 접근용 셀렉터를 지원하고 있으며, 각각의 사용법에 대해서는 간단한 예제와 함께 알아 보도록 하겠습니다.

|  |  |
| --- | --- |
| **형식(셀렉터)** | **셀렉터 표현식** |
| Child Selector | $(“parent > child”) |
| Descendant Selector | $(“ancestor descendant”) |
| Next Adjacent Selector | $(“prev + next”) |
| Next Siblings Selector | $(“prev ~ siblings”) |
| 표 1. [jQuery 계층(Hierarchy) 셀렉터의 종류] | |

**Child Selector : $(“Parent > Child”)**

부모(Parent) 요소 바로 아래 자식(Child)인 요소를 반환 합니다.

최근에 출시된 대부분의 브라우저를 지원하고 있으나, IE6를 포함한 이전의 브라우저에서는 지원을 하고 있지 않으니 사용에 각별한 주의가 필요합니다. IE6를 포함한 이전의 브라우저의 지원을 위해서는 3번째 강좌에서 설명한 기본 셀렉터를 사용 하시기 바랍니다. 기본 셀렉터의 경우 모든 브라우저를 지원하고 있으므로 브라우저의 버전이나, 종류에 신경 쓸 필요가 없습니다.

<**html** xmlns="http://www.w3.org/1999/xhtml">

<**head**>

<**title**>jQuery Selector</**title**>

<**link** href="../Styles/Site.css" rel="stylesheet" type="text/css" />

<**style**>

**div** { background-color:#EEEEEE; padding:10px; }

</**style**>

<**script src="http://code.jquery.com/jquery-latest.js"**></**script**>

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

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

$("ul.siteUrl > li").css("border", "1px solid #ff0000");

});

</**script**>

</**head**>

<**body** style="padding:10px;">

<**h2**>jQuery 시작 Selector</**h2**>

<**p**>jQuery에 대한 자세한 내용을 보려면 jquery.com 을 방문하세요.</**p**>

<**div**>

<**ul** class="siteUrl">

<**li**>Sqler : http://www.sqler.com</**li**>

<**li**>jQuery : http://jquery.com</**li**>

<**li**>

<**ul**>

<**li**>Blog : http://www.hows.kr</**li**>

<**li**>Community : http://www.hoons.kr</**li**>

</**ul**>

</**li**>

<**li**>Items</**li**>

<**li**>Items</**li**>

<**li**>Items</**li**>

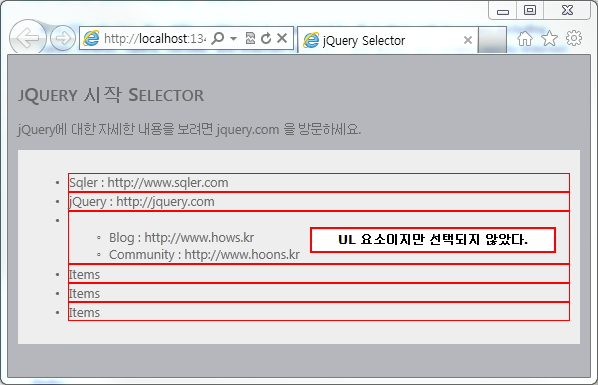
</**ul**>

</**div**>

</**body**>

</**html**>

[부모요소(ul)에서 자식요소(li)과 일치하는 항목 선택]



[부모요소(ul)에서 자식요소(li)과 일치하는 항목 선택된 모습]

$("ul.siteUrl > li").css("border", "1px solid #ff0000"); 코드설명

1. HTML 문서에서 ul 요소 중 “siteUrl”이란 클래스 명을 가진 요소를 찾습니다.

2. 1번 항목에서 찾은 요소의 자식 중에 “li” 항목과 일치하는 요소를 찾습니다.

3. 2번 항목에서 찾은 요소를 jQuery의 “css” 메소드를 이용해 테두리 값을 빨강색으로 변경합니다.

**Descendant Selector : $(“ancestor descendant”)**

조상(Ancestor)과 일치하는 요소에 포함된 모든 후손(Descendant)중에 후손의 요소와 일치하는 모든 항목을 반환합니다. 부모(Parent)와 자식(Child)간의 관계가 조상(Ancestor)과 후손(Descendant)로 변경이 되었을 뿐 Child 셀렉터와 동일한 기능을 제공 합니다.

<**html** xmlns="http://www.w3.org/1999/xhtml">

<**head**>

<**title**>jQuery Selector</**title**>

<**link** href="../Styles/Site.css" rel="stylesheet" type="text/css" />

<**style**>**div** { background-color:#EEEEEE; padding:10px; }</**style**>

<**script src="http://code.jquery.com/jquery-latest.js"**></**script**>

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

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

$("#siteUrl1 > li").css("border", "1px solid #ff0000");

$("#siteUrl2 li").css("border", "1px solid blue");

});

</**script**>

</**head**>

<**body** style="padding:10px;">

<**h2**>jQuery 시작 Selector</**h2**>

<**p**>jQuery에 대한 자세한 내용을 보려면 jquery.com 을 방문하세요.</**p**>

<**div**>

<**ul** id="siteUrl1">

<**li**>Sqler : http://www.sqler.com</**li**>

<**li**>jQuery : http://jquery.com</**li**>

<**li**>

<**ul**>

<**li**>Blog : http://www.hows.kr</**li**>

<**li**>Community : http://www.hoons.kr</**li**>

</**ul**>

</**li**>

<**li**>Items</**li**>

<**li**>Items</**li**>

<**li**>Items</**li**>

</**ul**>

<**ul** id="siteUrl2">

<**li**>Sqler : http://www.sqler.com</**li**>

<**li**>jQuery : http://jquery.com</**li**>

<**li**>

<**ul**>

<**li**>Blog : http://www.hows.kr</**li**>

<**li**>Community : http://www.hoons.kr</**li**>

</**ul**>

</**li**>

<**li**>Items</**li**>

<**li**>Items</**li**>

<**li**>Items</**li**>

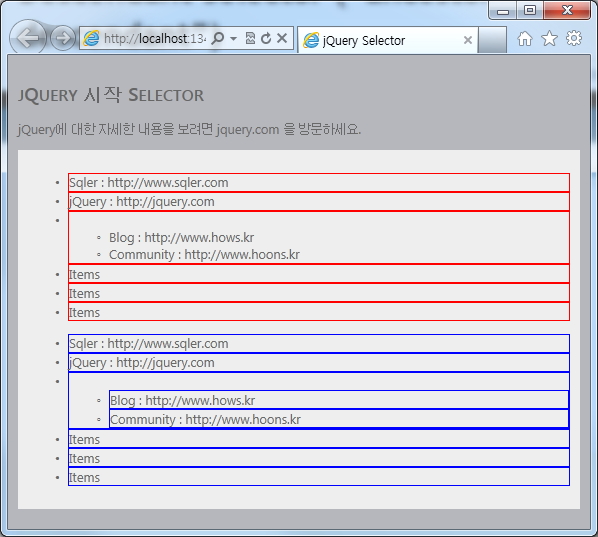
</**ul**>

</**div**>

</**body**>

</**html**>

[자식(Child) 셀렉터와 후손(Descendant) 셀렉터]



[자식(Child) 셀렉터와 후손(Descendant) 셀렉터에 의해 선택 된 모습]

빨강색으로 표시된 부분은 자식(Child) 셀렉터를 이용하여 개체를 선택한 모습이며, 파랑 색으로 표시된 부분은 후손(Descendant) 셀렉터에 의해 개체를 선택한 모습입니다.

자식(Child) 셀렉터를 이용했을 경우 ul -> li -> ul -> li 에 해당되는 요소는 선택이 되지 않고 바로 아래에 있는 자식(Child) 요소만 선택이 된 반면에 후손(Descendant) 셀렉터를 이용했을 경우 li에 해당하는 모든 요소를 선택 한 것을 확인 할 수 있습니다.

간단한 문법의 차이로 인해서 반환하는 요소의 결과의 차이가 많으니 두 가지 셀렉터의 차이점을 확실히 이해를 하시는 것은 매우 중요합니다.

**Next Adjacent Selector : $(“prev + next”)**

이전에는 요소와 요소 사이의 상, 하 관계를 통해 일치하는 항목을 찾았다면 이번 셀렉터와 다음에 설명드릴 셀렉터는 평행관계를 통해 일치하는 요소를 반환합니다.

Prev 요소 바로 다음에 나오는 형제(Adjacent) 수준의 next 요소와 일치하는 항목을 반환합니다.

<**html** xmlns="http://www.w3.org/1999/xhtml">

<**head**>

<**title**>jQuery Selector</**title**>

<**link** href="../Styles/Site.css" rel="stylesheet" type="text/css" />

<**style**>

**div** { background-color:#EEEEEE; padding:10px; }

</**style**>

<**script src="http://code.jquery.com/jquery-latest.js"**></**script**>

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

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

$("label + input").css("border", "1px solid #ff0000");

});

</**script**>

</**head**>

<**body**>

<**h2**>jQuery 시작 Selector</**h2**>

<**p**>jQuery에 대한 자세한 내용을 보려면 jquery.com 을 방문하세요.</**p**>

<**div**>

<**form**>

<**label**>Name:</**label**>

<**input** name="name" />

<**fieldset**>

<**label**>Newsletter:</**label**>

<**input** name="newsletter" />

</**fieldset**>

</**form**>

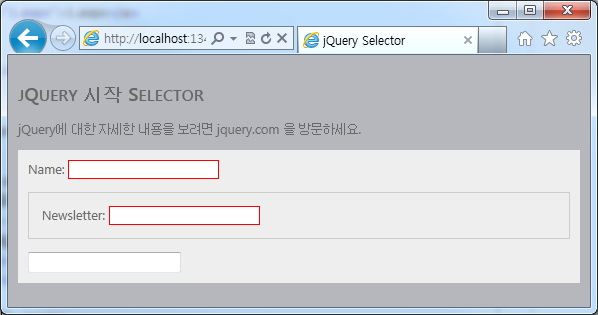
<**input** name="none" />

</**div**>

</**body**>

</**html**>

[label 다음의 input의 요소와 형제 수준이 동일한 요소를 찾습니다.]



[지정한 요소와 형제 수준이 동일한 요소의 모습]

$("label + input").css("border", "1px solid #ff0000");

<label> 요소 바로 다음(형제요소)에 있는 <input> 요소를 찾습니다. 동일한 패턴으로 된 모든 요소를 찾아 반환합니다.

$(“label > input”) 부분과 혼동하시는 분들이 많아 두 셀렉터의 차이를 다시 설명하자면 $(“label > input”)의 경우 <label>을 부모로 가지고 있는 <input> 요소를 찾는 것이고, $(“label + input”)의 경우는 <label>과 평등한 관계에 있는 <input>요소를 찾는 것 입니다.

**Next Siblings Selector : $(“prev ~ siblings”)**

Prev 요소 이후에 형제 요소 중 siblings와 동일한 요소를 반환합니다.

$(“#prev ~ div”)의 경우 id의 값이 prev인 요소를 찾고 해당 요소를 제외한 다음 형제 요소 중에 div와 동일한 요소를 찾아 반환을 합니다.

<**html** xmlns="http://www.w3.org/1999/xhtml">

<**head**>

<**title**>jQuery Selector</**title**>

<**link** href="../Styles/Site.css" rel="stylesheet" type="text/css" />

<**style**>

**div**,**span**

{

width:100px;

height:80px;

float:left;

padding:10px;

margin:10px;

background-color:#EEEEEE;

}

**div**#small

{

width:60px;

height:20px;

font-size:12px;

background:#fab;

}

</**style**>

<**script src="http://code.jquery.com/jquery-latest.js"**></**script**>

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

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

$("#prev ~ div").css("border", "3px groove blue");

});

</**script**>

</**head**>

<**body** style="padding:10px;">

<**h2**>jQuery 시작 Selector</**h2**>

<**p** id="content">jQuery에 대한 자세한 내용을 보려면 jquery.com 을 방문하세요.</**p**>

<**div**>div (doesn't match since before #prev)</**div**>

<**span** id="prev">span#prev</**span**>

<**div**>div sibling</**div**>

<**div**>div sibling

<**div** id="small">div niece</**div**>

</**div**>

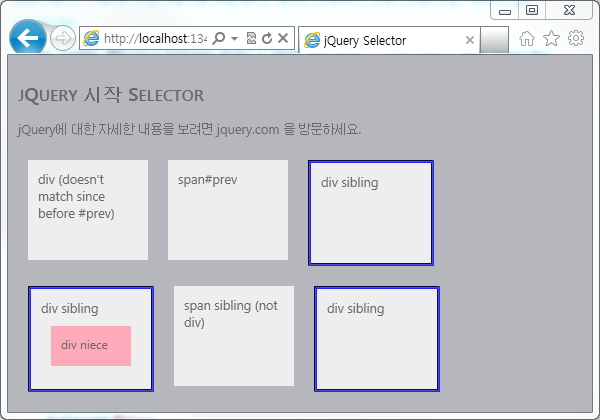
<**span**>span sibling (not div)</**span**>

<**div**>div sibling</**div**>

</**body**>

</**html**>

[자신을 제외한 이후 형제 요소 선택]



[자신을 제외한 동일한 형제 요소에 속하는 div 요소를 선택한 모습]

이번 시간에는 DOM 계층을 이용한 접근 방법에 대해 알아 보았습니다. 동적인 화면의 구성과 복잡한 UI에서 원하는 요소(개체)에 접근하기 위해서는 DOM의 구조에 대해 이해하고 접근방법에 익숙해 있어야 합니다.

**웹 프런티어와 함께하는 jQuery  기초강좌**

**7th - jQuery Filter의 기본필터의 사용**

이번 시간에는 jQuery 에서 제공하는 필터에 대해서 알아 보겠습니다. 부제를 붙이자면 "셀렉터와 필터의 만남"이 될꺼 같습니다.

**셀렉터에 날개를 달자 ? Filter를 이용한 개체 접근**이번 강좌에서는 jQuery에서 지원하고 있는 여러가지 필터중에 기본이 되는 부분과 jQuery의 DOM 필터 메서드,(filter())를 사용하여 사용자가 원하는 요소를 선택하는 방법에 대해 알아 보도록 하겠습니다.

|  |  |
| --- | --- |
| **형식(표현식)** | **설명** |
| :animated | 에니메이션이 동작중인 모든 요소와 일치하는 요소를 반환합니다. |
| :eq(index) | Index에 해당하는 요소를 반환합니다.(단일요소) |
| :even | 짝수의 요소를 반환합니다. (0부터 시작) |
| :odd | 홀수의 요소를 반환합니다. (0부터 시작) |
| :first | 첫번째 요소를 반환합니다. |
| :last | 마지막 요소를 반환합니다. |
| :gt(index) | Index 보다 높은 Index에 해당되는 요소를 모두 반환합니다. |
| :lt(index) | Index 보다 낮은 Index에 해당되는 요소를 모두 반환합니다. |
| :header | 모든 헤더 요소들을 반환합니다.(h1,h2,h3….) |
| :not(selector) | Selector와 일치되는 요소를 제외한 나머지 요소를 반환합니다. |
| :focus | 현재 포커스가 위치한 요소를 반환합니다. (1.6 이상에서 지원) |
| 표 1. [jQuery 필터의 종류] | |

jQuery의 필터는 말 그대로 거르다, 여과하다의 뜻으로 원하는 요소를 다양한 방식으로 걸러내는 역할을 합니다.

일반적으로 셀렉터와 함께 사용하는 경우가 대부분으로 셀렉터와 함께 써야 하는하는 것 알고 있는 분이 많이 있습니다.

하지만 필터는 단독으로도 사용이 가능하며, 필터와 필터를 연결해서도 사용이 가능합니다.  
테이블의 컬럼인 td 항목을 예로 들어 설명을 하자면 다음과 같습니다.

|  |  |
| --- | --- |
| $(“td:eq(0)”) | td 요소중에 첫번째 항목만을 선택합니다. (eq의 index는 0부터 시작입니다.)   eq의 index는 0부터 시작을 하며 0 이면 요소중 첫번째, 1이면 두번째 요소가 됩니다. |
| $(“td:even”)   $(“td:odd”) | td 요소중에 짝/홀수번째 요소를 선택합니다. 짝/홀수의 구분은 eq의 index와 동일하게 구분이 됩니다. |
| $(“td:first”)   $("td:last") | td 요소중에 첫번째 요소, 마지막 요소를 선택합니다. |
| $(“td:gt(2)”)   $(“td:lt(2)”) | td 요소중에 2번째 요소 이후의 모든 요소를 선택하거나 2번째 요소 이전의 모든 요소를 선택합니다. |

각각의 설명만으로도 각 필터가 어떠한 역할을 하고 있는지 쉽게 이해를 하실 수 있을것입니다. 간단한 예제를 통해 위에 다시 한번 살펴보도록 하겠습니다.

<**html** xmlns="http://www.w3.org/1999/xhtml">

<**head**>

<**title**>jQuery Selector</**title**>

<**link** href="../Styles/Site.css" rel="stylesheet" type="text/css" />

<**style**>

**div** { background : #FFF; padding:10px; margin:10px; }

**table** { border:1px solid #AAA; }

**td** { border:1px solid #AAA; width:40px; height:40px; }

</**style**>

<**script src="http://code.jquery.com/jquery-latest.js"**></**script**>

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

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

$("tr:eq(0) > td:eq(7)").css("background", "#EEE");

$("tr:eq(1) > td:gt(2)").css("background", "#EEE");

$("tr:eq(2) > td:odd").css("background", "#EEE");

$("tr:eq(3) > td:first").css("background", "#AAA");

$("tr:eq(3) > td:last").css("background", "#BBB");

});

</**script**>

</**head**>

<**body** style="padding:10px;">

<**h2**>jQuery 시작 Selector</**h2**>

<**p**>jQuery에 대한 자세한 내용을 보려면 jquery.com 을 방문하세요.</**p**>

<**div**>

<**table**>

<**tr**>

<**td**>#1</**td**><**td**>#2</**td**><**td**>#3</**td**><**td**>#4</**td**><**td**>#5</**td**>

<**td**>#6</**td**><**td**>#7</**td**><**td**>#8</**td**><**td**>#9</**td**><**td**>#10</**td**>

</**tr**>

<**tr**>

<**td**>#11</**td**><**td**>#12</**td**><**td**>#13</**td**><**td**>#14</**td**><**td**>#15</**td**>

<**td**>#16</**td**><**td**>#17</**td**><**td**>#18</**td**><**td**>#19</**td**><**td**>#20</**td**>

</**tr**>

<**tr**>

<**td**>#21</**td**><**td**>#22</**td**><**td**>#23</**td**><**td**>#24</**td**><**td**>#25</**td**>

<**td**>#26</**td**><**td**>#27</**td**><**td**>#28</**td**><**td**>#29</**td**><**td**>#30</**td**>

</**tr**>

<**tr**>

<**td**>#31</**td**><**td**>#32</**td**><**td**>#33</**td**><**td**>#34</**td**><**td**>#35</**td**>

<**td**>#36</**td**><**td**>#37</**td**><**td**>#38</**td**><**td**>#39</**td**><**td**>#40</**td**>

</**tr**>

</**table**>

</**div**>

</**body**>

</**html**>

[예제 1]



[예제 1을 적용한 결과 화면]

예제에서는 $(“td:even”)으로 사용했지만 $(“tr:even”).css(“background”, “#DDD”)와 같이 사용하면 각 tr 부분에 스타일을 작성하지 않고 간단하게 “zebra” 스타일의 목록을 만들 때 매우 유용합니다.

<**html** xmlns="http://www.w3.org/1999/xhtml">

<**head**>

<**title**>jQuery Selector</**title**>

<**link** href="../Styles/Site.css" rel="stylesheet" type="text/css" />

<**style**>

**div** { background : #FFF; padding:10px; margin:10px; }

**table** { border:1px solid #AAA; width:100% }

**td** { border:1px solid #AAA; }

</**style**>

<**script src="http://code.jquery.com/jquery-latest.js"**></**script**>

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

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

$("tr:even").css("background", "#AAA");

});

</**script**>

</**head**>

<**body** style="padding:10px;">

<**h2**>jQuery 시작 Selector</**h2**>

<**p**>jQuery에 대한 자세한 내용을 보려면 jquery.com 을 방문하세요.</**p**>

<**div**>

<**table**>

<**tr**>

<**td**>#1</**td**>

</**tr**>

<**tr**>

<**td**>#2</**td**>

</**tr**>

<**tr**>

<**td**>#3</**td**>

</**tr**>

<**tr**>

<**td**>#4</**td**>

</**tr**>

<**tr**>

<**td**>#5</**td**>

</**tr**>

<**tr**>

<**td**>#6</**td**>

</**tr**>

<**tr**>

<**td**>#7</**td**>

</**tr**>

<**tr**>

<**td**>#8</**td**>

</**tr**>

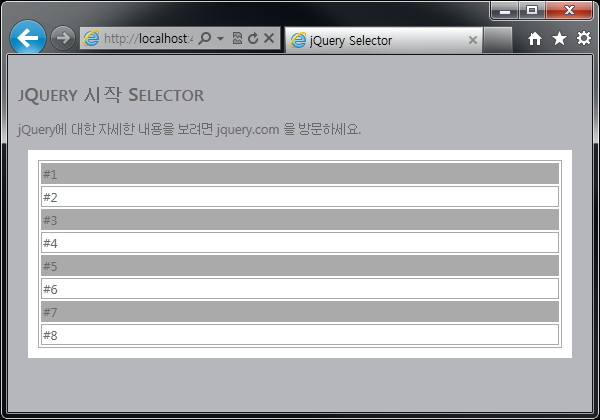
</**table**>

</**div**>

</**body**>

</**html**>

[예제 2. :even 필터를 이용한 Zebra 스타일 목록 만들기]



[예제 2를 이용한 Zebra 스타일 목록 결과]

예제중에서 “:even, :odd” 경우 “zebra” 줄무늬 스타일의 목록을 만들 때 자주 사용이 됩니다.

**:not(selector)**

**$(“td:not(‘.noselect’)”)**  
td 요소중에 클래스명이 “notSelect”인 항목을 제외한 요소의 집합을 선택합니다.

<**html** xmlns="http://www.w3.org/1999/xhtml">

<**head**>

<**title**>jQuery Selector</**title**>

<**link** href="../Styles/Site.css" rel="stylesheet" type="text/css" />

<**style**>

**div** { background : #FFF; padding:10px; margin:10px; }

**table** { border:1px solid #AAA; width:100% }

**td** { border:1px solid #AAA; }

</**style**>

<**script src="http://code.jquery.com/jquery-latest.js"**></**script**>

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

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

$("td:not(.notSelect)").css("background", "#AAA");

});

</**script**>

</**head**>

<**body** style="padding:10px;">

<**h2**>jQuery 시작 Selector</**h2**>

<**p**>jQuery에 대한 자세한 내용을 보려면 jquery.com 을 방문하세요.</**p**>

<**div**>

<**table**>

<**tr**>

<**td**>#1</**td**>

</**tr**>

<**tr**>

<**td**>#2</**td**>

</**tr**>

<**tr**>

<**td** class="notSelect">#3</**td**>

</**tr**>

<**tr**>

<**td** class="notSelect">#4</**td**>

</**tr**>

<**tr**>

<**td** class="notSelect">#5</**td**>

</**tr**>

<**tr**>

<**td**>#6</**td**>

</**tr**>

<**tr**>

<**td**>#7</**td**>

</**tr**>

<**tr**>

<**td**>#8</**td**>

</**tr**>

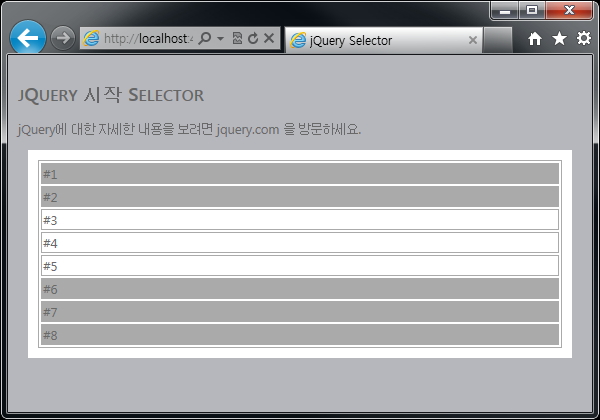
</**table**>

</**div**>

</**body**>

</**html**>

[예제 3. :not 필터의 사용]



[예제 3 :not 필터를 사용한 결과 화면]

**:focus**  
jQuery 1.6 버전에서 새롭게 추가된 필터로 현재 포커스가 위치한 요소를 선택 하며, 유저의 포커스가 위치한 지점의 위치를 표시하거나, 입력하고 있는 폼 요소를 강조하고 싶을 때 사용할 수 있습니다.

이번 시간에는 jQuery의 필터중에서 첫번째인 Basic Filter에 대해 알아 보았으며, 다음 시간에는 jQuery의 필터 두번째인 폼필터(FormFilter)에 대해 알아 보도록 하겠습니다.

**웹 프런티어와 함께하는 jQuery 기초강좌**

**8th - jQuery Filter의 폼(Form) 필터의 사용**

지난 시간에 설명드린 기본 필터의 사용법에 이어 폼(Form) 필터에 대해 알아 보기로 하겠습니다.

**폼 필터**  
폼 필터는 형식(text, checkbox, password, radio, file)을 기반으로 하는 폼 요소를 선택할 때 사용하는 필터로 사용법은 필터와 동일하여, 자세한 설명보다는 아래의 표로 대신 합니다.

|  |  |
| --- | --- |
| **폼 필터 종류** | **선택 폼** |
| :button | <input type=”butto” /> |
| :checkbox | <input type=”checkbox” /> |
| :checked | <input type=”checkbox” checked=”checked” /> |
| :disabled | <input type=”text” disabled=”disabled” /> |
| :enabled | <input type=”text” enabled=”enabled” /> |
| :file | <input type=”file” /> |
| :focus | (1.6 이상에서 지원) |
| :image | <input type=”image” /> |
| :input | <input> 모든 input 요소 |
| :password | <input type=”password” /> |
| :radio | <input type=”radio” /> |
| :reset | <input type=”reset” /> |
| :selected | <select><option selected="selected"></option></select> |
| :submit | <input type=”submit” /> |
| :text | <input type=”text” /> |
| :hidden | <input type=”hidden” /> |
| **표 1. [jQuery 폼 필터의 종류]** | |

표에서 보듯이 폼 필터 사용은 매우 간단하여 자세한 설명을 생략하고 필터 중 상태를 체크하는 필터에 대해 알아 보도록 하겠습니다.

**:checked , :selected**

<input type=”checkbox”/>체크박스, 라디오버튼<input type=”radio” />, 셀렉트<select>의 상태를 확인 하여 선택된 상태 요소들의 집합을 선택 반환합니다.

<**html** xmlns="http://www.w3.org/1999/xhtml">

<**head**>

<**title**>jQuery Selector</**title**>

<**link** href="../Styles/Site.css" rel="stylesheet" type="text/css" />

<**style**>

**div**,**pre** { background : #FFF; padding:10px; margin:10px; }

**table** { border:1px solid #AAA; }

**td** { border:1px solid #AAA; width:25px; height:25px; }

</**style**>

<**script src="http://code.jquery.com/jquery-latest.js"**></**script**>

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

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

$(":checked,:selected").css("border", "2px solid red");

$(":selected").css("background", "red");

});

</**script**>

</**head**>

<**body** style="padding:10px;">

<**h2**>jQuery 시작 Selector</**h2**>

<**p**>jQuery에 대한 자세한 내용을 보려면 jquery.com 을 방문하세요.</**p**>

<**div**>

<**ul**>

<**li**><**input** type="checkbox" checked="checked" />CheckBox #1 - </**li**>

<**li**><**input** type="checkbox" />CheckBox #2</**li**>

<**li**><**input** type="checkbox" checked="checked" />CheckBox #3</**li**>

</**ul**>

<**select** multiple="multiple">

<**option**>Option #1</**option**>

<**option** selected="selected">Option #2</**option**>

<**option**>Option #3</**option**>

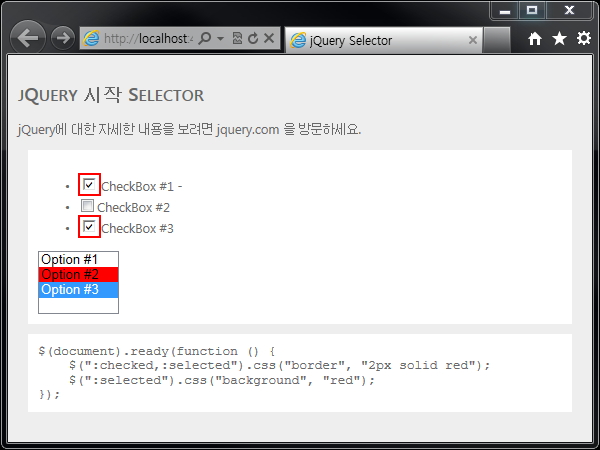
</**select**>

</**div**>

</**body**>

</**html**>

**[예제 1. 폼필터의 사용예제]**



**[예제 1 Selected, Checked를 폼 필터를 이용한 결과 모습]**

**:enabled, :disabled**사용이 가능하거나, 사용이 불가능한 요소를 선택하여 해당 요소의 집합을 반환합니다.  
<input type="text" disabled="disabled" />의  경우 :disabled로, 일반적인 모습일 경우 :enabled로 관련 요소의 집합을 선택 할 수 있습니다.

이번 시간에는 jQuery의 필터중에서 두 번째인 Form Filter에 대해 알아 보았으며, 다음 시간에는 jQuery의 필터 세번 째인 자식필터(Child Filter)에 대해 알아 보도록 하겠습니다.

**웹 프런티어와 함께하는 jQuery 기초강좌**

**9th - jQuery Filter의 자식(Child) 필터의 사용**

지난 시간에 설명드린  필터의 사용법에 이어 자식(?Child) 필터에 대해 알아 보기로 하겠습니다.

자식(Child)에 대한 내용을 모르고 계신다면 이전에 설명드린 HTML DOM에 대한 강좌를 한번 확인 하시기 바랍니다.

**자식필터<Child Filter>**  
아래 목록을 통해 나열된 내용을 보면 이번강의 초반에 나왔던 필터와 비슷하게 보이실 겁니다.  사용하는 구문도 많은 부분이 비슷하지만 결과는 정말 다릅니다. 우선 자식필터(Child Fiter)에 대해 간단히 정리하고 앞서 말한 결과의 차이에 대해 알아 보기로 하겠습니다.

|  |  |
| --- | --- |
| **필터 종류 및 형식** | **필터 설명** |
| :first-child | 자식 요소 중 첫번째에 해당하는 요소를 모두 반환합니다. |
| :last-child | 자식 요소 중 마지막에 해당하는 요소를 모두 반환합니다. |
| :nth-child(index/odd/even/equation) | 자식 요소 중 지정된 값에 해당하는 요소를 모두 반환합니다. |
| :only-child | 자신이 부모 요소와 유일한 자식인 모든 요소를 반환합니다. |
| 표 1. [jQuery 자식(Child)필터의 종류] | |

앞서 다룬 기본 필터와 유사한 모습을 하고 있어 많은 분들이 비슷한 동작을 하지 않을까 하고 생각 하시는 경우가 많으나, 기본 필터와는 많은 부분에서 차이가 존재합니다. 기본필터의 :first, :last의 경우 해당되는 단일 요소만 선택하지만 자식필터(Child Filter)의 경우 단일요소가 아닌 집합을 선택 하게 됩니다.

테이블의 “td” 요소를 기준으로 두개의 차이점을 비교해 보았습니다.

<**html** xmlns="http://www.w3.org/1999/xhtml">

<**head**>

<**title**>jQuery Selector</**title**>

<**link** href="../Styles/Site.css" rel="stylesheet" type="text/css" />

<**style**>

**div** { background : #FFF; padding:10px; margin:10px; }

**table** { border:1px solid #AAA; }

**td** { border:1px solid #AAA; width:25px; height:25px; }

</**style**>

<**script src="http://code.jquery.com/jquery-latest.js"**></**script**>

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

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

//Script Here

});

</**script**>

</**head**>

<**body** style="padding:10px;">

<**h2**>jQuery 시작 Selector</**h2**>

<**p**>jQuery에 대한 자세한 내용을 보려면 jquery.com 을 방문하세요.</**p**>

<**div**>

<**table**>

<**tr**>

<**td**>#1</**td**><**td**>#2</**td**><**td**>#3</**td**><**td**>#4</**td**><**td**>#5</**td**>

</**tr**>

<**tr**>

<**td**>#6</**td**><**td**>#7</**td**><**td**>#8</**td**><**td**>#9</**td**><**td**>#10</**td**>

</**tr**>

<**tr**>

<**td**>#11</**td**><**td**>#12</**td**><**td**>#13</**td**><**td**>#14</**td**><**td**>#15</**td**>

</**tr**>

<**tr**>

<**td**>#16</**td**><**td**>#17</**td**><**td**>#18</**td**><**td**>#19</**td**><**td**>#20</**td**>

</**tr**>

<**tr**>

<**td**>#21</**td**><**td**>#22</**td**><**td**>#23</**td**><**td**>#24</**td**><**td**>#25</**td**>

</**tr**>

</**table**>

</**div**>

</**body**>

</**html**>

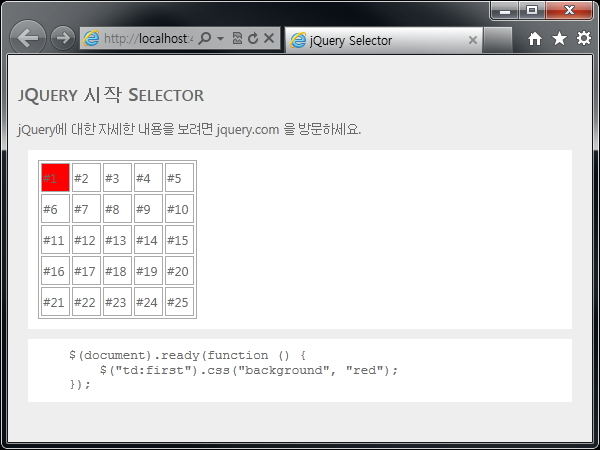
<**script type="text/javascript"**>

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

$("td:first").css("background", "red");

});

</**script**>



[기본 필터 :first를 사용한 결과]

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

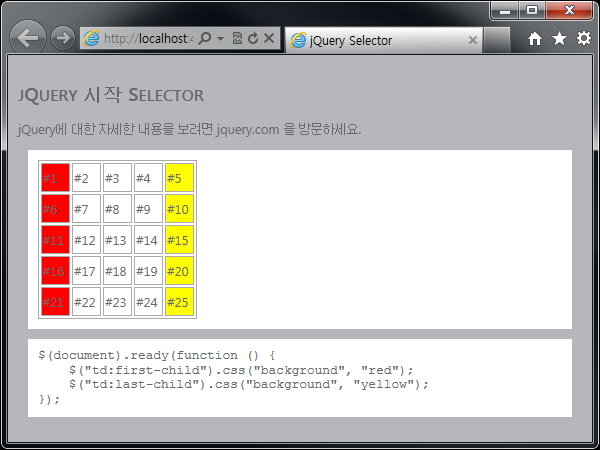
$(document).ready(**function** () {

$("td:first-child").css("background", "red");

$("td:last-child").css("background", "yellow");

});

</**script**>



[자식필터인 :first-child, :last-child를 사용한 결과]

두개의 차이점이 보이시나요 ? 기본 필터인 :first의 경우 “td”요소 하나만 선택하는 반면 :first-child, :last-child는 첫번째 마지막 수준의 “td”요소를 모두 선택합니다.

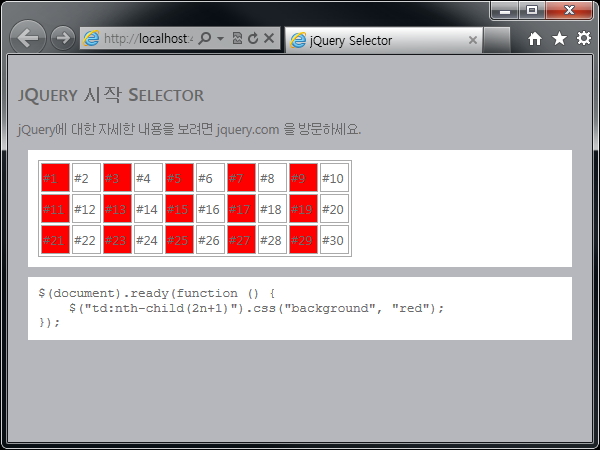
**:nth-child**

마지막으로 nth-child라는 필터에 대해 알아 보겠습니다.

기본적인 동작은 다른 자식필터와 동일합니다. 특이한 점은 index, even, odd 같은 값을 지정하여 사용을 하거나 수식을 이용하여 규칙적인 동작을 하도록 할 수 있다는 점입니다.

예를 들면 다음과 같은 형태를 가지고 있습니다.  
$(“td:nth-child(0)”), $(“td:nth-child(even)”), $(“td:nth-child(2n+1)”) 앞에 두가지 경우에 대해서는 앞에서 설명한 내용만으로 충분히 이해가 되실꺼라 생각하며 마지막 수식을 이용한 자식필터에 대해 간단히 설명을 하도록 하겠습니다.

$(“td:nth-child(2n+1)”)을 풀어보면 2의배수(2n)에 1을 더한 값에 해당하는 위치에 있는 요소를 선택을 하게 됩니다. 여기서 가장 중요한 부분은 n은 1이 아닌 0부터 시작을 한다는 것입니다. 1이 아닌 0부터 시작을 하므로 수식을 풀면 “1, 3, 5, 7, 9, 11…”의 형식이 됩니다. 다음 예제를 통해 정말 위에 풀이한 값이 맞는지 확인을 해 보도록 하겠습니다.



[:nth-child 필터를 사용한 결과 (수식에 일치하는 항목을 찾아 붉은색으로 표시합니다.) ]

앞에서 계산 값(1, 3, 5, 7, 9, 11…)처럼 해당 요소가 선택 되어진 결과를 볼 수 있습니다. 1이 아닌 0으로 시작한다는 것만 다시 강조 드리면서 필터에 대한 이야기를 여기서 정리를 합니다.

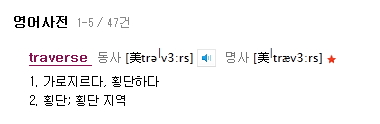
지금까지 jQuery에서 제공하고 있는 필터에 대해 알아 보았습니다.

**웹 프런티어와 함께하는 jQuery 기초강좌**

**10th - jQuery Traverse(탐색)**

**첫 번째 이야기 필터링**

**jQuery의 탐색(Traverse) 메서드를 이용하여 셀렉터에 날개를 달자**



“Traverse”는 위에 보시는 바와 같이 가로지르다, 횡단이란 뜻을 가지고 있으며, jQuery에서 지원하는 Traverse 메서드 또한 HTML을 가로지르거나, 횡단하는 것 처럼 탐색하는 역할을 합니다. 해당 메서드를 간략히 설명하면 셀렉터를 통해 선택한 개체에서 다시 한번 개체를 찾고, 필터링하고, 추가하는 작업을 쉽게 구현 할 수 있도록 도와주는 메서드라고 생각 하시면 됩니다.

물론 기존의 셀렉터를 통해서도 충분히 원한는 개체를 탐색하거나, 선택 하는데 큰 지장이 없습니다만, Traverse 메서드를 이용하면 셀렉터를 통해 가져온 개체를 즉 1차 결과물에 추가적인 작업을 통해 2차, 3차 등의 결과를 쉽게 얻을 수 있다는 큰 장점이 있습니다.

이러한 Traverse 메서드는 크게 Filtering, Miscellaneous Traverse, Tree Traverse 3가지로 구분을 하고 있으며, 이번 시간을 통해 자세히 알아 보도록 하겠습니다.

**Filtering**

첫 번째로 알아볼 필터링 관련 메서드 입니다.

우선 필터링 메서드의 종류에 대해서 알아 보기로 하겠습니다.

|  |  |
| --- | --- |
| **메서드 종류(형식)** | **메서드 설명** |
| .eq(index) | 선택한 요소들 중에서 인덱스와 일치하는 단일 요소를 선택 반환합니다. |
| .filter(expr) | 선택한 요소에서 표현식(expr)과 일치하는 요소의 집합을 선택 반환합니다.   표현식에는 selector, function, element, jQuery object가 올 수 있습니다. |
| .first() | 선택한 요소에서 첫 번째 단일 요소를 선택 반환합니다. |
| .has(selector) | 선택한 요소에서 selector항목을 가지고 있는 요소의 집합을 선택 반환합니다. |
| .is(expr) | 표현식과 일치하는 조건이 있으면 true를 반환합니다.   표현식에는 selector, function, element, jQuery object가 올 수 있습니다. |
| .last() | .first() 와 반대되는 메서드로 마지막 단일 요소를 선택 반환합니다. |
| .map(callback) | jQuery 개체에 있는 요소의 집합을 다른 집합으로 변경해서 이동 시킵니다. |
| .not(expr) | 표현식과 일치하지 않는 요소의 집합을 선택 반환합니다. |
| .slice(start,[end] | 선택한 요소에서 start, end번 째에 해당하는 집합을 선택 반환합니다. |
| 표 1. [jQuery 탐색 필터 메서드 종류] | |

생각보다 많은 메서드를 제공하고 있으나, 이중에 몇 개는 기존에 설명 드렸던 셀렉터의 필터와 겹치는 내용이 있습니다.

대표적으로 first(), last(), eq(), not() 메서드를 들 수있습니다.

이전 필터와 살짝 비교를 하면 다음과 같습니다.

**$(“td:eq(0)”) == $(“td”).eq(0)**

**$(“td:first”) == $(“td”).first()**

**$(“td:last”) == $(“td”).last()**

**$(“td:not(‘.myClass’)”) == $(“td”).not(“.myClass”)**

양쪽다 모두 동일한 결과를 보입니다.

그런데, 결과적으로 동일한 역할을 하는데 왜 여러 개를 만들어 놓았을까요 ? 단순히 개발자의 맘일까요 ? 이번강의 앞 부분에 언급한 1차 결과물에 추가적인 작업과 이후에 설명을 드릴 miscellaneous항목에 존재하는 .end() 라는 메서드를 보시면 좀더 쉽게 이해가 되실꺼라 생각합니다.

이제부터는 앞서 비교한 메서드를 제외한 나머지 메서드에 대해 간단히 알아 보도록 하겠습니다.

**.filter(expr)**

1차적으로 셀렉터를 통해 선택한 요소들의 집합에서 또 한번의 필터링을 통해 원하는 개체를 선택하는 메서드입니다.

표현식 부분에는 Selector, function, element, jQuery object가 위치할 수 있습니다.

이 메서드 또한 앞의 필터와 중복되는 부분이 존재합니다. 간단한 예를 들면 다음과 같습니다.

$(“td:even”) == $(“td”).filter(“:even”)

동일한 결과를 나타냅니다. 하지만 필터 메서드의 강점은 function을 이용하여 마치 필터를 확장해서 쓰는것과 같은 효과를 얻을 수 있는 점입니다.

다음은 셀렉터를 통해 선택한 개체의 집합을 필터 메서드를 통해 3의 배수에 해당하는 요소를 선택하는 예제이며, function(index) index는 $(“li”)를 통해 가져온 개체 집합의 번호입니다.

<**html** xmlns="http://www.w3.org/1999/xhtml">

<**head**>

<**title**>jQuery Selector</**title**>

<**link** href="../Styles/Site.css" rel="stylesheet" type="text/css" />

<**style**>

**div**,**pre** { background : #FFF; padding:10px; margin:10px; }

**table** { border:1px solid #AAA; }

**td** { border:1px solid #AAA; width:25px; height:25px; }

</**style**>

<**script src="http://code.jquery.com/jquery-latest.js"**></**script**>

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

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

$("td").filter(**function** (index) {

**return** index % 3 == 0;

}).css("background", "red");

});

</**script**>

</**head**>

<**body** style="padding:10px;">

<**h2**>jQuery 시작 Selector</**h2**>

<**p**>jQuery에 대한 자세한 내용을 보려면 jquery.com 을 방문하세요.</**p**>

<**div**>

<**table**>

<**tr**>

<**td**>#1</**td**><**td**>#2</**td**><**td**>#3</**td**><**td**>#4</**td**><**td**>#5</**td**>

</**tr**>

<**tr**>

<**td**>#6</**td**><**td**>#7</**td**><**td**>#8</**td**><**td**>#9</**td**><**td**>#10</**td**>

</**tr**>

<**tr**>

<**td**>#11</**td**><**td**>#12</**td**><**td**>#13</**td**><**td**>#14</**td**><**td**>#15</**td**>

</**tr**>

<**tr**>

<**td**>#16</**td**><**td**>#17</**td**><**td**>#18</**td**><**td**>#19</**td**><**td**>#20</**td**>

</**tr**>

<**tr**>

<**td**>#21</**td**><**td**>#22</**td**><**td**>#23</**td**><**td**>#24</**td**><**td**>#25</**td**>

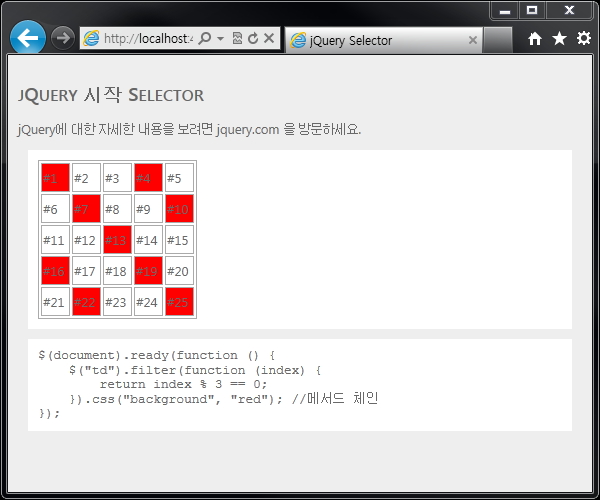
</**tr**>

</**table**>

</**div**>

</**body**>

</**html**>



[필터 메서드를 사용하여 3의 배수에 해당하는 요소에 스타일을 적용]

물론 위 예제는 자식필터(child filter)를 이용하여 구현이 가능합니만 여기서 눈 여겨 볼 내용은 이렇게 자신이 원하는 기능을 추가로 하여 개체를 선택 할 수 있다는 것입니다.

**.not()**

필터 메서드와는 반대로 동작하는 메서드입니다. 필터 메서드는 표현식과 일치하는 요소를 선택하는 반면에 이(.not()) 메서드는 표현식과 일치하지 않는 요소를 선택 반환합니다.

앞의 예제에서 스크립트만 변경을 해 보도록 하겠습니다.

<**html** xmlns="http://www.w3.org/1999/xhtml">

<**head**>

<**title**>jQuery Selector</**title**>

<**link** href="../Styles/Site.css" rel="stylesheet" type="text/css" />

<**style**>

**div**,**pre** { background : #FFF; padding:10px; margin:10px; }

**table** { border:1px solid #AAA; }

**td** { border:1px solid #AAA; width:25px; height:25px; }

</**style**>

<**script src="http://code.jquery.com/jquery-latest.js"**></**script**>

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

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

$("td").not(**function** (index) {

**return** index % 3 == 0;

}).css("background", "red");

});

</**script**>

</**head**>

<**body** style="padding:10px;">

<**h2**>jQuery 시작 Selector</**h2**>

<**p**>jQuery에 대한 자세한 내용을 보려면 jquery.com 을 방문하세요.</**p**>

<**div**>

<**table**>

<**tr**>

<**td**>#1</**td**><**td**>#2</**td**><**td**>#3</**td**><**td**>#4</**td**><**td**>#5</**td**>

</**tr**>

<**tr**>

<**td**>#6</**td**><**td**>#7</**td**><**td**>#8</**td**><**td**>#9</**td**><**td**>#10</**td**>

</**tr**>

<**tr**>

<**td**>#11</**td**><**td**>#12</**td**><**td**>#13</**td**><**td**>#14</**td**><**td**>#15</**td**>

</**tr**>

<**tr**>

<**td**>#16</**td**><**td**>#17</**td**><**td**>#18</**td**><**td**>#19</**td**><**td**>#20</**td**>

</**tr**>

<**tr**>

<**td**>#21</**td**><**td**>#22</**td**><**td**>#23</**td**><**td**>#24</**td**><**td**>#25</**td**>

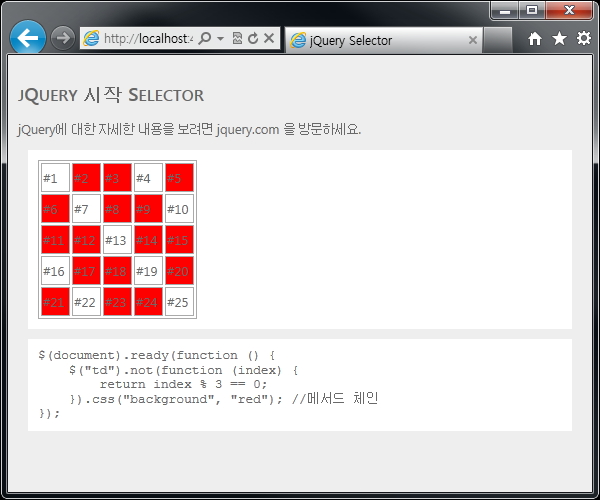
</**tr**>

</**table**>

</**div**>

</**body**>

</**html**>



[앞의 결과와는 다르게 3의 배수가 아닌 요소를 표시합니다.]

**.has(selector)**

기본 셀렉터를 통해 가져온 개체의 집합에서 selector에 해당하는 요소를 가진 개체만을 선택 반환합니다.

다음 예제를 통해 간단히 알아보고 넘어 가도록 하겠습니다.

<**html** xmlns="http://www.w3.org/1999/xhtml">

<**head**>

<**title**>jQuery Selector</**title**>

<**link** href="../Styles/Site.css" rel="stylesheet" type="text/css" />

<**style**>

**div**,**pre** { background : #FFF; padding:10px; margin:10px; }

**table** { border:1px solid #AAA; }

**td** { border:1px solid #AAA; width:25px; height:25px; }

</**style**>

<**script src="http://code.jquery.com/jquery-latest.js"**></**script**>

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

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

$("li").has("ul").css("background", "red");

});

</**script**>

</**head**>

<**body** style="padding:10px;">

<**h2**>jQuery 시작 Selector</**h2**>

<**p**>jQuery에 대한 자세한 내용을 보려면 jquery.com 을 방문하세요.</**p**>

<**div**>

<**ul**>

<**li**>list item 1</**li**>

<**li**>list item 2

<**ul**>

<**li**>list item 2-a</**li**>

<**li**>list item 2-b</**li**>

</**ul**>

</**li**>

<**li**>list item 3</**li**>

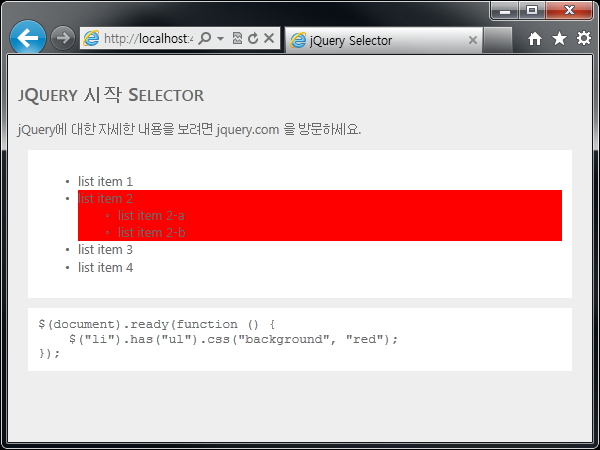
<**li**>list item 4</**li**>

</**ul**>

</**div**>

</**body**>

</**html**>



셀렉터를 통해 가져온 li개체에서 has() 함수를 통해 “ul” 요소를 가지고 있는 개체만을 선택 하는걸 확인 할 수 있습니다.

**.is(expr)**

기본 셀렉터에서 가져온 개체의 집합에서 표현 식과 일치하는 부분이 있는지에 대한 여부를 반환합니다.

집합에서 단 하나라도 일치하는 항목이 존재한다면 “true”를 반환합니다. 물론 일치하는 항목이 존재하지 않다면 “false”를 반환하게 됩니다.

<**html** xmlns="http://www.w3.org/1999/xhtml">

<**head**>

<**title**>jQuery Selector</**title**>

<**link** href="../Styles/Site.css" rel="stylesheet" type="text/css" />

<**style**>

**div**,**pre** { background : #FFF; padding:10px; margin:10px; }

**table** { border:1px solid #AAA; }

**td** { border:1px solid #AAA; width:25px; height:25px; }

</**style**>

<**script src="http://code.jquery.com/jquery-latest.js"**></**script**>

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

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

$("\*").each(**function** () {

**if** ($(**this**).is("li")) {

$(**this**).css("border", "1px solid red");

}

});

});

</**script**>

</**head**>

<**body** style="padding:10px;">

<**h2**>jQuery 시작 Selector</**h2**>

<**p**>jQuery에 대한 자세한 내용을 보려면 jquery.com 을 방문하세요.</**p**>

<**div**>

<**ul**>

<**li**>list item 1</**li**>

<**li**>list item 2

<**ul**>

<**li**>list item 2-a</**li**>

<**li**>list item 2-b</**li**>

</**ul**>

</**li**>

<**li**>list item 3</**li**>

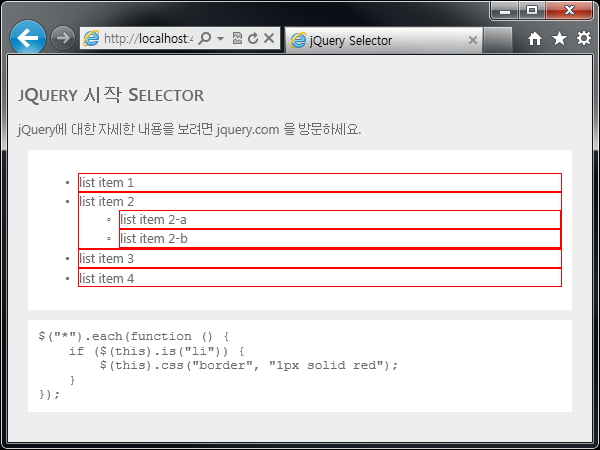
<**li**>list item 4</**li**>

</**ul**>

</**div**>

</**body**>

</**html**>



모든 요소를 선택하고 “each()”라는 메서드를 통해서 개체의 집합에 접근 후 “li”와 동일한 요소일 경우 지정한 스타일이 적용된 걸 확인 할 수 있습니다. “each()” 라는 메서드는 이후에 설명을 다시 드리겠지만, jQuery를 통해 가져온 개체의 집합을 foreach와 같이 순환하는 기능을 제공합니다. 자세한 설명은 이후에 다시 드리도록 하겠습니다.

**.map(callback)**

셀렉터를 통해 가져온 개체의 집합을 jQuery개체의 배열 형식으로 반환합니다.

설명을 쓰고보니, 그냥 셀렉터와의 큰 차이점이 없어 보입니다, 하지만 다행이도 아직 callback이라는 녀석이 남아 있습니다.

Callback 함수를 이용하면 셀렉터를 이용해 가져온 개체의 집합에서 원하는 작업을 진행하고 해당 작업의 결과를 jQuery의 배열 형식으로 반환 할 수 있습니다. 설명을 하면 할 수록 말이 점점 어려워 지고 있는 느낌은 저 뿐인가요 ? 다음 예제를 보시면 쉽게 이해가 되실겁니다.

<**html** xmlns="http://www.w3.org/1999/xhtml">

<**head**>

<**title**>jQuery Selector</**title**>

<**link** href="../Styles/Site.css" rel="stylesheet" type="text/css" />

<**style**>

**div**,**pre** { background : #FFF; padding:10px; margin:10px; }

**table** { border:1px solid #AAA; }

**td** { border:1px solid #AAA; width:25px; height:25px; }

</**style**>

<**script src="http://code.jquery.com/jquery-latest.js"**></**script**>

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

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

**var** liText = $("li").map(**function** (index) {

**return** "[(" + index + ")" + $(**this**).text() + "]";

}).get().**join**(',');

$("#result").text(liText);

});

</**script**>

</**head**>

<**body** style="padding:10px;">

<**h2**>jQuery 시작 Selector</**h2**>

<**p**>jQuery에 대한 자세한 내용을 보려면 jquery.com 을 방문하세요.</**p**>

<**div**>

<**ul**>

<**li**>list item 1</**li**>

<**li**>list item 2

<**ul**>

<**li**>list item 2-a</**li**>

<**li**>list item 2-b</**li**>

</**ul**>

</**li**>

<**li**>list item 3</**li**>

<**li**>list item 4</**li**>

</**ul**>

</**div**>

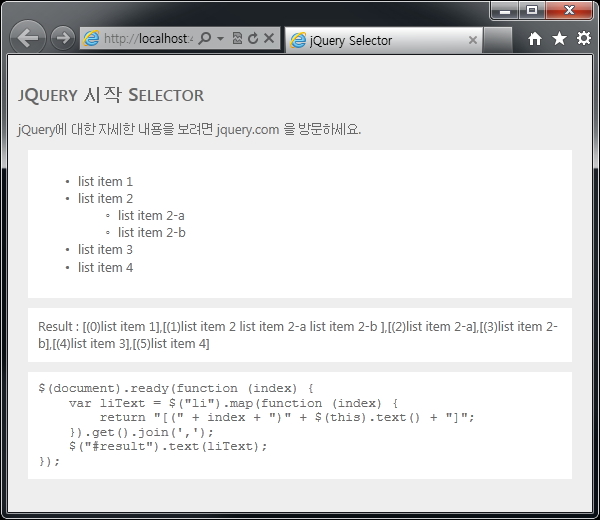
<**div**>

Result : <**span** id="result"></**span**>

</**div**>

</**body**>

</**html**>



이번에는 get(), join(), text() 라는 새로운 메서드가 보입니다.

예제의 설명을 위해 간단히 말씀을 드리면, get(), join() 메서드는 집합(배열) 형식의 개체에 접근하는 방법과 join() 메서드의 인자 값으로 연결된 문자를 반환하게 됩니다. text()메서드는 셀렉터를 통해 가져온 개체의 text를 가져오는 역할을 합니다. Value 값에 접근 하기 위해서는 val()이라는 메서드를 사용합니다.

위 예제는 “li” 요소를 선택 후 해당 개체의 text값을 map() 메서드를 통해 반환하고 있습니다. 지금까지 설명 드린 기본 셀렉터, 필터에서도 위와 동일한 기능을 제공하고 있으나 이번 경우에도 개발자가 원하는 작업을 정의, 확장하여 사용할 수 있다는 큰 장점이 있습니다.

처음 이 메서드의 설명을 어떻게 해야 할지 막막했습니다. 이게 뭔지 어떠한 건지는 알겠는데.. 설명은 어려운.. 아직은 내공이 많이 부족한거 같습니다. 특정 패턴의 집합을 만들기에 차~암 좋은데 어떻게 설명할 방법이...

**.slice(start, [end])**

선택한 요소에서 start 번째에서 시작하고 end번째에서 끝나는 개체의 집합을 반환합니다.

기본 셀렉터에서 선택된 요소는 0부터 시작을 합니다. start는 포함을 하나, end 번째 요소는 포함을 하지 않으니, 이점 유의 하시기 바랍니다. 간단한 두 개의 예시를 통해 알아보면 다음과 같습니다.

총 8개의 요소를 선택했다고 가정합니다.

각 요소의 집합에 번호를 주면, 해당 인덱스는 0부터 시작이므로 0, 1, 2, 3, 4, 5, 6, 7이 됩니다.

1) .slice(2) 경우 : 2, 3, 4, 5, 6, 7에 해당하는 개체의 집합을 반환합니다.

2) .slice(2,4) 경우 : 2, 3 에 해당하는 개체의 집합을 반환합니다.

1)의 경우 0, 1, 2, 3, 4, 5, 6, 7 앞에서 2개의 집합을 제거 합니다.

2)의 경우 0, 1, 2, 3, 4, 5, 6, 7 앞에서 2개의 집합을 동일하게 제거 후 4번째를 포함한 집합을 한번 더 제거 후 반환하게 됩니다.

두 경우의 차이가 이해가 되시나요 ?

jQuery 사이트에서 제공하는 .slice() 메서드 설명에 있는 예제를 통해 살펴 본다면 아마도 쉽게 이해가 되리라 생각합니다.

<!DOCTYPE html>

<**html**>

<**head**>

<**style**>

**div** { width:40px; height:40px; margin:10px; float:left;

border:2px solid blue; }

**span** { color:red; font-weight:bold; }

**button** { margin:5px; }

</**style**>

<**script src="http://code.jquery.com/jquery-latest.js"**></**script**>

</**head**>

<**body**>

<**p**><**button**>Turn slice yellow</**button**>

<**span**>Click the button!</**span**></**p**>

<**div**></**div**>

<**div**></**div**>

<**div**></**div**>

<**div**></**div**>

<**div**></**div**>

<**div**></**div**>

<**div**></**div**>

<**div**></**div**>

<**div**></**div**>

<**script**>

**function** colorEm() {

**var** $div = $("div");

**var** start = Math.**floor**(Math.**random**() \*

$div.length);

**var** end = Math.**floor**(Math.**random**() \*

($div.length - start)) +

start + 1;

**if** (end == $div.length) end = undefined;

$div.css("background", "");

**if** (end)

$div.**slice**(start, end).css("background", "yellow");

**else**

$div.**slice**(start).css("background", "yellow");

$("span").text('$("div").slice(' + start +

(end ? ', ' + end : '') +

').css("background", "yellow");');

}

$("button").click(colorEm);

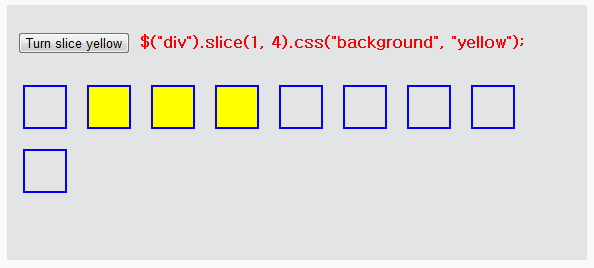
</**script**>

</**body**>

</**html**>

해당 소스는 버튼을 클릭할 때 마다 랜덤 한 숫자를 slice() 메서드에 대입 후 결과를 보여주는 코드이며, 아직 설명하지 않은 부분이 있어 자세한 설명은 생략을 하도록 하겠습니다.

실행 화면은 다음과 같습니다.



버튼을 누를 때마다 어떠한 .slice() 메서드에 어떠한 인자가 들어갔으며, 인자 값에 따른 결과를 보여주고 있습니다. 아직 이해가 안 되신다면 http://api.jquery.com/slice/ 페이지에서 예제를 실행해 보시기 바랍니다.

**웹 프런티어와 함께 하는 jQuery 기초강좌**

**11th - jQuery Traverse(탐색)**

**두 번째 이야기 - Miscellaneous**

탐색 메서드 두 번째 시간으로 이번에는 Miscellaneous 메서드에 대해 알아 보도록 하겠습니다.

우선 Miscellaneous 메서드의 종류를 정리하면 다음과 같습니다.

|  |  |
| --- | --- |
| **메서드** | **메서드 설명** |
| .add() | 일치하는 요소의 집합에 요소를 추가합니다. |
| .andSelf() | 현재 설정 스택에 요소의 이전 설정을 추가합니다. |
| .contents() | 텍스트 및 주석 노드를 포함 일치하는 요소 집합의 자식 집합을 반환합니다. |
| .end() | 이전 상태로 일치하는 집합을 반환합니다. |
| 표 1. [jQuery Miscellaneous 메서드 종류] | |

Miscellaneous의 뜻을 사전에서 찾아 보면 ‘여러 가지 종류의, 이것저것 다양한’ 이란 뜻을 가지고 있습니다. 아마도 해당 메서드의 성격이 너무나도 달라서 카테고리의 이름을 이렇게 잡은 듯 합니다.

**add()**

이름에서도 쉽게 알 수 있듯이 뭔가를 더하는 역할을 하는 메서드로 다음과 같이 사용을 합니다.

.add(selector)

.add(elements)

.add(html)

셀렉터를 통해 선택한 개체에 ()안에 값을 더하는 메서드로 selector를 통해 가져온 개체, span과 같은 요소, <span>testSpan</span>과 같은 html 요소가 올 수 있습니다.

<**html** xmlns="http://www.w3.org/1999/xhtml">

<**head**>

<**title**>jQuery Selector</**title**>

<**link** href="../Styles/Site.css" rel="stylesheet" type="text/css" />

<**style**>

**div**,**pre** { background : #FFF; padding:10px; margin:10px; }

**table** { border:1px solid #AAA; }

**td** { border:1px solid #AAA; width:25px; height:25px; }

</**style**>

<**script src="http://code.jquery.com/jquery-latest.js"**></**script**>

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

$(document).ready(function() {

$("li").add("div > p").css("border", "1px solid red");

});

</**script**>

</**head**>

<**body** style="padding:10px;">

<**h2**>jQuery 시작 Selector</**h2**>

<**p**>jQuery에 대한 자세한 내용을 보려면 jquery.com 을 방문하세요.</**p**>

<**div**>

<**ul**>

<**li**>list item 1</**li**>

<**li**>list item 2</**li**>

<**li**>list item 3</**li**>

<**li**>list item 4</**li**>

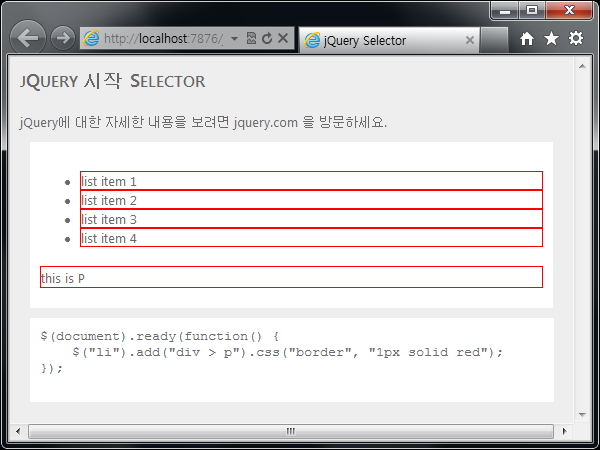
</**ul**>

<**p**>this is P</**p**>

</**div**>

</**body**>

</**html**>



**.andSelf()**

사용빈도가 상당히 낮을 것으로 추측을 하는 메서드로 저 또한 이번 강좌를 준비하면서 알게 된 메서드입니다. 앞서 설명한 부분을 다시 보면 “현재 설정 스택에 요소의 이전 설정을 추가합니다.” 말이 좀 어렵습니다.

우선 jQuery 문서에 있는 예제를 살펴 보기로 하겠습니다.

<!DOCTYPE html>

<**html**>

<**head**>

<**style**>

**p**, **div** { margin:5px; padding:5px; }

.border { border: 2px solid red; }

.background { background:yellow; }

</**style**>

<**script src="http://code.jquery.com/jquery-latest.js"**></**script**>

</**head**>

<**body**>

<**div**>

<**p**>First Paragraph</**p**>

<**p**>Second Paragraph</**p**>

</**div**>

<**script**>

$("div").find("p").andSelf().addClass("border");

$("div").find("p").addClass("background");

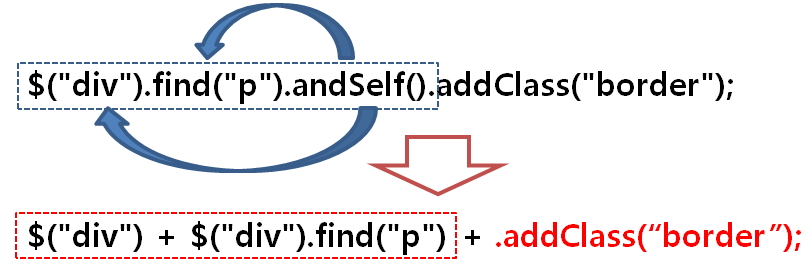
</**script**>

</**body**>

</**html**>



다음 그림을 보면 좀더 이해가 쉽습니다.



**.contents()**

셀렉터를 통해 선택된 개체 안에 있는 모든 요소를 반환합니다.

단어의 뜻대로 선택된 개체 안에 있는 요소의 집합을 반환하며, 이 메서드의 경우 filter 메서드와 조합을 통해 많이 사용을 하고 있습니다.

다음 예제를 통해 간단한 사용법을 살펴 보도록 하겠습니다.

<**html** xmlns="http://www.w3.org/1999/xhtml">

<**head**>

<**title**>jQuery Selector</**title**>

<**link** href="../Styles/Site.css" rel="stylesheet" type="text/css" />

<**style**>

**div**,**pre** { background : #FFF; padding:10px; margin:10px; }

**table** { border:1px solid #AAA; }

**td** { border:1px solid #AAA; width:25px; height:25px; }

</**style**>

<**script src="http://code.jquery.com/jquery-latest.js"**></**script**>

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

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

$("ul").find("li").contents().remove("ul");

});

</**script**>

</**head**>

<**body** style="padding:10px;">

<**h2**>jQuery 시작 Selector</**h2**>

<**p**>jQuery에 대한 자세한 내용을 보려면 jquery.com 을 방문하세요.</**p**>

<**div**>

<**ul**>

<**li**>list item 1</**li**>

<**li**>list item 2

<**ul**>

<**li**>sub item 1</**li**>

<**li**>sub item 2</**li**>

</**ul**>

</**li**>

<**li**>list item 3</**li**>

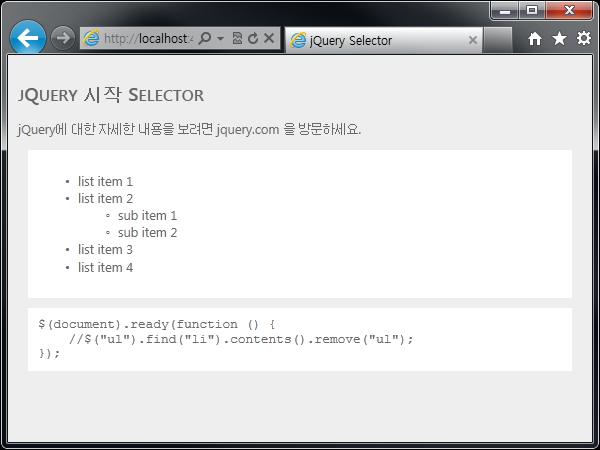
<**li**>list item 4</**li**>

</**ul**>

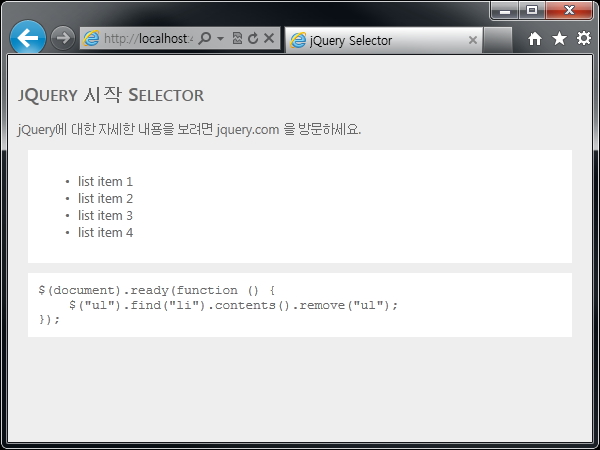
</**div**>

</**body**>

</**html**>



[메서드 적용전의 모습]



[메서드 적용 후의 모습]

remove(selector) 메서드는 "selector"와 일치하는 내용을 삭제합니다.  
예제를 간단히 설명하면, 셀렉터를 통해 ul 요소를 찾은 후 find()를 통해 "li" 요소를 찾습니다. 마지막으로 contents() 메서드를 통해 li 요소에 담겨있는 모든 요소를 찾은 후 remove() 메서드를 통해 "ul"이 제거가 되었습니다.

**.end()**

이 메서드는 정말 중요하고 많은 곳에서 사용이 되며, 실제로 복잡한 스크립트를 작성할 때 마법과 같은 역할을 해 주는 메서드입니다.

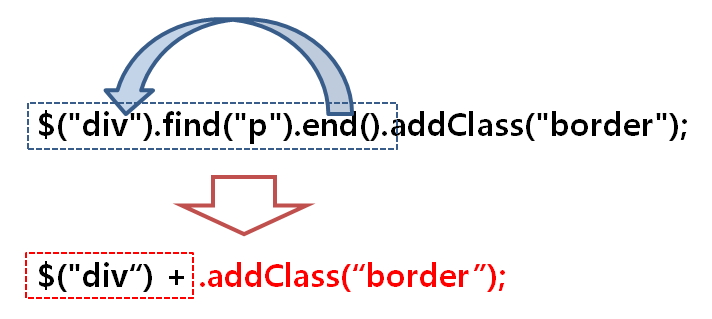
.end() 메서드를 간단히 설명 드리자면 .end() 메서드가 호출 후 바로 이전의 상태로 돌아가는 것을 의미합니다.

역시나 설명이 어렵우시나요 ?

$(div).find(p).end().addClass(“myClass”);와 같은 구문이 있다고 한다면 myClass라는 class 속성이 div에 적용이 됩니다.

.end() 메서드가 없다면 “div” 요소 안에 있는 “p” 요소에 class 속성이 적용이 되지만 .end() 메서드의 호출로 인해 “p” 요소 이전의 개체인 $(“div”) 개체가 다시 호출이 되고 호출된 $(“div)에 class 속성이 적용이 되는 것입니다.

그림으로 다시 설명을 드리자면 아래와 같이 동작을 한다고 보시면 됩니다.



.end() 메서드의 경우 앞에 강좌에서 설명한 jQuery의 특징 중에 플러그인과 메서드체인을 이용한 스크립트 작성에 빠질 수 없는 요소로 정확히 이해를 하신다면 간결하고 클라이언트에 부담 확~ 줄일 수 있는 프로그래밍이 가능하므로 해당 메서드에 대해 정확히 이해를 하시기 바랍니다.