

## About jQuery

### 1) jQuery의 시작

jQuery에서의 노드 접근을 알아보시다.

노드란 가지(branch)라는 의미를 가지고 있습니다.

노드(node)는 JavaScript에서 계층이 되는 요소를 지정할 때 사용되는 것으로, HTML 문서에서의 분류될 수 있는 형식을 의미합니다. 태그(tag)가 될 수도 있고, id, 클래스가 될 수도 있습니다.

다음은 예제에서 공통으로 사용할 HTML입니다.

JavaScript 주석 부분이 jQuery가 작성될 영역입니다.

#### 시작 파일

sample/basic\_jquery/basic\_jquery\_sample.html

#### HTML

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <meta http-equiv="X-UA-Compatible" content="IE=edge">
6 <meta name="viewport" content="width=device-width, initial-scale=1.0,
  minimum-scale=1.0">
7 <title>basic :: jQuery basic study</title>
8 <link rel="stylesheet"
  href="http://fonts.googleapis.com/css?family=Open+Sans:400,300,300italic,400italic,600,6
  00italic,700,700italic,800,800italic">
9 <style>
10 body {
11     margin: 20px;
12     padding: 20px;
13     line-height: 1;
14     font-family: "Open Sans", sans-serif;
15     font-size: 1em;
16     background-color: #555;
17     color: #000;
18 }
```

```

19     ul, li {
20         margin: 0;
21         padding: 0;
22         list-style: none;
23     }
24     li {
25         margin-top: 20px;
26     }
27     .title {
28         margin: 0;
29         padding: 0;
30         font-size: 1.5em;
32         font-weight: 300;
32     }
33     .container {
34         margin-top: 35px;
35     }
36 </style>
37 <script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
38 <script>
39 $(function(){
40     // jQuery가 작성될 예정입니다.
41 });
42 </script>
43 </head>
44
45 <body>
46 <h1 class="title">basic :: jQuery basic study</h1>
47 <div class="container">
48     <ul id="tutorial">
49         <li class="html">HTML</li>
50         <li class="css">CSS</li>
51         <li class="css3">CSS3</li>
52         <li class="javascript">JavaScript</li>
53         <li class="jquery"><p>jQuery</p></li>
54         <li class="blank"></li>
55     </ul>
56     <p id="portfolio">portfolio</p>
57 </div>
58 </body>
59 </html>

```

37행 : <script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>

jQuery 코드를 작성하기 위해서 관련 js를 연결합니다. jQuery CDN의 경로는 다음과 같습니다.

#### 참고 URL

<http://code.jquery.com>

39행 : `$(function(){`

문서가 다 로딩 완료되는 시점에서 실행되는 jQuery 이벤트입니다. 이벤트에 대한 부분은 이벤트 절에서 설명하겠습니다.

웹 페이지는 문서가 다 로드되기 전까지는 JavaScript를 정확히 처리할 수 없습니다.

다시 말해서, 문서가 모두 읽힌 후에야 JavaScript를 위한 준비가 된 것입니다.

문서가 다 로드 되는 시점을 점검하기 위해 아래와 같은 코드를 작성합니다.

```
$("#document").ready(function(){  
});
```

'\$("#document").ready(' 구문을 간단히 '\$('로 바꿀 수도 있습니다.

```
$(function(){  
});
```

#### 이벤트(컴퓨팅)

컴퓨팅에서 이벤트(event)란 프로그램에 의해 감지되고 처리될 수 있는 동작이나 사건을 말한다. 대체로 이벤트는 프로그램 동작 과정과 함께 동시에 처리되도록 되어 있다. 즉 프로그램은 이벤트를 처리하기 위한 하나 이상의 전용 공간(또는 핸들러)를 가지게 되는데, 보통의 경우 이벤트 루프라고 불리는 곳에서 이를 처리하게 된다.

사용자가 키보드의 키를 누르는 것이 가장 대표적인 이벤트 발생 중의 하나이며, 타이머와 같은 하드웨어 장치가 이벤트를 발생시키기도 한다.

출처 : [https://ko.wikipedia.org/wiki/이벤트\\_\(컴퓨팅\)](https://ko.wikipedia.org/wiki/이벤트_(컴퓨팅))

## 1-1) 선택자(Selector)

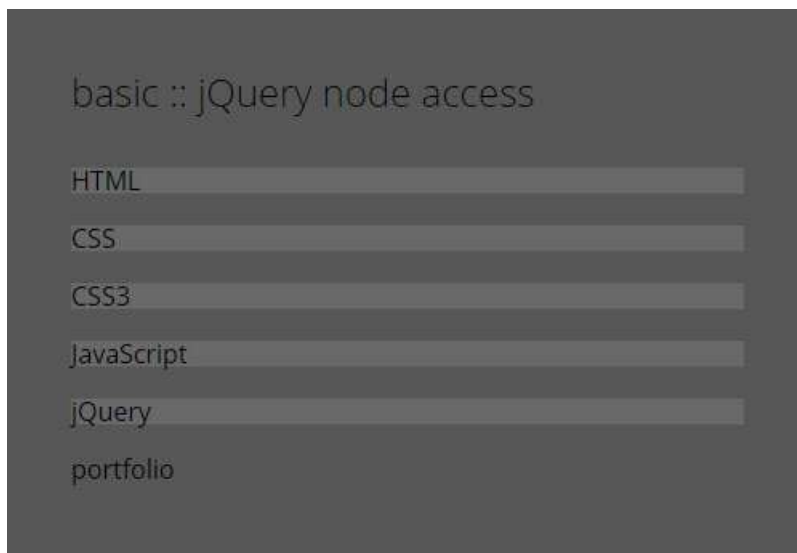
jQuery는 JavaScript에 비해 HTML 문서에서 선택자를 쉽게 선택하여, jQuery 기능을 구현할 수 있습니다.

### 1-1-1) 기본 선택자

종류	예	설명
태그 선택자	\$("#p")	기본 태그 요소가 선택됩니다.
클래스 선택자	\$(".class")	클래스 요소가 선택됩니다.
id 선택자	\$("#id")	id 요소가 선택됩니다.
그룹 선택자	\$("#p, .class, #id")	여러 가지 요소의 형식이 선택됩니다.
전체 선택자	\$("*")	전체가 선택됩니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery1.html



#### JavaScript

```
38 <script>
39 $(function(){
40     $("li").css({ "background-color": "#666" });
```

```
41     });  
42     </script>
```

40행 : `$("li").css({ "background-color": "#666" });`

\$를 이용해서 jQuery 객체를 지정합니다.

`$("li")`는 li 태그 요소를 지정하는 선택자입니다.

`css()`는 선택자의 CSS 속성을 변경할 때 활용되는 명령어입니다.

`css()`명령어를 통해서 li 요소는 Inline Style의 스타일이 추가되는 것을 확인할 수 있습니다.

다음은 예제에서 처리된 후의 Debugger 창에서 확인한 HTML Elements입니다.

```
<ul id="tutorial">  
  <li class="html" style="background-color: rgb(102, 102, 102);">HTML</li>  
  <li class="css" style="background-color: rgb(102, 102, 102);">CSS</li>  
  <li class="css3" style="background-color: rgb(102, 102, 102);">CSS3</li>  
  <li class="javascript" style="background-color: rgb(102, 102, 102);">JavaScript</li>  
  <li class="jquery" style="background-color: rgb(102, 102, 102);">jQuery</li>  
  <li class="blank" style="background-color: rgb(102, 102, 102);"> </li>  
</ul>
```

아래의 표는 li 요소에 대한 스타일 지정 방식입니다.

```
li {  
    background-color: #666;  
}
```

해당 요소의 CSS 속성을 얻어내기 위해 아래와 같이 스크립트 구문만 바꿔보세요.

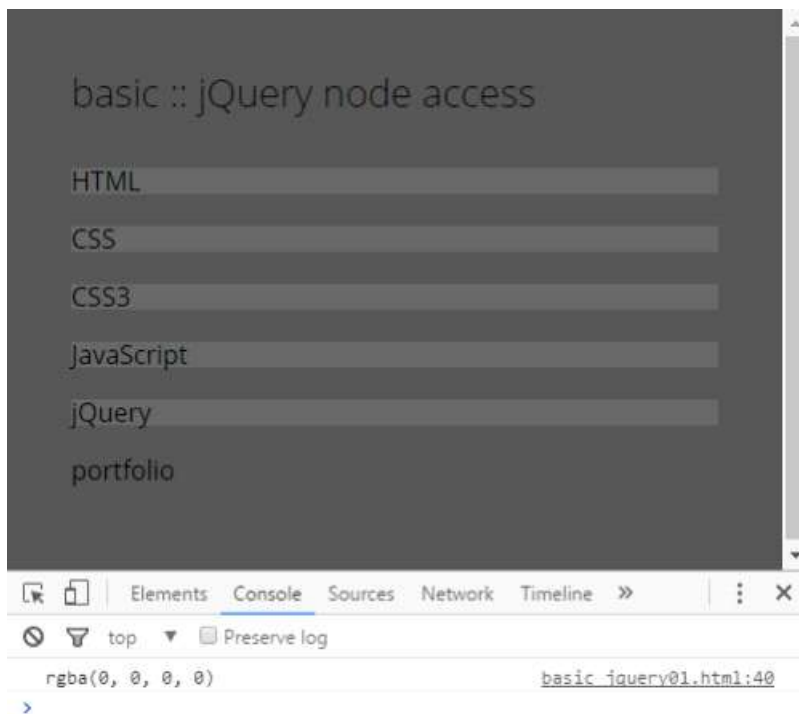
#### JavaScript

```
38 <script>
39 $(function(){
40     console.log($(".li").css("background-color") :', $(".li").css("background-color"));
41     $(".li").css({ "background-color": "#666" });
42 });
43 </script>
```

40행 : `console.log($(".li").css("background-color") :', $(".li").css("background-color"));`  
CSS 속성을 얻기 위해서는 아래와 같은 용법으로 작성합니다.

```
css("속성 이름");
```

Chrome 브라우저의 [검사(N)] 창에서 [Console]을 선택하면 아래와 같은 내용을 확인할 수 있습니다.



li 태그 요소의 배경색으로는 아무 것도 없는 것을 의미합니다.

```
rgba(0, 0, 0, 0)
```

JavaScript 디버깅(Debugging)에 대해 알아보시다.

프로그래밍 언어를 작성하다보면 오류를 확인하기 위한 과정이 반드시 필요합니다.  
이를 디버깅(Debugging)이라고 합니다.

기존에 JavaScript 코드를 디버깅하는 가장 단순한 방법은 alert() 명령어를 사용하여 경고 창을 띄우는 방식입니다. alert()를 사용하는 방법은 훌륭하진 않지만, 간단히 사용할 수 있는 꽤 쓸 만한 디버깅 방법입니다.

다만 일일이 경고 창을 닫아 주어야하는 불편함이 있던 것이 사실입니다. 가끔 for 구문 안에 alert()를 넣고 반복하는 황당한 경험을 할 수도 있을 것입니다. 경험상 디버깅 시에 기록 파일인 로그(log)에 필요한 정보를 기록하게 되면 디버깅이 훨씬 쉬워집니다. 다른 프로그래밍 언어들도 이러한 기능들을 지원하고 있습니다.

JavaScript에서 기록을 위한 객체는 console 객체입니다. console 객체는 Internet Explorer 8 이상 버전과 Safari, Chrome, Opera 같은 최신 브라우저에서 사용할 수 있습니다.

예제의 코드를 바꾸어 보겠습니다.

#### JavaScript

```
38 <script>
39 $(function(){
40     $("li").css({ "background-color": "#666" });
41     console.log('$("li").css("background-color") :', $("li").css("background-color"));
42 });
43 </script>
```

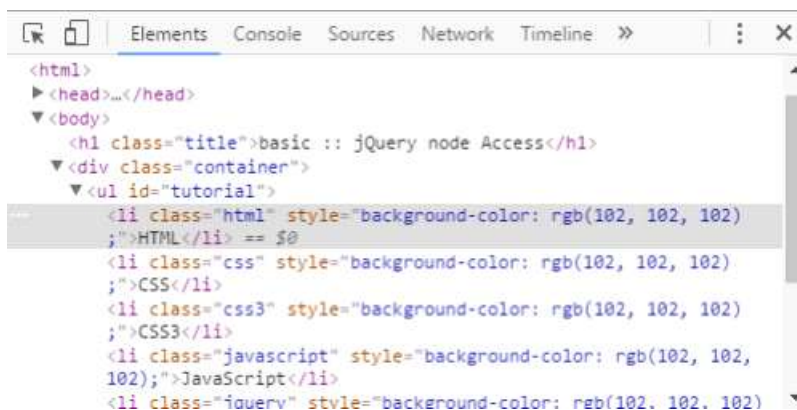
코드의 순서를 바꾸었습니다. 다시 console 로그를 확인해 봅시다.

```
rgb(102, 102, 102)
```

rgb(102, 102, 102) 코드는 16진수 코드로 환산하면 #666 색상이 됩니다.

\$("li").css({"background-color": "#666"}); 방식으로 적용한 후에 console 로그를 확인해서 다음과 같은 결과를 확인할 수 있었습니다.

[검사(N)] 창에서 해당 요소를 확인해 보면 다음과 같습니다.



css() 명령어로 적용된 스타일은 Inline Style로 적용되는 것을 확인할 수 있습니다.

CSS의 속성이 두 단어의 결합 형태일 경우에는 문자열 형태를 적용하기 위해 큰 따옴표나 작은따옴표를 사용합니다. CSS 스타일 속성 중에서 background-color, background-image, background-repeat과 같은 속성들입니다.



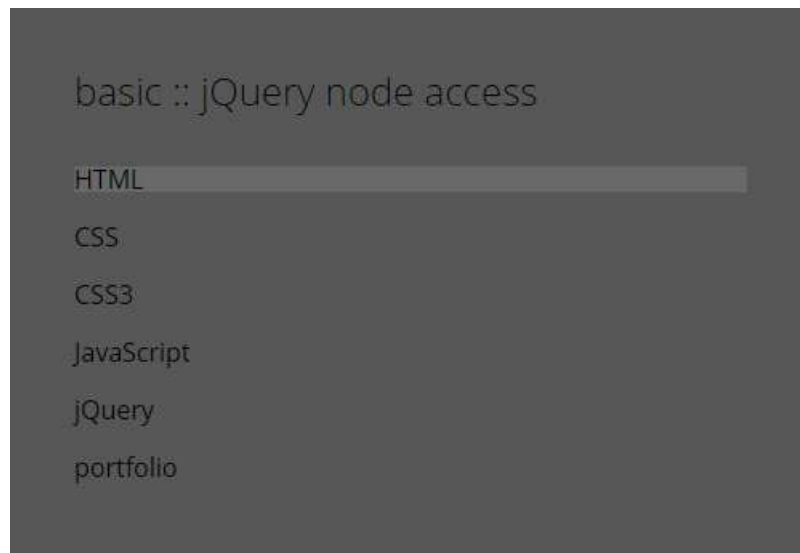
다음과 같은 용법으로 작성하면 됩니다.

```
$("#li").css("background-color", "#666");  
$("#li").css({ "background-color": "#666" });
```

`$("#li").css({ "background-color": "#666" });` 방식은 두 개 이상의 속성을 `css()` 명령어로 적용할 때 유용합니다. 두 개 이상의 속성을 적용한 예시입니다.

```
$("#li").css({ "background-color": "#666", color: "#fff" });
```

클래스 선택자에 대해 알아보시다.



#### JavaScript

```
38 <script>
39 $(function(){
40     $(".html").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$(".html").css({ "background-color": "#666" });

html 클래스 요소가 선택됩니다.

id 선택자에 대해 알아보시다.

basic :: jQuery node access

HTML

CSS

CSS3

JavaScript

jQuery

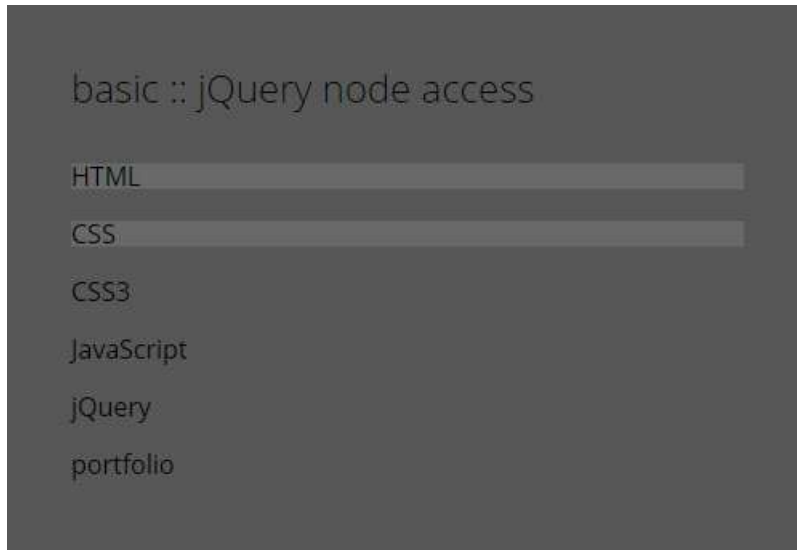
portfolio

#### JavaScript

```
38 <script>
39 $(function(){
40     $("#tutorial").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$("#tutorial").css({ "background-color": "#666" });  
tutorial id 요소가 선택됩니다.

그룹 선택자에 대해 알아보시다.



#### JavaScript

```
38 <script>
39 $(function(){
40     $(".html, .css").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$(".html, .css").css({ "background-color": "#666" });

html 클래스 요소와 css 클래스 요소가 선택됩니다.

CSS에서도 같은 방식으로 선택합니다.

```
.html, .css {
    background-color: #666;
}
```

전체 선택자에 대해 알아보시다.

basic :: jQuery node access

HTML

CSS

CSS3

JavaScript

jQuery

portfolio

#### JavaScript

```
38 <script>
39 $(function(){
40     $("").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$("").css({ "background-color": "#666" });

전체 영역이 선택됩니다.

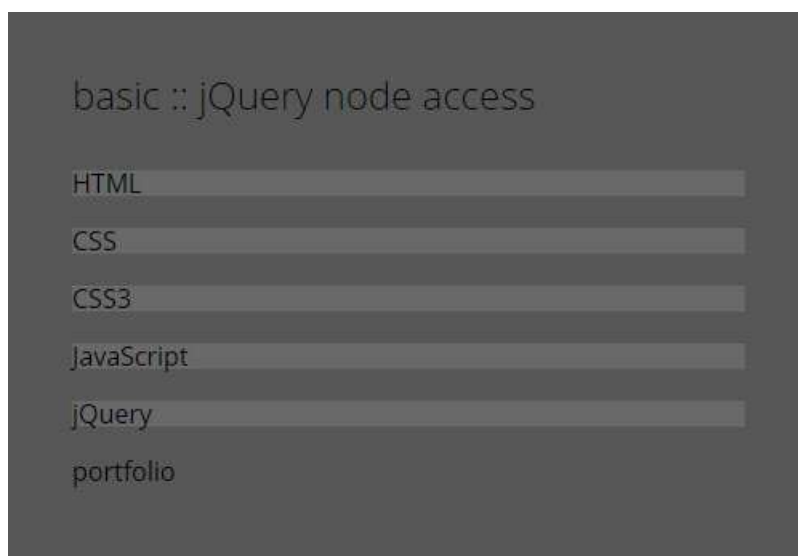
### 1-1-2) 계층 선택자

계층적인 노드 구조에서 선택하는 방법입니다. 이는 CSS에서 요소를 선택하는 방식과 유사합니다.

종류	예	설명
하위 선택자	<code>\$("div li")</code>	하위 노드가 모두 선택됩니다.
child 선택자	<code>\$("div &gt; p")</code>	기준이 되는 노드 위치에서 바로 아래 노드가 선택됩니다. 이를 자식 노드라고 호칭합니다.
sibling 선택자	<code>\$("div + p")</code>	기준이 되는 노드 다음 노드가 선택됩니다.
siblings 선택자	<code>\$("div ~ li")</code>	기준이 되는 노드 다음 모든 노드가 선택됩니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery2.html



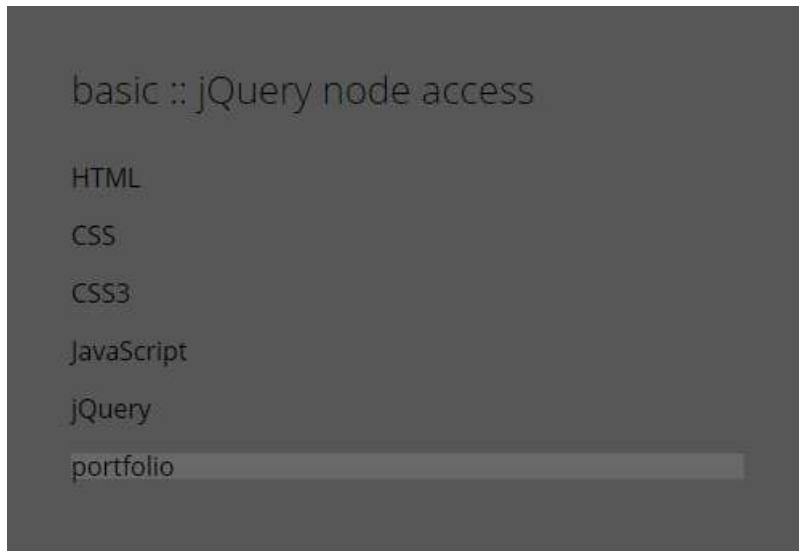
#### JavaScript

```
38 <script>
39 $(function(){
40     $(".container li").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : `$(".container li").css({ "background-color": "#666" });`

container 클래스 하위의 li 요소가 선택됩니다.

child 선택자(>)에 대해 알아보시다.



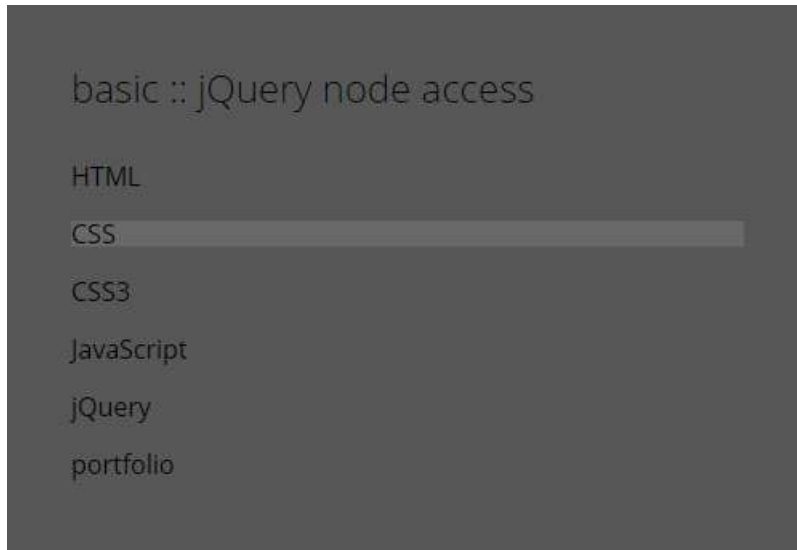
#### JavaScript

```
38 <script>
39 $(function(){
40     $(".container > p").css({ "background-color": "#666", color: "#fff" });
41 });
42 </script>
```

40행 : \$(".container > p").css({ "background-color": "#666", color: "#fff" });

container 클래스 바로 아래의 p 요소가 선택됩니다. 바로 아래에 위치하는 노드를 자식 노드(children)라고 합니다.

sibling(+) 선택자에 대해 알아보시다.



#### JavaScript

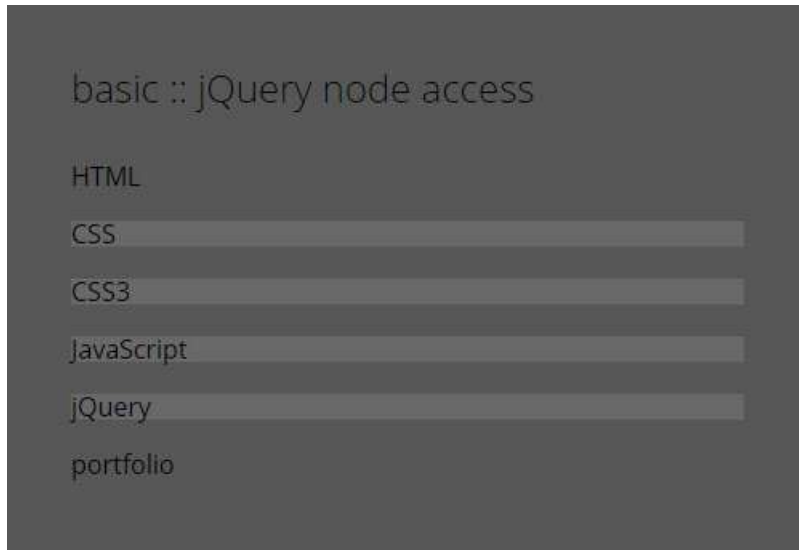
```
38 <script>
39 $(function(){
40     $(".html + li").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$(".html + li").css({ "background-color": "#666" });

html 클래스 다음에 해당하는 노드가 선택됩니다. 따라서 css 클래스 요소가 선택됩니다.



siblings(~) 선택자에 대해 알아보시다.



#### JavaScript

```
38 <script>
39 $(function(){
40     $(".html ~ li").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$(".html ~ li").css({ "background-color": "#666" });

html 클래스 다음에 해당하는 모든 노드가 선택됩니다. .css, .css3, .javascript, .jquery 요소가 선택됩니다.

### 1-1-3) 속성 선택자

요소의 속성을 사용하여 선택하는 방법입니다.

종류	예	설명
[name="value"]	\$("#id='#tutorial'")	아이디가 '#tutorial'과 일치하는 요소가 선택되는 방법입니다.
[name^="value"]	\$("#[class^='css']")	클래스가 'css'로 시작하는 요소가 선택되는 방법입니다.
[name\$="value"]	\$("#[css\$='3']")	클래스가 '3'으로 끝나는 요소가 선택되는 방법입니다.
[name*="value"]	\$("#[class*='as']")	클래스가 'as'가 포함되어 있는 요소가 선택되는 방법입니다.
[name!="value"]	\$("#img[src!='.png']")	'png'와 일치하지 않는 img 요소가 선택되는 방법입니다.
[name="value"][name="value"]	\$("#[src][alt]")	'src' 속성과 'alt' 속성이 있는 모든 요소가 선택되는 방법입니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery3.html

basic :: jQuery node access

HTML

CSS

CSS3

JavaScript

jQuery

portfolio

## JavaScript

```
38 <script>
39 $(function(){
40     $("li[class='html']").css({ "background-color": "#666" });
41 });
42 </script>
```

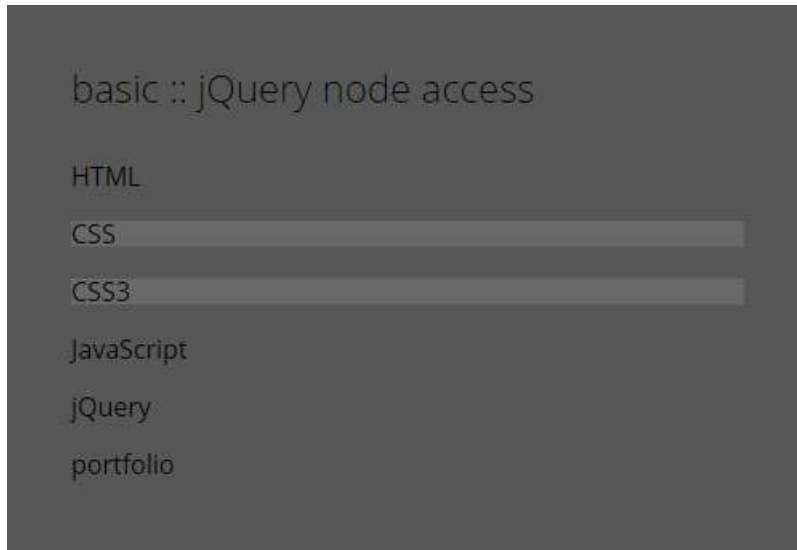
40행 : `$("li[class='html']").css({ "background-color": "#666" });`

li 요소 중에서 class 속성이 'html'인 요소가 선택됩니다.

이전에 적용된 선택자 방식과 동일합니다.

```
$(".html").css({ "background-color": "#666" });
```

^= 선택자에 대해 알아보시다.



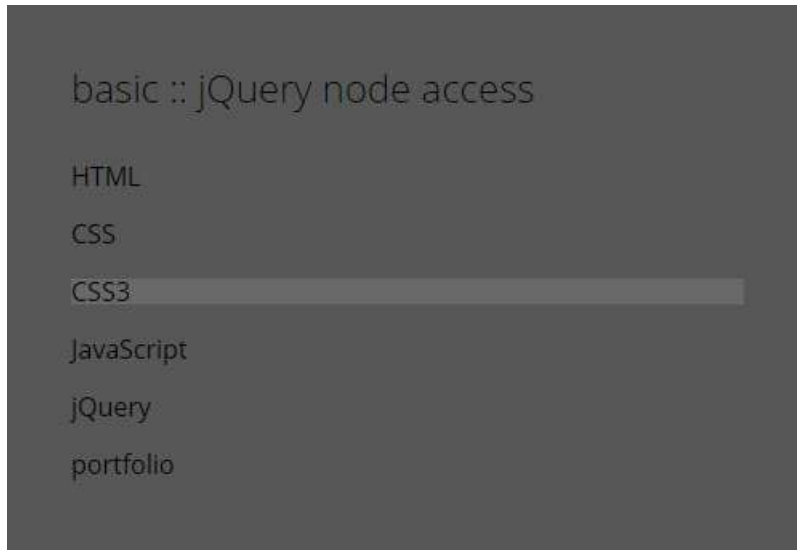
#### JavaScript

```
38 <script>
39 $(function(){
40     $("li[class^='css']").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : `$("li[class^='css']").css({ "background-color": "#666" });`

li 요소 중에서 class 속성 값이 'css'로 시작하는 요소가 선택됩니다. 시작 문자열을 지정하는 방식입니다.

\$= 선택자에 대해 알아보시다.



#### JavaScript

```
38 <script>
39 $(function(){
40     $("li[class$='3']").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : `$("li[class$='3']").css({ "background-color": "#666" });`

li 요소 중에서 class 속성 값이 '3'으로 끝나는 요소가 선택됩니다. 끝 문자열을 지정하는 방식입니다.

\*= 선택자에 대해 알아보시다.

basic :: jQuery node access

HTML

CSS

CSS3

JavaScript

jQuery

portfolio

#### JavaScript

```
38 <script>
39 $(function(){
40     $("li[class*='as']").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$("li[class\*='as']").css({ "background-color": "#666" });

li 요소 중에서 class 속성 값이 'as'가 포함되어있는 요소가 선택됩니다. 포함된 문자열을 지정하는 방식입니다.

속성이 있는 요소를 선택하는 방식을 알아보시다.

basic :: jQuery node access

HTML

CSS

CSS3

JavaScript

jQuery

portfolio

#### JavaScript

```
38 <script>
39 $(function(){
40     $("p[id]").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$("p[id]").css({ "background-color" : "#666" });

p 요소 중에서 id 속성이 있는 요소가 선택됩니다.

#### 1-1-4) 기본 필터 선택자

필터는 선택자 앞에 :(콜론)을 붙여 표현을 하며, 필터를 사용해 다양한 조건에 따라 선택될 수 있습니다.

종류	설명
:animated	show(), hide(), slideDown(), slideUp() 등의 애니메이션 요소를 찾아내는 필터입니다.
:eq(index)	선택된 요소들을 인덱스(index)로 찾을 수 있는 필터입니다.
:gt(index)	선택된 집합에서 인덱스(index)보다 큰 번호를 가지고 있는 요소들이 선택됩니다.
:lt(index)	선택된 집합에서 인덱스(index)보다 작은 번호를 가지고 있는 요소들이 선택됩니다.
:header	제목 요소(h1, h2, h3, h4, h5, h6)들이 선택되는 필터입니다.
:first	선택된 집합 중에서 첫 번째 요소를 찾는 필터입니다.
:last	선택된 집합 중에서 마지막 요소를 찾는 필터입니다.
:odd	선택된 집합 중에서 홀수인 요소를 찾는 필터입니다.
:even	선택된 집합 중에서 짝수인 요소를 찾는 필터입니다.
:not()	현재 선택한 집합의 반대 집합이 선택되는 필터입니다.



## 시작 파일

sample/basic\_jquery/start/basic\_jquery4.html

basic :: jQuery node access

HTML

CSS

CSS3

JavaScript

jQuery

portfolio

## JavaScript

```
38 <script>
39 $(function(){
40     $(".container li:eq(2)").css({ "background-color": "#666" });
41 });
42 </script>
```

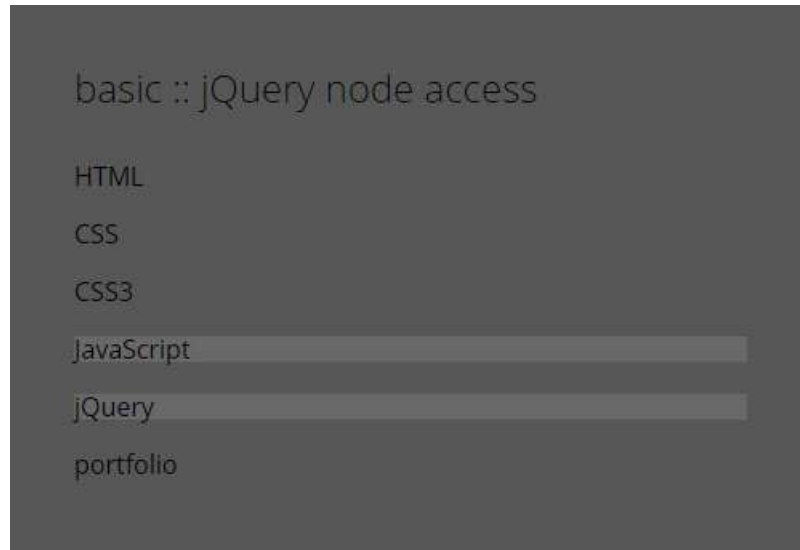
40행 : \$(".container li:eq(2)").css({ "background-color": "#666" });

container 클래스 하위의 li 요소 중에서 인덱스(index)가 2인 요소가 선택됩니다. 인덱스는 0에서 시작되므로 li:eq(2)은 3번째 요소가 선택됩니다.

이는 CSS의 기법인 nth-child()를 활용한 방법과 유사합니다.

```
$(".container li:nth-child(3)").css({ "background-color": "#666" });
```

이번에는 :gt() 필터에 대해 알아보시다.



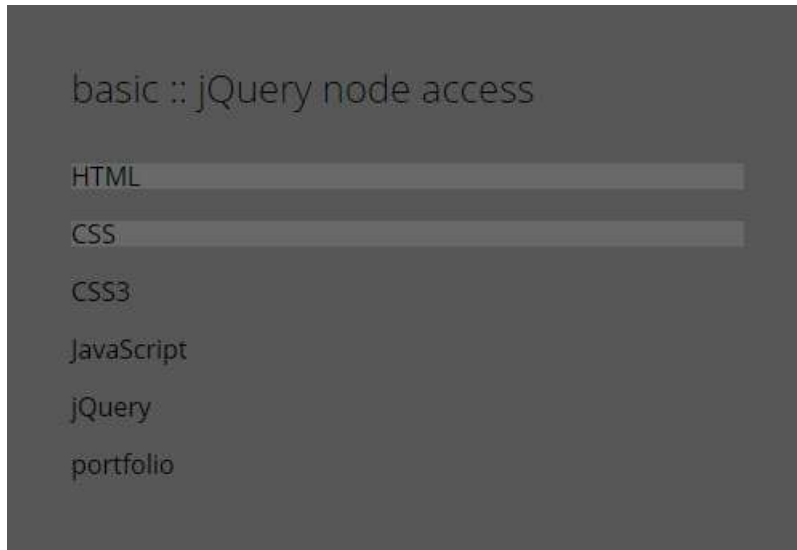
#### JavaScript

```
38 <script>
39 $(function(){
40     $(".container li:gt(2)").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$(".container li:gt(2)").css({ "background-color": "#666", color: "#fff" });

container 클래스 하위의 li 요소 중에서 인덱스가 2인 요소를 기준으로 다음 요소가 선택됩니다. 결국, li:eq(3), li:eq(4) 요소가 선택됩니다. gt는 'greater than'의 약자입니다.

:lt() 필터에 대해 알아보시다.



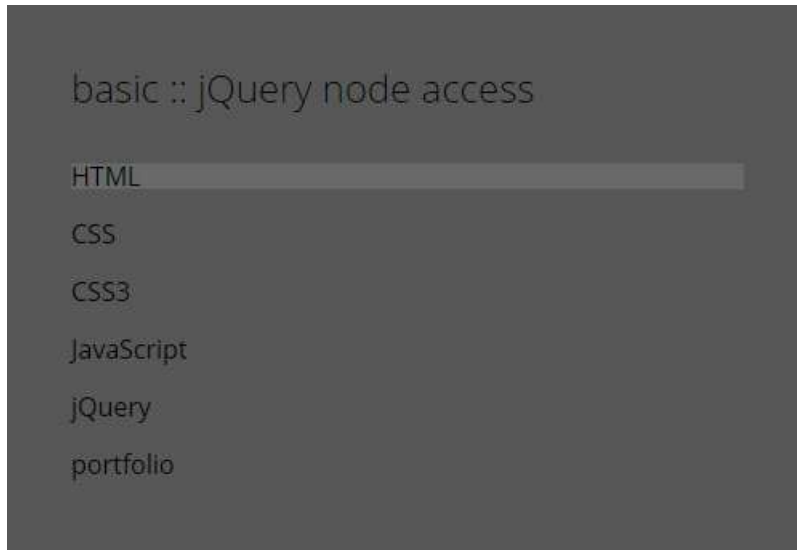
#### JavaScript

```
38 <script>
39 $(function(){
40     $(".container li:lt(2)").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$(".container li:lt(2)").css({ "background-color": "#666" });

container 클래스 하위의 li 요소 중에서 인덱스가 2인 요소를 기준으로 이전 요소가 선택됩니다. 결국, li:eq(0), li:eq(1) 요소가 선택됩니다. lt는 'less than'의 약자입니다.

:first 필터에 대해 알아보시다.



#### JavaScript

```
38 <script>
39 $(function(){
40     $(".container li:first").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$(".container li:first").css({ "background-color": "#666" });  
container 클래스 하위의 li 요소 중에서 첫 번째가 선택됩니다.

반대로 :last 필터는 마지막 요소가 선택됩니다.

```
$(".container li:last").css({ "background-color": "#666" });
```

하지만 :last 필터로 지정된 방식은 예제에서 확인할 수 없습니다. li의 마지막 요소는 empty 클래스로서 내용이 비어있기 때문입니다.

:header 필터에 대해 알아보시다.

basic :: jQuery node access

HTML

CSS

CSS3

JavaScript

jQuery

portfolio

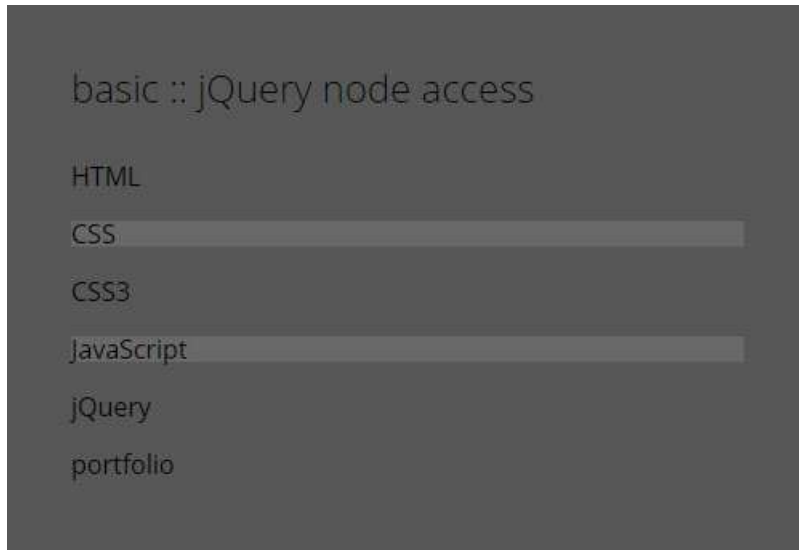
#### JavaScript

```
38 <script>
39 $(function(){
40     $(".header").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$(".header").css({ "background-color": "#666" });

제목 요소인 h2 요소가 선택됩니다.

:odd 필터에 대해 알아보시다.



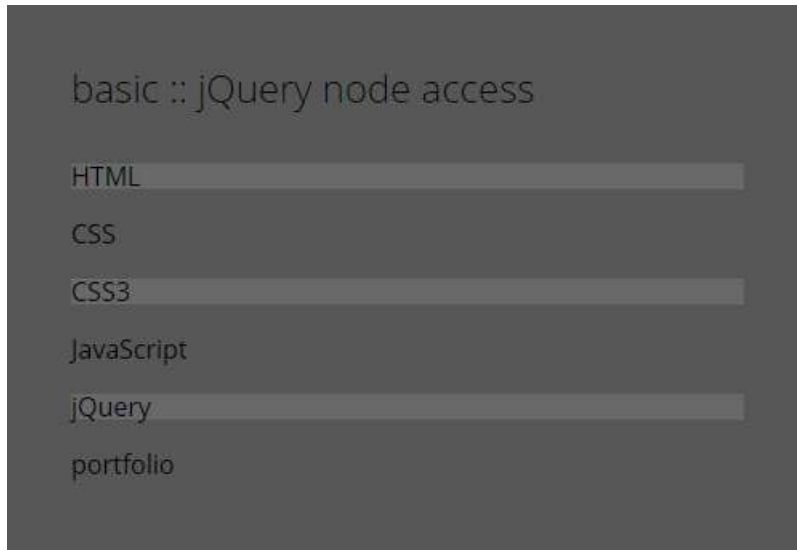
#### JavaScript

```
38 <script>
39 $(function(){
40     $(".container li:odd").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$(".container li:odd").css({ "background-color": "#666" });

container 클래스 하위의 li 요소 중에서 index가 홀수인 li 요소가 선택됩니다.

이번에는 :even 필터에 대해 알아보시다.

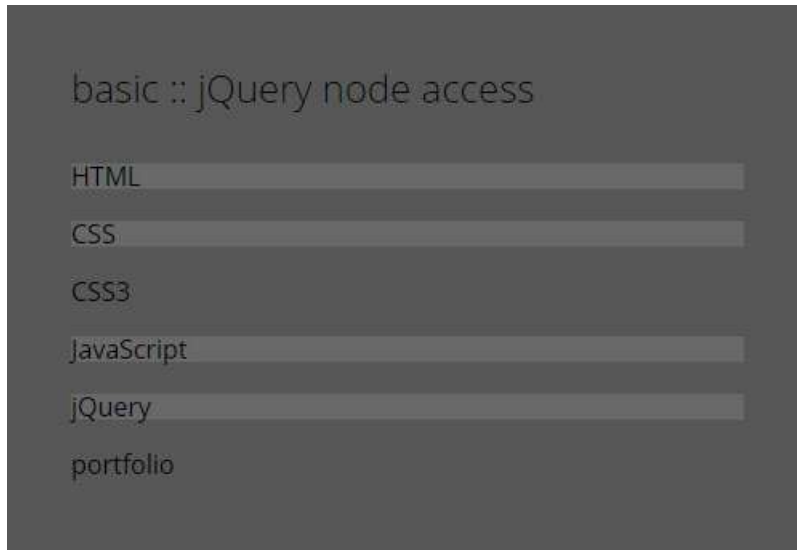


#### JavaScript

```
38 <script>
39 $(function(){
40     $(".container li:even").css({ "background-color": "#666", color: "#fff" });
41 });
42 </script>
```

40행 : \$(".container li:even").css({ "background-color": "#666", color: "#fff" });  
container 클래스 하위의 li 요소 중에서 index가 짝수인 li 요소가 선택됩니다.

:not() 필터에 대해 알아보시다.



#### JavaScript

```
38 <script>
39 $(function(){
40     $(".container li:not(:eq(2))).css({ "background-color": "#666", color: "#fff" });
41 });
42 </script>
```

40행 : \$(".container li:not(:eq(2))).css({ "background-color": "#666" });

container 클래스 하위의 li 요소 중에서 index가 2인 요소를 제외한 li 요소가 선택됩니다.



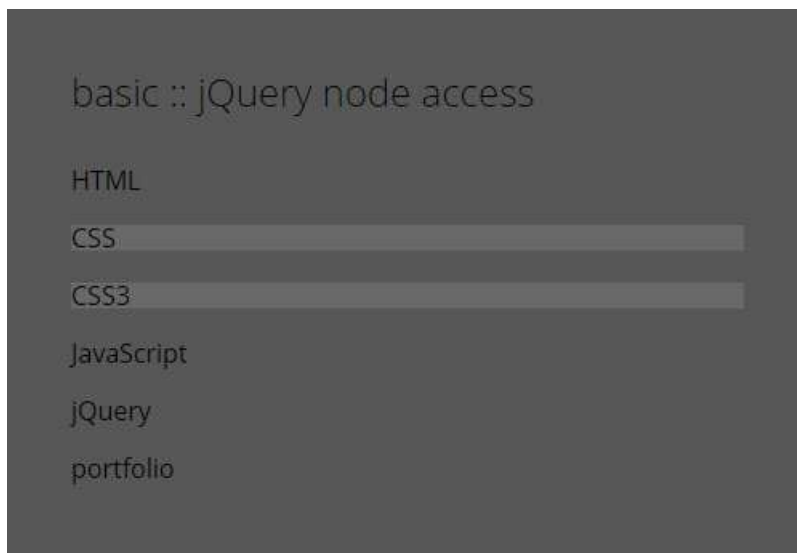
#### 1-1-5) 내용 필터 선택자

콘텐츠와 관련된 요소가 선택되는 방법입니다.

종류	설명
:contains()	() 안의 텍스트와 일치하는 문자열이 요소의 내용 중에 있을 때 선택됩니다.
:empty	요소에 텍스트 없을 때 선택됩니다.
:has()	요소 내부에서 찾고 싶은 요소를 하위 노드까지 살펴본 후 요소가 있으면 선택됩니다.
:parent	:empty와 반대로 요소에 상위 노드가 존재할 때에 선택됩니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery5.html



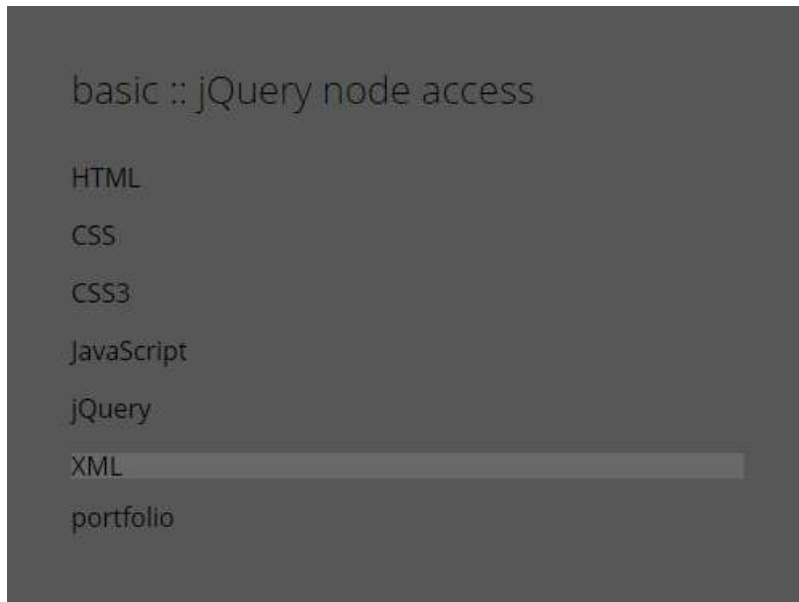
#### JavaScript

```
38 <script>
39 $(function(){
40     $(".container li:contains('CSS')").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$(".container li:contains('CSS')").css({ "background-color": "#666" });

container 클래스 하위의 li 요소 중에서 'CSS' 문자를 가지고 있는 요소가 선택됩니다.

:empty 필터에 대해 알아보시다.



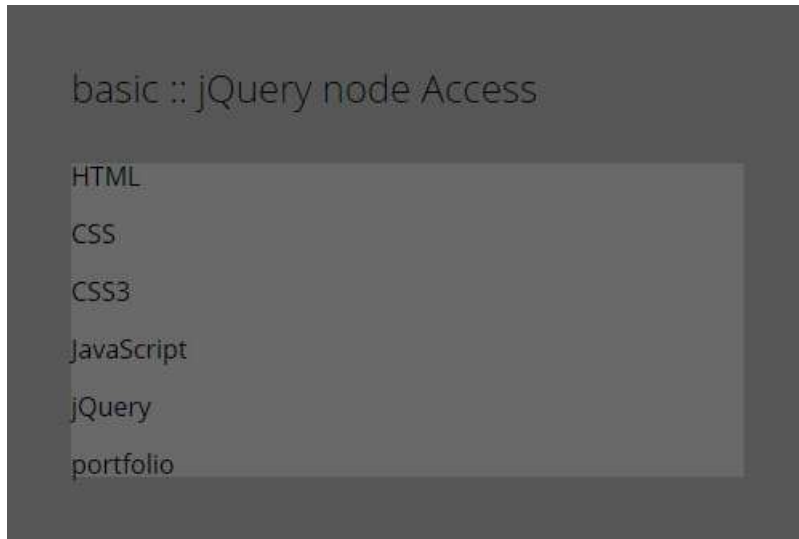
#### JavaScript

```
38 <script>
39 $(function(){
40     $(".container li:empty").text("XML").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$(".container li:empty").text("XML").css({ "background-color": "#666" });

container 클래스 하위의 li 요소 중에서 텍스트가 없는 요소가 선택됩니다. 비어있는 요소를 선택하고 text() 명령어로 'XML' 텍스트를 작성하고 스타일을 변경합니다.

:has() 필터에 대해 알아보시다.



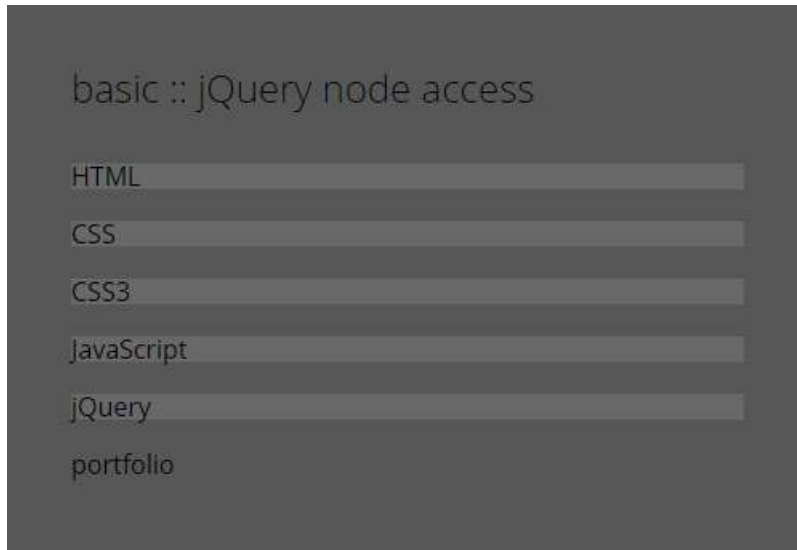
#### JavaScript

```
38     <script>
39     $(function(){
40         $(".container:has(p)").css({ "background-color": "#666" });
41     });
42     </script>
```

40행 : \$(".container:has(p)").css({ "background-color": "#666" });

container 클래스 하위에 p 요소가 있을 경우, container 클래스가 선택됩니다.

:parent 필터에 대해 알아보시다.



#### JavaScript

```
38 <script>
39 $(function(){
40     $(".container li:parent").css({ "background-color": "#666", color: "#fff" });
41 });
42 </script>
```

40행 : \$(".container li:parent").css({ "background-color": "#666" });

container 클래스 하위의 li 요소 중에서 상위 노드가 있는 li 요소가 선택됩니다. 결국 모든 li 요소가 선택됩니다.

#### 1-1-6) 보임 필터 선택자

화면에 보이거나 보이지 않는 요소가 선택됩니다.

컨텐츠 영역을 보이지 않게 하는 방법은 아래와 같습니다. :

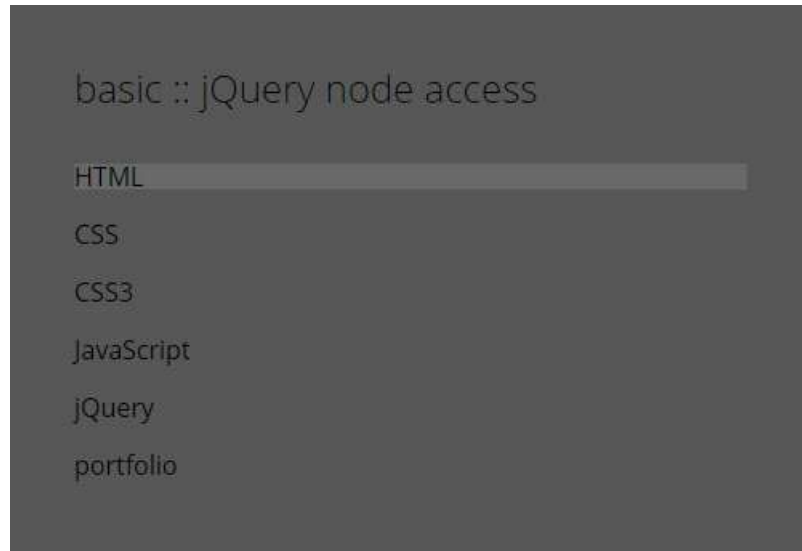
- 스타일 적용 : display: none;
- 스타일 적용 : width: 0; height: 0;
- form 요소 중에 type 속성이 'hidden'인 경우 : type="hidden"
- 부모 요소가 보이지 않거나 숨겨져 있을 경우

visibility: hidden이나 opacity: 0;이 적용된 요소는 위치가 제거되지 않기 때문에 :hidden 선택자에 의해 선택되지 않습니다.

종류	설명
:hidden	보이지 않는 요소가 선택됩니다.
:visible	보이는 요소가 선택됩니다.

## 시작 파일

sample/basic\_jquery/start/basic\_jquery6.html



## JavaScript

```
38 <script>
39 $(function(){
40     $(".html").css({ display: "none" });
41     $(".container li:hidden").css({ display: "block", "background-color": "#666" });
42 });
43 </script>
```

40행 : \$(".html").css({ display: "none" });

html 클래스 요소를 보이지 않도록 스타일합니다.

41행 : \$(".container li:hidden").css({ display: "block", "background-color": "#666" });

container 클래스 하위의 li 요소 중에서 보이지 않는 요소가 선택됩니다.

선택된 요소는 보이도록 스타일하며 배경색을 바꿉니다.

### 1-1-7) 자식 요소 필터 선택자

자식 요소들이 선택되는 방법입니다. CSS에서 활용하는 선택자와 유사합니다.

종류	설명
:first-child	첫 번째 자식 요소가 선택됩니다.
:last-child	마지막 자식 요소가 선택됩니다.
:nth-child(n)	n 순번에 해당하는 자식 요소가 선택됩니다.
:nth-child(even)	짝수 번째에 있는 자식 요소가 선택됩니다.
:nth-child(odd)	홀수 번째에 있는 자식 요소가 선택됩니다.
:nth-child(2n)	2배수 번째에 있는 자식 요소가 선택됩니다.
:nth-child(2n+1)	2배수+1 번째에 있는 자식 요소가 선택됩니다.
:only-child	자식 요소가 오직 하나인 요소가 선택됩니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery7.html

basic :: jQuery node access

HTML

CSS

CSS3

JavaScript

jQuery

portfolio

#### JavaScript

38

<script>

```
39 $(function(){
40     $("li:first-child").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : `$("li:first-child").css({ "background-color": "#666" });`

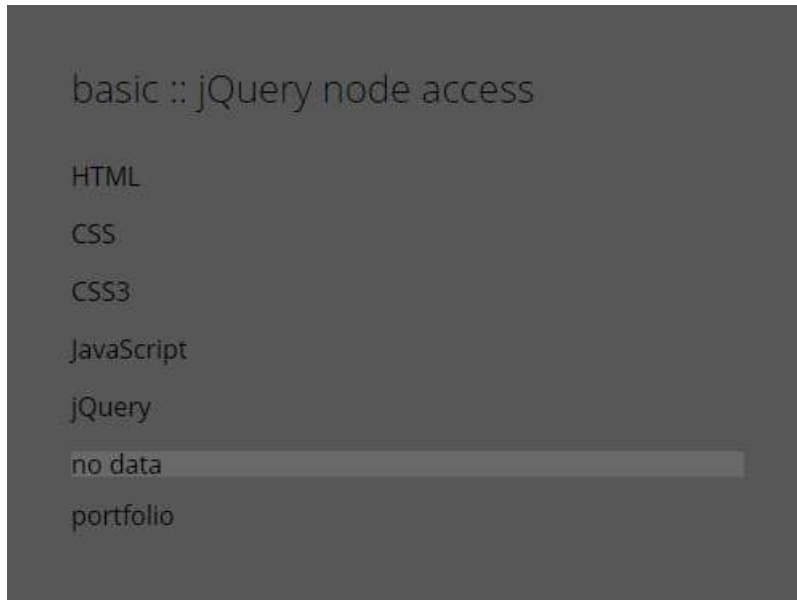
container 클래스 하위의 li 요소 중에서 첫 번째 자식 요소가 선택됩니다.

이는 `:first` 필터 선택자와 같습니다.

```
$("li:first").css({ "background-color": "#666" });
```



이번에는 :last-child 자식 선택 필터에 대해 알아보시다.



#### JavaScript

```
38 <script>
39 $(function(){
40     $("li:last-child").text("no data").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$("li:last-child").text("no data").css({ "background-color": "#666" });

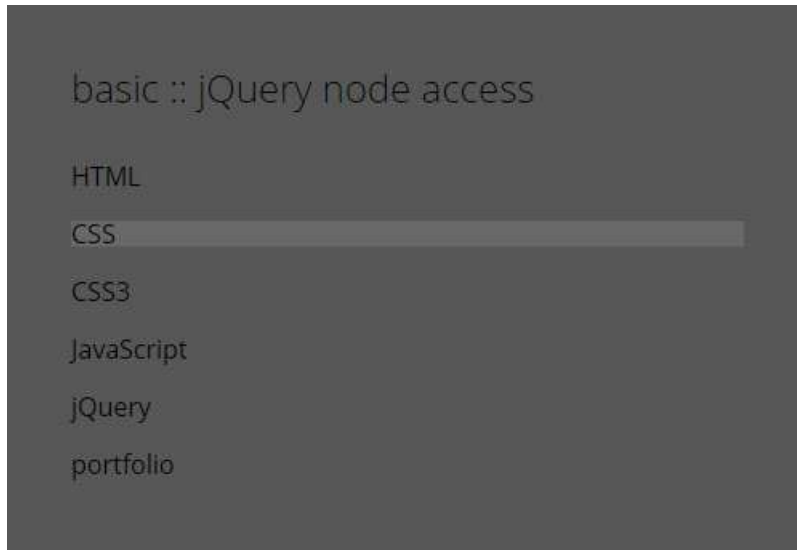
container 클래스 하위의 li 요소 중에서 마지막 자식 요소가 선택됩니다.

마지막 li 요소는 실제로 비어있는 영역이므로 'no data' 텍스트를 작성하고 스타일을 바꾸어 줍니다.

이는 :last 필터 선택자와 같습니다.

```
$("li:last").css({ "background-color": "#666" });
```

:nth-child() 자식 선택 필터에 대해 알아보시다.



#### JavaScript

```
38 <script>
39 $(function(){
40     $("li:nth-child(2)").css({ "background-color": "#666" });
41 });
42 </script>
```

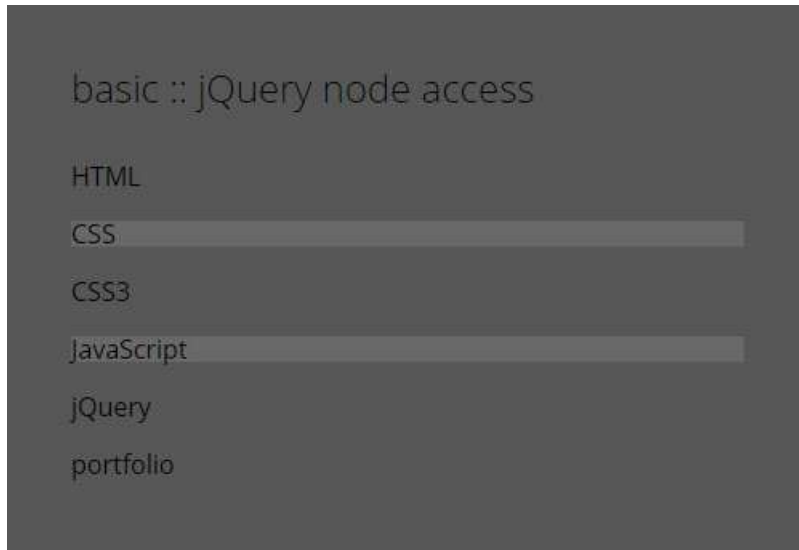
40행 : `$("li:nth-child(2)").css({ "background-color": "#666" });`

container 클래스 하위의 li 요소 중에서 2번째 자식 요소가 선택됩니다.

이는 :eq() 기본 필터 선택자와 용법이 유사합니다. 하지만 :eq() 필터의 첫 시작은 0인데 반해, :nth-child() 필터는 첫 시작이 1입니다. 따라서 :nth-child() 필터를 활용한 코드는 아래의 표와 같습니다.

```
$("li:eq(1)").css({ "background-color": "#666" });
```

:nth-child(even) 자식 필터 선택자에 대해 알아보시다.



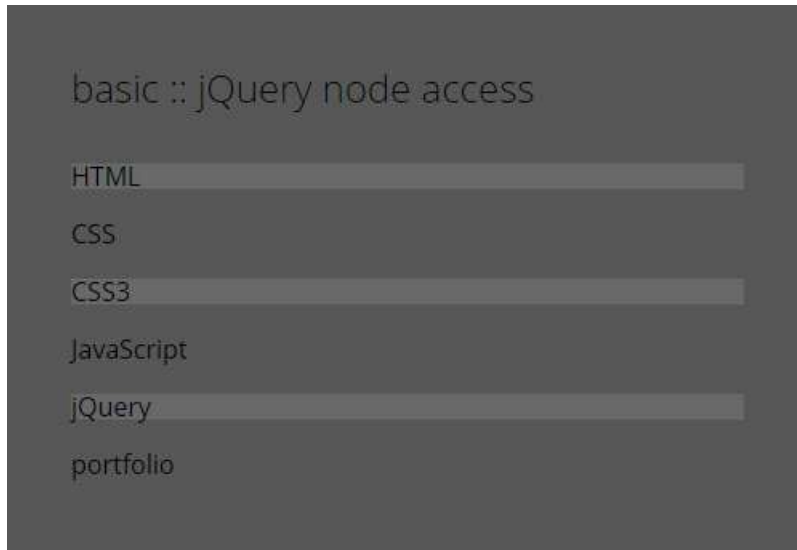
#### JavaScript

```
38 <script>
39 $(function(){
40     $("li:nth-child(even)").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : `$("li:nth-child(even)").css({ "background-color": "#666" });`

container 클래스 하위의 li 요소 중에서 짝수 번째에 있는 자식 요소가 선택됩니다.

:nth-child(odd) 자식 필터 선택자에 대해 알아보시다.



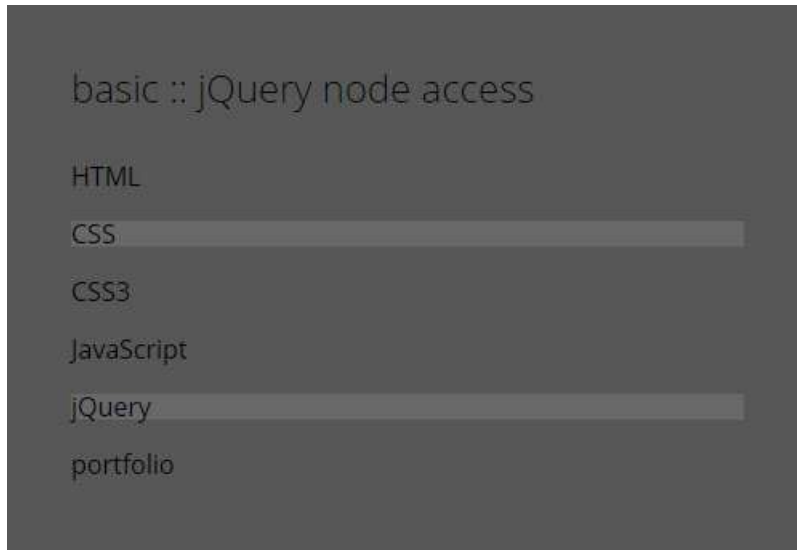
#### JavaScript

```
38 <script>
39 $(function(){
40     $("li:nth-child(odd)").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$("li:nth-child(odd)").css({ "background-color": "#666" });

container 클래스 하위의 li 요소 중에서 홀수 번째에 있는 자식 요소가 선택됩니다.

마지막으로 :nth-child(n+1) 자식 필터 선택자에 대해 알아보시다.



#### JavaScript

```
38 <script>
39 $(function(){
40     $("li:nth-child(3n+2)").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$("li:nth-child(3n+2)").css({ "background-color": "#666" });

container 클래스 하위의 li 요소 중에서 3배수+2 번째에 있는 자식 요소가 선택됩니다. 결국, 2번째와 5번째 li 요소가 선택됩니다.

## 1-2) 탐색(Traversing)

탐색에 나열되어 있는 jQuery 명령어들은 jQuery의 선택자를 기준으로 하여 새로운 요소들을 찾아가는 방법을 제공합니다.

### 1-2-1) 트리 구조 탐색

트리 구조를 기반으로 찾아가는 방식입니다.

| 명령어            | 설명                                 |
|----------------|------------------------------------|
| children()     | 선택한 요소의 모든 자식 요소가 선택됩니다.           |
| find()         | 선택한 요소의 하위 요소 중 조건에 맞는 요소가 선택됩니다.  |
| next()         | 선택한 요소의 다음 요소가 선택됩니다.              |
| nextAll()      | 선택한 요소의 다음 모든 요소가 선택됩니다.           |
| parent()       | 선택한 요소의 부모 요소가 선택됩니다.              |
| parents()      | 선택한 요소의 모든 부모 요소가 선택됩니다.           |
| prev()         | 선택한 요소의 이전 요소가 선택됩니다.              |
| prevAll()      | 선택한 요소의 이전 모든 요소가 선택됩니다.           |
| closest()      | 선택한 요소를 포함하면서 가장 가까운 상위 요소가 선택됩니다. |
| nextUntil()    | 다음에 위치한 요소를 조건에 맞을 때까지 찾습니다.       |
| parentsUntil() | 조건이 참이 될 때까지 부모 요소를 찾습니다.          |
| prevUntil()    | 이전에 위치한 요소를 조건에 맞을 때까지 찾습니다.       |
| siblings()     | 형제 요소를 모두 찾습니다.                    |

## 시작 파일

sample/basic\_jquery/start/basic\_jquery8.html

basic :: jQuery node access

HTML

CSS

CSS3

JavaScript

jQuery

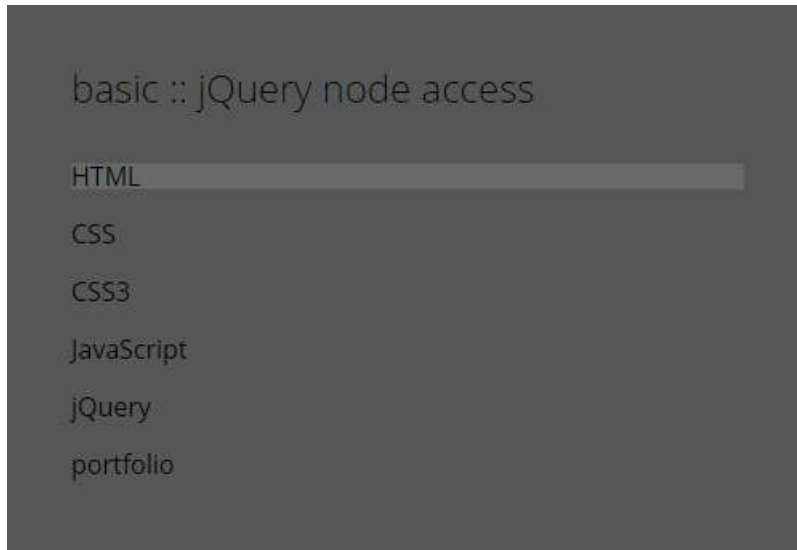
portfolio

## JavaScript

```
38 <script>
39 $(function(){
40     $(".container").children("p").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$(".container").children("p").css({ "background-color": "#666" });  
container 클래스의 자식 p 요소가 선택됩니다.

find() 탐색에 대해 알아보시다.



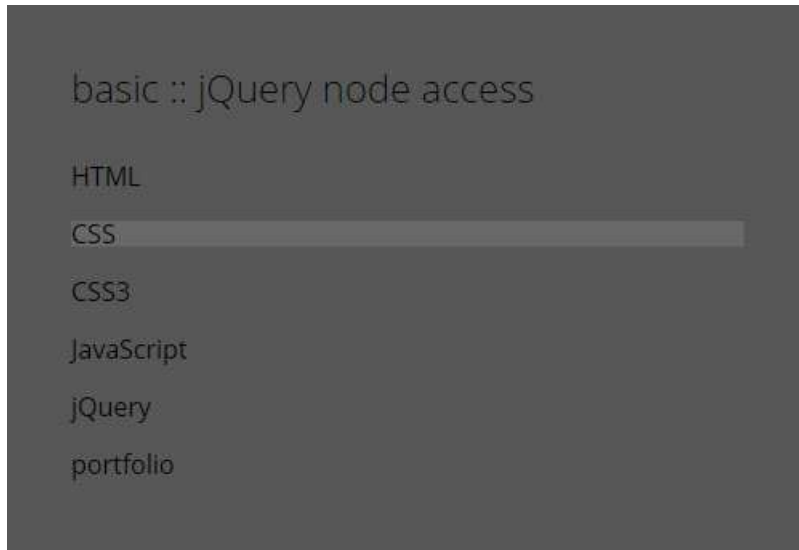
#### JavaScript

```
38 <script>
39 $(function(){
40     $(".container").find(".html").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$(".container").find(".html").css({ "background-color": "#666" });  
container 클래스의 하위 html 클래스 요소가 선택됩니다.



next() 탐색에 대해 알아보시다.



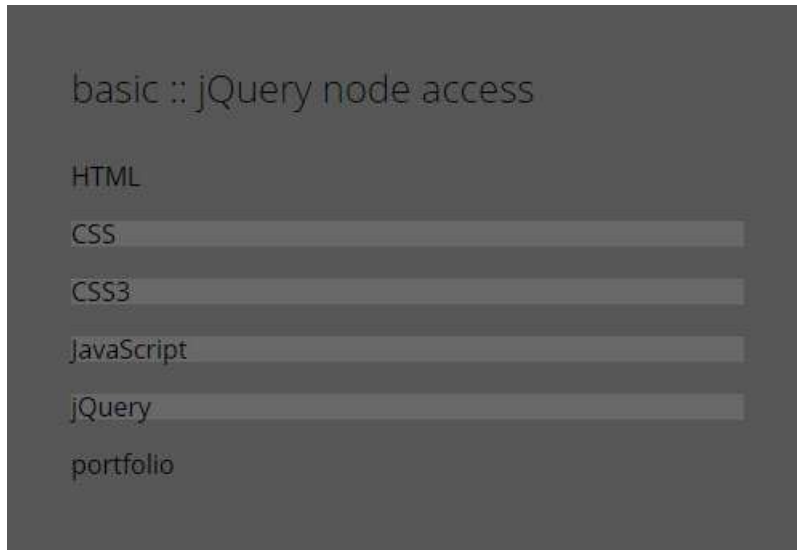
#### JavaScript

```
38 <script>
39 $(function(){
40     $(".html").next().css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$(".html").next().css({ "background-color": "#666" });

html 클래스의 다음 li 요소가 선택됩니다.

nextAll() 탐색에 대해 알아보시다.



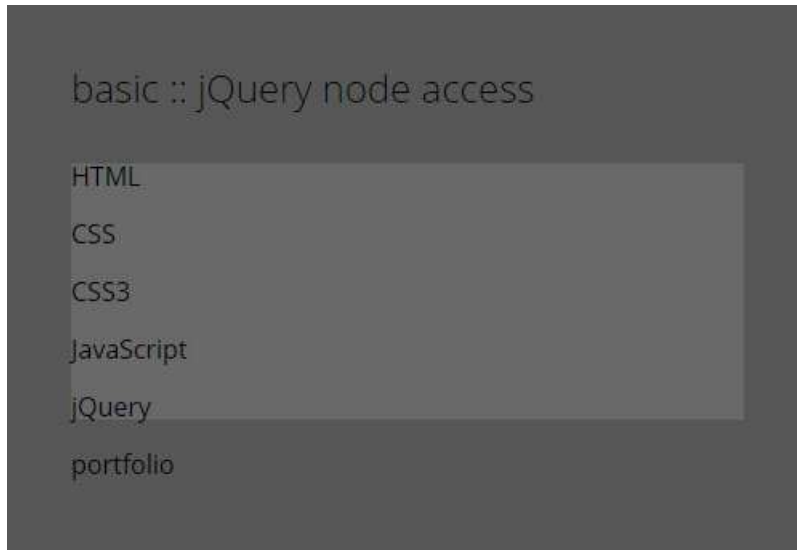
#### JavaScript

```
38 <script>
39 $(function(){
40     $(".html").nextAll().css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$(".html").nextAll().css({ "background-color": "#666" });

html 클래스의 다음 li 요소가 모두 선택됩니다.

parent() 탐색에 대해 알아보시다.



#### JavaScript

```
38 <script>
39 $(function(){
40     $(".html").parent().css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$(".html").parent().css({ "background-color": "#666" });  
html 클래스의 상위 요소가 선택됩니다. ul 요소가 선택됩니다.

parents() 탐색에 대해 알아보시다.

basic :: jQuery node access

HTML

CSS

CSS3

JavaScript

jQuery

portfolio

#### JavaScript

```
38 <script>
39 $(function(){
40     $(".html").parents().css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$(".html").parents().css({ "background-color": "#666" });

html 클래스의 모든 상위 요소가 선택됩니다. html, body, .container, ul 요소가 모두 선택됩니다.

parents(".container")라고 지정한다면, container 클래스만 선택됩니다.

예제의 코드를 바꾸어 보겠습니다.

basic :: jQuery node access

HTML

CSS

CSS3

JavaScript

jQuery

portfolio

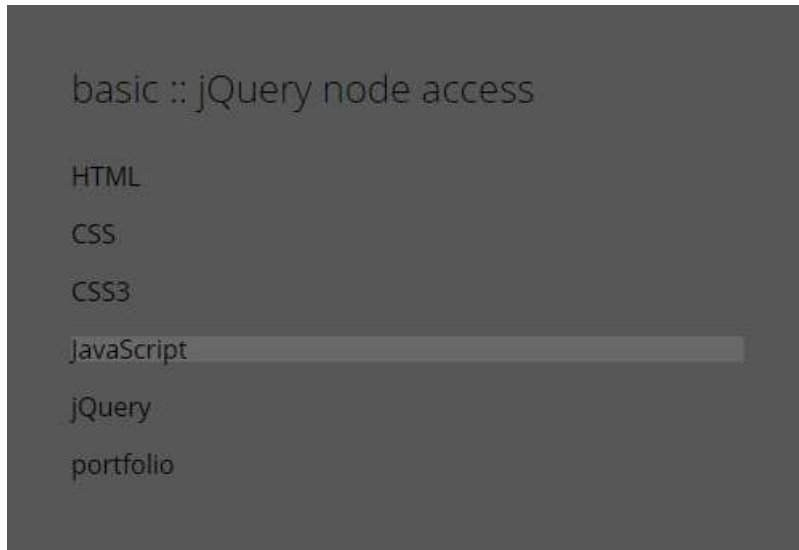
#### JavaScript

```
38 <script>
39 $(function(){
40     $(".html").parents("body").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$(".html").parents("body").css({ "background-color": "#666" });

html 클래스의 모든 상위 요소 중에서 body 요소가 선택됩니다.

prev() 탐색에 대해 알아보시다.



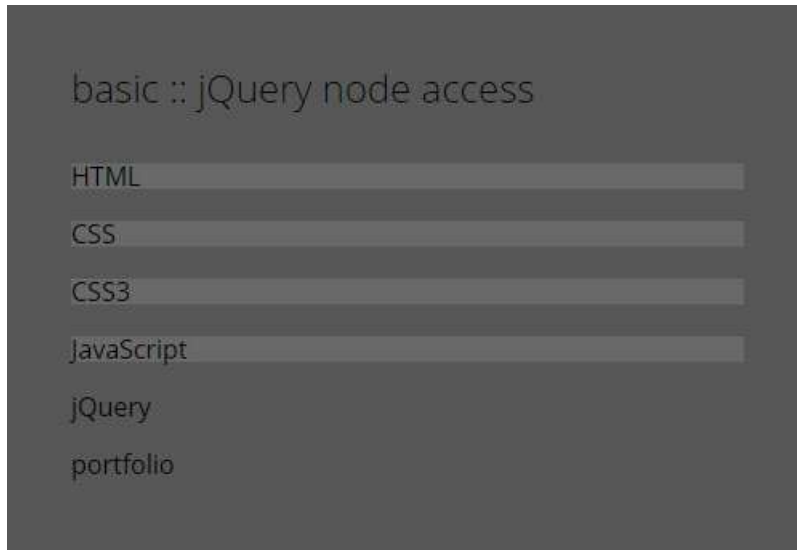
#### JavaScript

```
38 <script>
39 $(function(){
40     $(".jquery").prev().css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$(".jquery").prev().css({ "background-color": "#666" });

jquery 클래스의 이전 li 요소가 선택됩니다. javascript 클래스가 선택됩니다.

prevAll() 탐색에 대해 알아보시다.

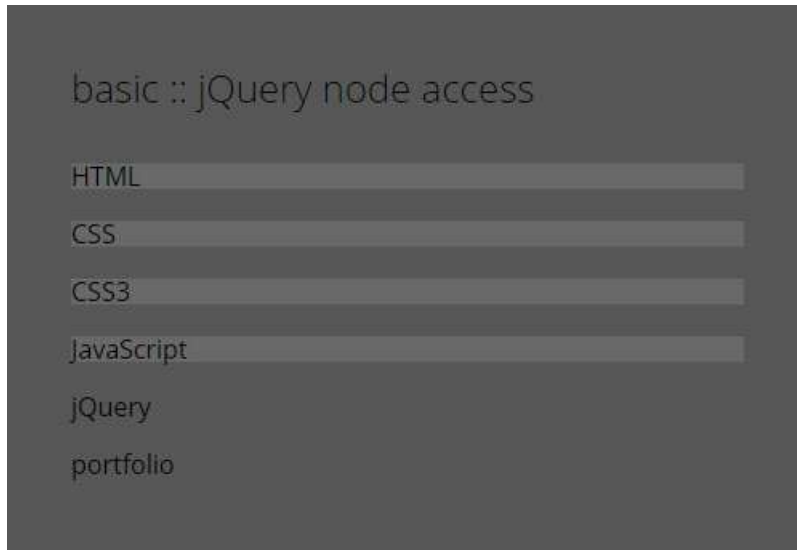


#### JavaScript

```
38 <script>
39 $(function(){
40     $(".jquery").prevAll().css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$(".jquery").prevAll().css({ "background-color": "#666" });  
jquery 클래스의 이전 모든 li 요소가 선택됩니다.

siblings() 탐색에 대해 알아보시다.



#### JavaScript

```
38 <script>
39 $(function(){
40     $(".jquery").siblings().css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$(".jquery").siblings().css({ "background-color": "#666" });

jquery 클래스의 형제 요소가 모두 선택됩니다.



### 1-2-2) 필터링

선택된 집합에서 다시 조건에 맞는 집합을 선택하려고 한다면, 필터링을 활용합니다.

| 명령어      | 설명                                            |
|----------|-----------------------------------------------|
| eq()     | 인덱스(index)에 해당하는 요소를 선택합니다.                   |
| filter() | 선택된 요소 집합에서 선택자를 추가하는 방식으로 사용될 수 있습니다.        |
| first()  | 선택된 노드 집합에서 첫 번째 자식 요소를 선택합니다.                |
| last()   | 선택된 노드 집합에서 마지막 자식 요소를 선택합니다.                 |
| has()    | 선택된 요소들이 자신의 자식 요소에서 주어진 선택자가 있는지를 확인합니다.     |
| is()     | 매개변수에 다양한 값을 입력하여, 입력한 요소가 있으면 true 값을 표시합니다. |
| map()    | 대상이 되는 요소 집합을 변경합니다.                          |
| not()    | 조건에 맞지 않는 요소를 선택합니다.                          |
| slice()  | 선택된 집합을 조건의 범위로 다시 선택합니다.                     |

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery9.html

basic :: jQuery node access

HTML

CSS

CSS3

JavaScript

jQuery

portfolio

## JavaScript

```
38 <script>
39 $(function(){
40     $(".container li").eq(2).css({ "background-color": "#666" });
41 });
42 </script>
```

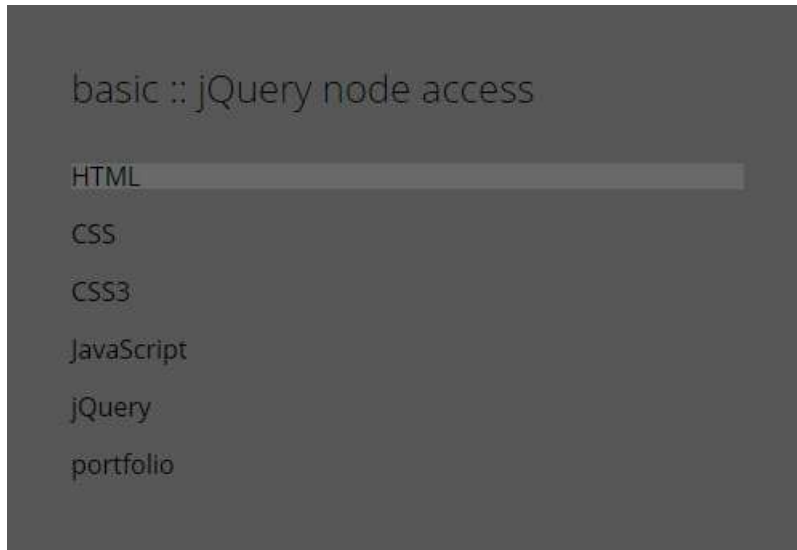
40행 : \$(".container li").eq(2).css({ "background-color": "#666" });

li 요소 중에서 인덱스가 2인 요소가 선택됩니다. 순서로는 3번째 요소가 선택됩니다.

기본 필터 선택자 방식으로 선택한다면 아래와 같습니다.

```
$(".container li:eq(2)").css({ "background-color": "#666" });
```

filter() 필터링에 대해 알아보시다.

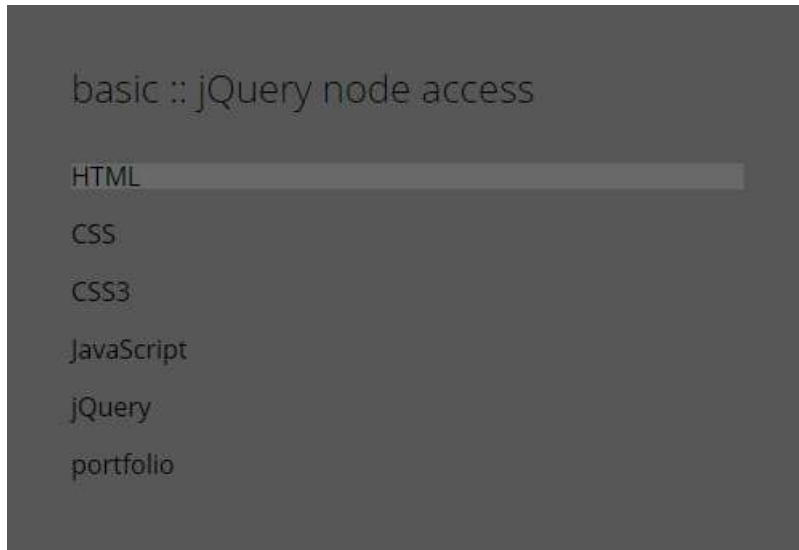


#### JavaScript

```
38 <script>
39 $(function(){
40     $(".container li").filter(".html").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : `$(".container li").filter(".html").css({ "background-color": "#666" });`  
li 요소 중에서 html 클래스가 선택됩니다.

first() 필터링에 대해 알아봅시다.



#### JavaScript

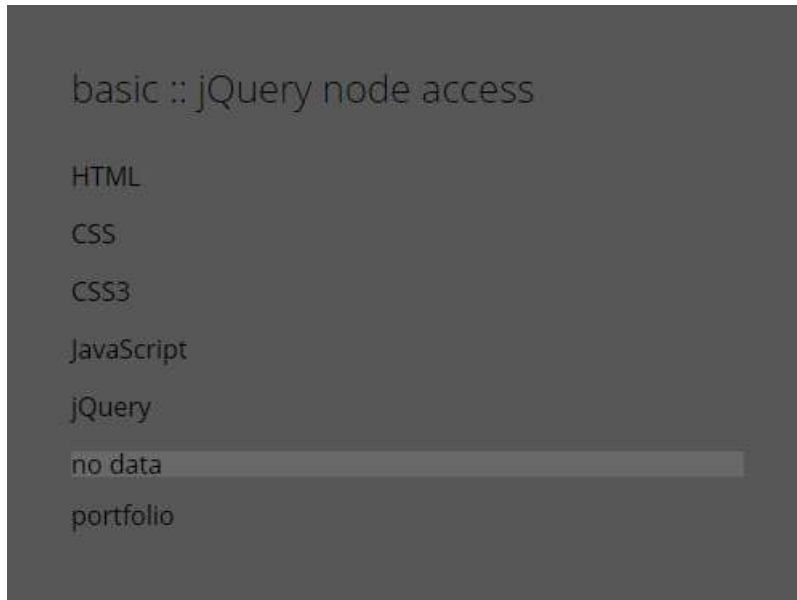
```
38 <script>
39 $(function(){
40     $(".container li").first().css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$(".container li").first().css({ "background-color": "#666" });  
li 요소 중에서 첫 번째 li 요소가 선택됩니다.

기본 필터 선택자 방식으로 선택한다면 아래와 같습니다.

```
$(".container li:first").css({ "background-color": "#666" });
```

last() 필터링에 대해 알아보시다.



#### JavaScript

```
38 <script>
39 $(function(){
40     $(".container li").last().text("no data").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$(".container li").last().text("no data").css({ "background-color": "#666" });

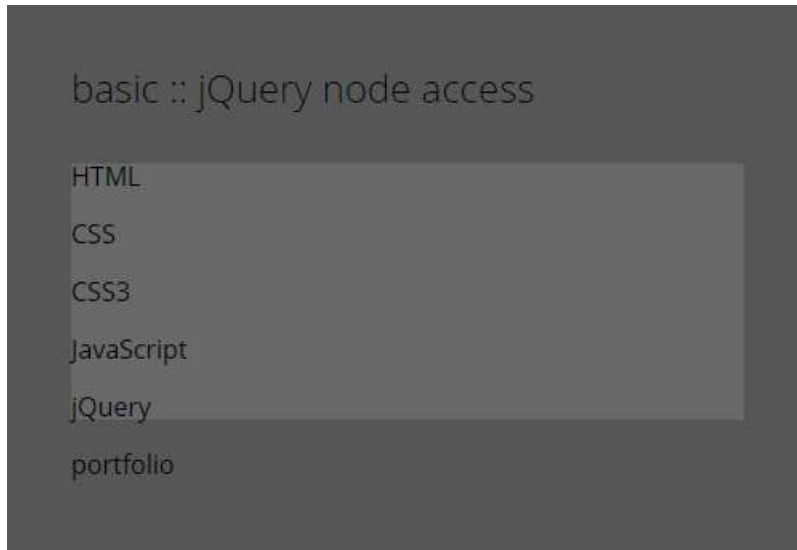
li 요소 중에서 마지막 li 요소가 선택됩니다.

마지막 li 요소는 비어있기 때문에 'no data'를 작성하고, 스타일을 변경합니다.

기본 필터 선택자 방식으로 선택한다면 아래와 같습니다.

```
$(".container li:last").text("no data").css({ "background-color": "#666" });
```

has() 필터링에 대해 알아보시다.



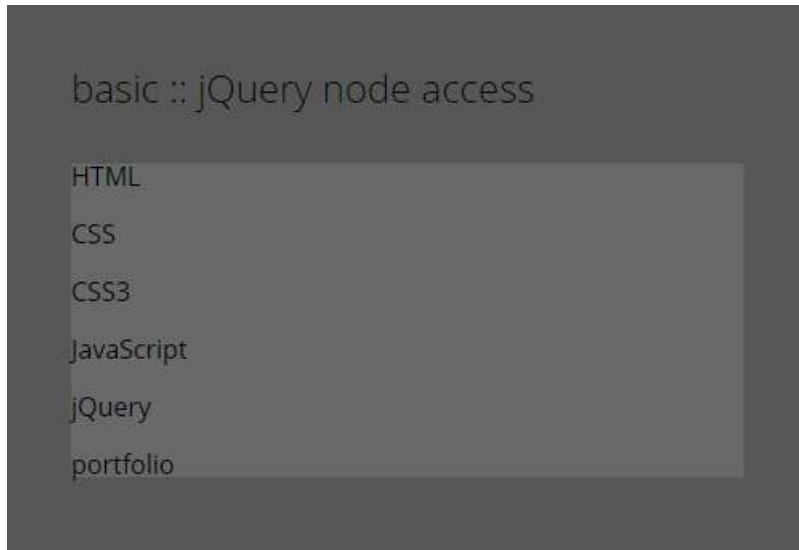
#### JavaScript

```
38 <script>
39 $(function(){
40     $(".container ul").has(".html").css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$(".container ul").has(".html").css({ "background-color": "#666" });

ul 요소 하위에 html 클래스가 있는지를 확인합니다. html 클래스가 있으면 ul 요소가 선택됩니다.

is() 필터링에 대해 알아보시다.



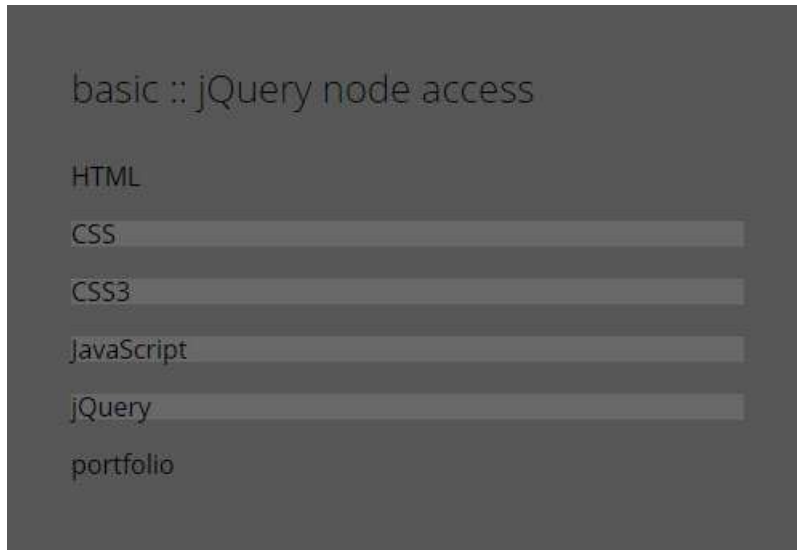
#### JavaScript

```
38 <script>
39 $(function(){
40     if($(".container ul").is("#tutorial")){
41         $(".container").css({ "background-color": "#666" });
42     }
43 });
44 </script>
```

40행 : if(\$(".container ul").is("#tutorial")){  
ul 요소의 아이디가 tutorial이면 하위 중괄호가 실행됩니다.

41행 : \$(".container").css({ "background-color": "#666" });  
container 클래스가 선택됩니다.

not() 필터링에 대해 알아보시다.



#### JavaScript

```
38 <script>
39 $(function(){
40     $(".container li").not(".html").css({ "background-color": "#666" });
41 });
42 </script>
```

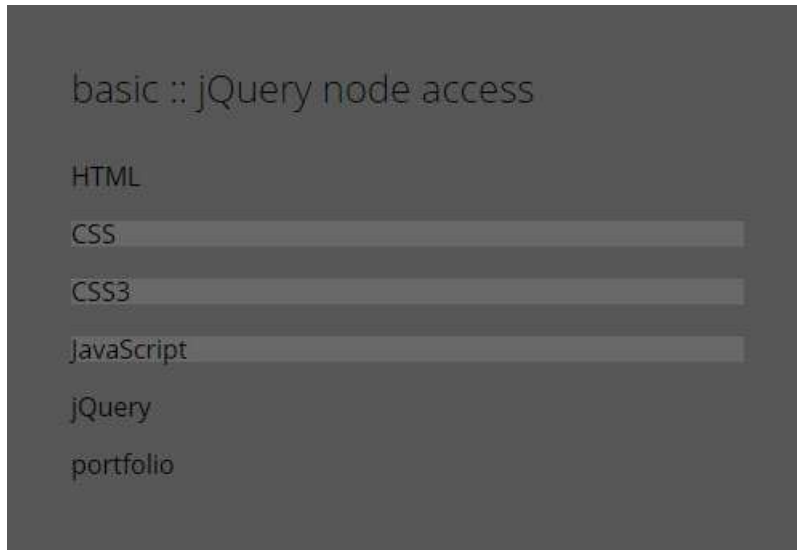
40행 : \$(".container li").not(".html").css({ "background-color": "#666" });  
container 클래스 하위의 li 요소 중에서 html 클래스가 아닌 요소들이 선택됩니다.

기본 필터 선택자 방식으로 선택한다면 아래와 같습니다.

```
$(".container li:not('.html']").css({ "background-color": "#666" });
```



마지막으로 slice() 필터링에 대해 알아보시다.



#### JavaScript

```
38 <script>
39 $(function(){
40     $(".container li").slice(1, 4).css({ "background-color": "#666" });
41 });
42 </script>
```

40행 : \$(".container li").slice(1, 4).css({ "background-color": "#666" });

container 클래스 하위의 li 요소 중에서 두 번째 요소에서 다섯 번째 이전 요소까지 선택됩니다.

slice() 명령어의 용법은 아래와 같습니다.

```
.slice(start [, end]);
```

start : 0에서 시작되는 시작 번호를 의미합니다. 음수로 처리될 경우에는 뒤에서부터 선택됩니다.

end : 0에서 시작되는 끝 번호를 의미합니다. 끝 번호 이전까지 선택되며, 생략될 경우 하위 모든 요소가 선택됩니다.

### 1-3) 노드 구조 변경(DOM)

노드 조작(Manipulation)은 DOM 구조를 변경할 수 있는 jQuery 명령어들의 집합입니다.

| 명령어                  | 설명                                     |
|----------------------|----------------------------------------|
| append()             | 선택 요소 내부의 맨 뒤에 자식 요소를 새로 추가합니다.        |
| appendTo(target)     | 선택 요소를 target에 해당하는 요소 내부의 뒤에 추가합니다.   |
| prepend()            | 선택 요소 내부의 맨 앞에 자식 요소를 새로 추가합니다.        |
| prependTo(target)    | 선택 요소를 target에 해당하는 요소 내부의 맨 앞에 추가합니다. |
| clone()              | 선택한 요소와 똑같은 요소를 복사합니다.                 |
| wrap()               | 선택된 집합을 각각의 매개변수로 넘긴 HTML 구조로 감쌉니다.    |
| wrapAll()            | 선택된 집합의 전체 외곽을 매개변수로 넘긴 HTML 구조로 감쌉니다. |
| wrapInner()          | 선택된 집합의 내부에 매개변수로 넘긴 HTML 구조로 감쌉니다.    |
| text()               | 요소의 텍스트를 읽고 쓸 수 있습니다.                  |
| html()               | 요소 내부의 HTML을 읽고 쓸 수 있습니다.              |
| after()              | 선택된 요소 뒤에 새로운 요소를 추가합니다.               |
| insertAfter(target)  | 선택 요소를 target에 해당하는 요소 내부의 맨 뒤에 추가합니다. |
| before()             | 선택된 요소 앞에 새로운 요소를 추가합니다.               |
| insertBefore(target) | 선택된 jQuery 객체를 target 앞에 추가합니다.        |
| detach()             | DOM에서 일치되는 요소를 제거합니다.                  |
| empty()              | DOM에서 일치되는 요소들의 자식 요소를 제거합니다.          |
| remove()             | DOM에서 일치되는 요소를 제거합니다.                  |
| replaceAll()         | 조건에 맞는 요소들을 매개변수로 넘긴 요소들로 대체합니다.       |
| replaceWith()        | 조건에 맞는 요소들을 매개변수로 넘긴 새로운 HTML로 대체합니다.  |

### 1-3-1) append()

선택된 요소 내부의 가장 뒤에 매개 변수로 전달되는 문자열 및 요소를 추가해주는 명령어입니다.

#### 시작 파일

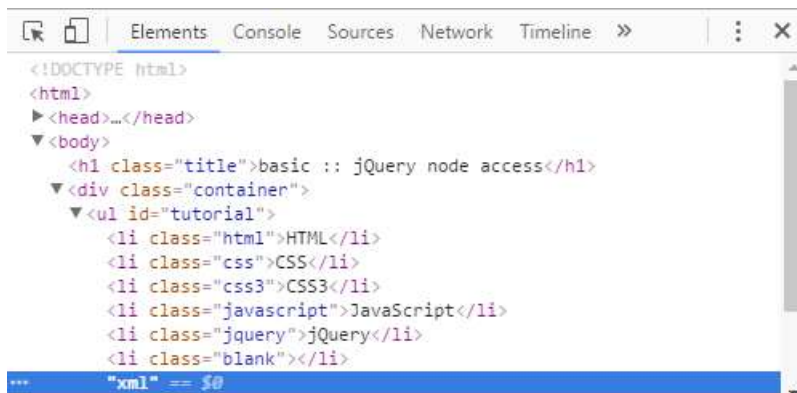
sample/basic\_jquery/start/basic\_jquery10.html

#### JavaScript

```
38     <script>
39     $(function(){
40         $("#tutorial").append("xml");
41     });
42     </script>
```

40행 : \$("#tutorial").append("xml");

li 요소 마지막 영역에 'xml' 텍스트가 추가됩니다.



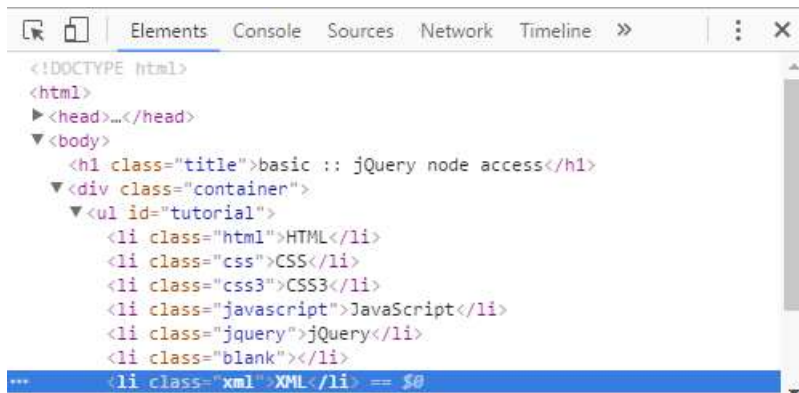
예제의 코드를 바꾸어 보겠습니다. HTML 문법에 맞게 li 노드 구조를 추가하는 코드입니다. append() 명령어는 단순한 텍스트를 추가할 수 있을 뿐 아니라, HTML 노드 또한 추가할 수 있습니다.

#### JavaScript

```
38 <script>
39 $(function(){
40     $("#tutorial").append('<li class="xml">XML</li>');
41 });
42 </script>
```

40행 : \$("#tutorial").append('<li class="xml">XML</li>');

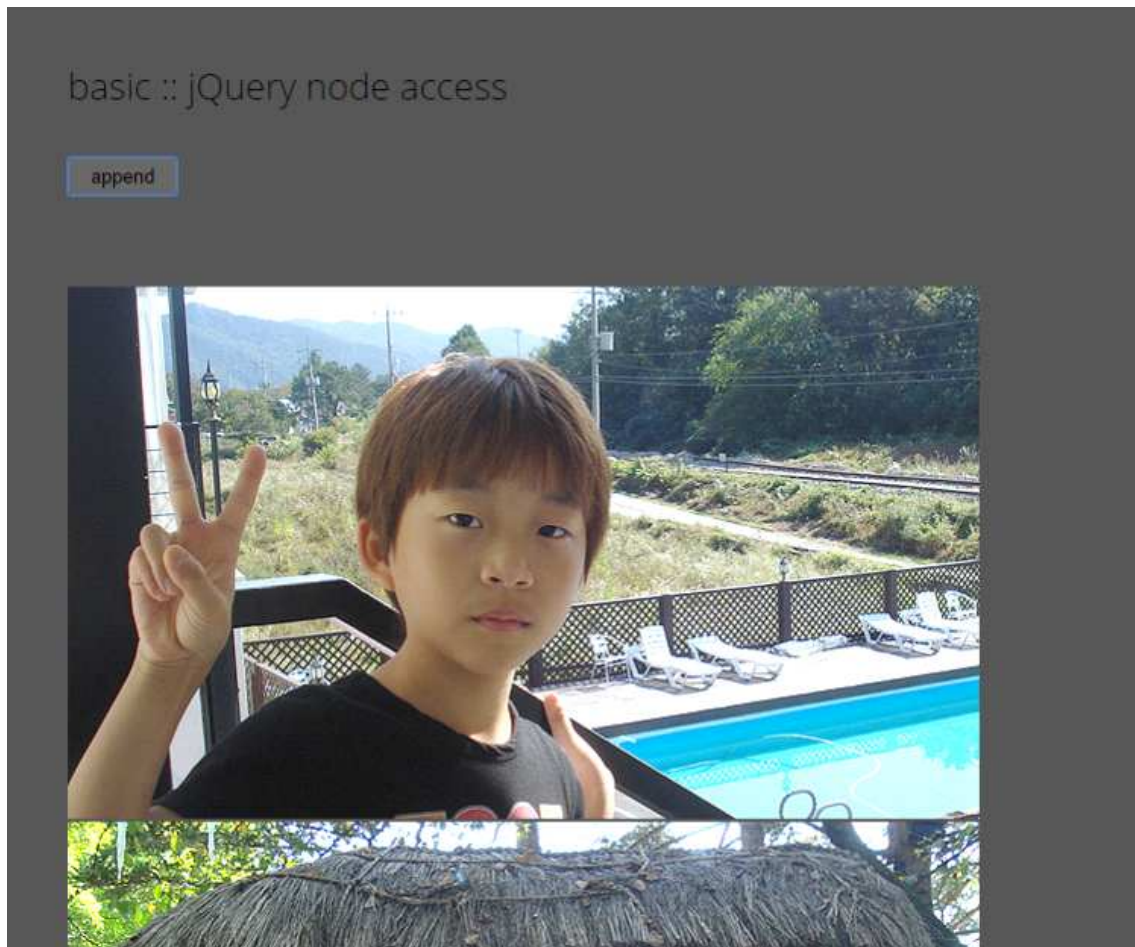
li 요소 마지막 영역에 '<li class="xml">XML</li>' 형식의 HTML 노드가 추가됩니다.



이미지를 추가하는 예제를 하나 더 확인해 봅시다. [append] 버튼을 누르면 이미지가 추가되는 예제입니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery11.html



#### JavaScript

```
54 <script>
55 $(function(){
56     var n=0;
57
58     $("#append").click(function(){
59         $(".gallery").append(fetchImg());
60     });
61     function fetchImg(){
62         if(n < 4){
```

```

63         n++;
64     }
65     var imgPath="<img src='images/slide'+n+'.jpg' alt=''>";
66     // console.log("imgPath : "+imgPath);
67     return imgPath;
68 }
69 });
70 </script>

```

56행 : var n=0;

이미지 추가 경로에 관련된 변수를 선언합니다. 이미지의 경로는 slide1.jpg ~ slide4.jpg입니다.

59행 : \$(".gallery").append(fetchImg());

append id 버튼을 클릭할 경우 실행되는 핸들러입니다. fetchImg() 함수에서 return되는 내용으로 추가합니다.

63행 : n++;

n 변수 값이 4보다 작을 경우 하나씩 증가합니다.

65행 : var imgPath="<img src='images/slide'+n+'.jpg' alt=''>";

n 변수 값을 이용해서 imgPath 변수에 대입합니다. 이 값은 gallery 클래스에 추가되는 노드가 될 것입니다.

### 1-3-2) appendTo()

선택 요소를 target에 해당하는 요소 내부의 맨 뒤에 추가합니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery12.html

basic :: jQuery node access

HTML

CSS

CSS3

JavaScript

jQuery

XML

portfolio

#### JavaScript

```
38 <script>
39 $(function(){
40     $('<li class="xml">XML</li>').appendTo("#tutorial");
41 });
42 </script>
```

40행 : \$('<li class="xml">XML</li>').appendTo("#tutorial");

tutorial id 요소에 매개인자로 넘긴 HTML 노드를 추가합니다.

append() 명령어로 추가되는 코드와 비교해 보세요.

```
$("#tutorial").append('<li class="xml">XML</li>');
```



The screenshot shows the 'Elements' panel of a web browser's developer tools. The document structure is as follows:

- <!DOCTYPE html>
- <html>
- <head>...</head>
- <body>
- <h1 class="title">basic :: jQuery node access</h1>
- <div class="container">
- <ul id="tutorial">
- <li class="html">HTML</li>
- <li class="css">CSS</li>
- <li class="css3">CSS3</li>
- <li class="javascript">JavaScript</li>
- <li class="jquery">jQuery</li>
- <li class="blank"></li>
- <li class="xml">XML</li> == \$0



### 1-3-3) prepend()

선택 요소 내부의 맨 앞에 자식 요소를 새로 추가합니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery13.html

basic :: jQuery node access

XML

HTML

CSS

CSS3

JavaScript

jQuery

portfolio

#### JavaScript

```
38 <script>
39 $(function(){
40     $("#tutorial").prepend('<li class="xml">XML</li>');
41 });
42 </script>
```

40행 : \$("#tutorial").prepend('<li class="xml">XML</li>');

tutorial id 요소에 해당하는 노드를 추가해 줍니다. append()는 해당하는 노드를 마지막에 추가하지만, prepend()는 해당하는 노드를 처음 부분에 추가해 줍니다.



```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <h1 class="title">basic :: jQuery node access</h1>
    <div class="container">
      <ul id="tutorial">
        <li class="xml">XML</li> == $0
        <li class="html">HTML</li>
        <li class="css">CSS</li>
        <li class="css3">CSS3</li>
        <li class="javascript">JavaScript</li>
        <li class="jquery">jQuery</li>
        <li class="blank"></li>
```

#### 1-3-4) prependTo()

선택 요소를 target에 해당하는 요소 내부의 맨 앞에 추가합니다.

##### 시작 파일

sample/basic\_jquery/start/basic\_jquery14.html

basic :: jQuery node access

XML

HTML

CSS

CSS3

JavaScript

jQuery

portfolio

##### JavaScript

```
38 <script>
39 $(function(){
40     $('<li class="xml">XML</li>').prependTo("#tutorial");
41 });
42 </script>
```

40행 : \$('<li class="xml">XML</li>').prependTo("#tutorial");  
tutorial id 요소에 해당하는 노드를 맨 앞에 추가해 줍니다.

prepend() 명령어로 추가되는 코드와 비교해 보세요.

```
$("#tutorial").prepend('<li class="xml">XML</li>');
```

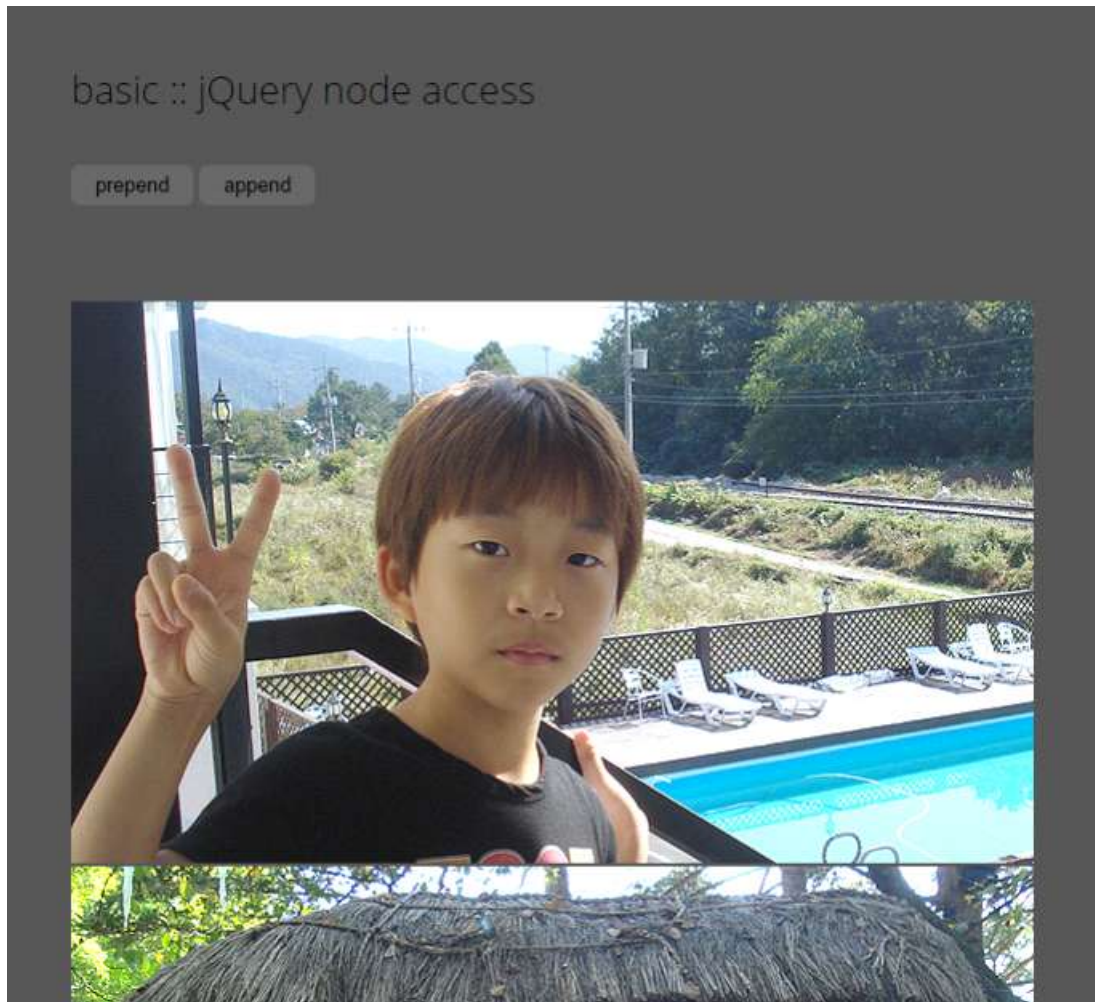


```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <h1 class="title">basic :: jQuery node access</h1>
    <div class="container">
      <ul id="tutorial">
        *** <li class="xml">XML</li> == $0
        <li class="html">HTML</li>
        <li class="css">CSS</li>
        <li class="css3">CSS3</li>
        <li class="javascript">JavaScript</li>
        <li class="jquery">jQuery</li>
        <li class="blank"></li>
```

이번에는 append(), prepend() 명령어를 사용해서 이미지를 추가하는 예제를 구현해 봅시다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery15.html



#### JavaScript

```
54 $(function(){
55     $("#prepend").click(function(){
56         $(".gallery").prepend($(".gallery img:last"));
57     });
58     $("#append").click(function(){
59         $(".gallery").append($(".gallery img:first"));
60     });
61 });
```

56행 : \$(".gallery").prepend(\$(".gallery img:last"));

prepend id 버튼을 누르면 이미지 중에서 마지막 위치가 처음 위치로 재배치됩니다. 기존에 있던 노드가 prepend() 명령어로 추가된다면 원래 있던 위치의 노드는 삭제되고 재배치됩니다.

59행 : \$(".gallery").append(\$(".gallery img:first"));

append id 버튼을 누르면 이미지 중에서 처음 위치가 마지막 위치로 재배치됩니다. 기존에 있던 노드가 append() 명령어로 추가된다면 원래 있던 위치의 노드는 삭제되고 재배치됩니다.

1-3-5) clone()

선택한 요소와 똑같이 요소를 복사합니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery16.html

basic :: jQuery node access

HTML

CSS

CSS3

JavaScript

jQuery

portfolio

portfolio

#### JavaScript

```
38 <script>
39 $(function(){
40     $("#portfolio").clone().appendTo("body");
41 });
42 </script>
```

40행 : \$("#portfolio").clone().appendTo("body");

portfolio id 요소를 복제하여, body 영역 중 제일 마지막 부분에 추가합니다.

```
Elements Console Sources Network Timeline » ⋮ ✕
<!DOCTYPE html>
<html>
  ▶ <head>...</head>
  ▼ <body>
    <h1 class="title">basic :: jQuery node access</h1>
    ▶ <div class="container">...</div>
*** <p id="portfolio">portfolio</p> == $0
    </body>
  </html>
```



### 1-3-6) wrap()

선택된 집합을 각각의 매개변수로 넘긴 HTML 구조로 감쌉니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery17.html

#### JavaScript

```
38 <script>
39 $(function(){
40     $("#tutorial").wrap('<div class="nav"></div>');
41 });
42 </script>
```

40행 : \$("#tutorial").wrap('<div class="nav"></div>');

tutorial id 요소를 nav 클래스로 감쌉니다.



### 1-3-7) wrapAll()

선택된 집합의 전체 외곽을 매개변수로 넘긴 HTML 구조로 감쌉니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery18.html

#### JavaScript

```
38 <script>
39 $(function(){
40     $("li").wrapAll('<div class="new"></div>');
41 });
42 </script>
```

40행 : \$("li").wrapAll('<div class="new"></div>');

li 요소를 new 클래스로 감쌉니다.



### 1-3-8) wrapInner()

선택된 집합의 내부에 매개변수로 넘긴 HTML 구조로 감쌉니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery19.html

#### JavaScript

```
38     <script>
39     $(function(){
40         $("#portfolio").wrapInner('<span class="new"> </span>');
41     });
42     </script>
```

40행 : \$("#portfolio").wrapInner('<span class="new"> </span>');  
portfolio id 요소 내부에 해당하는 노드를 추가해 줍니다.



1-3-9) text()

어떤 집합 요소의 텍스트를 읽거나 쓰거나 할 때 사용합니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery20.html

basic :: jQuery node access

HTML

CSS

CSS3

JavaScript

jQuery

portfolio

portfolio

#### JavaScript

```
38 <script>
39 $(function(){
40     var string=$("#portfolio").text();
41     $(".blank").text(string);
42 });
43 </script>
```

40행 : `var string=$("#portfolio").text();`

portfolio id 요소에 작성된 텍스트를 string 변수에 대입합니다. 'portfolio' 입니다.

41행 : `$(".blank").text(string);`

blank 클래스에 'portfolio'를 작성합니다.

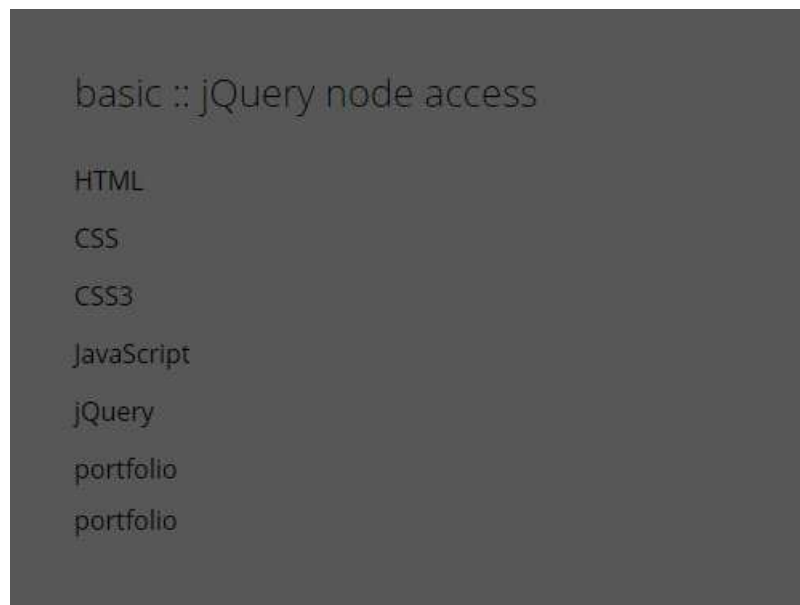
```
Elements Console Sources Network Timeline >> ⋮ ✕
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <h1 class="title">basic :: jQuery node access</h1>
    <div class="container">
      <ul id="tutorial">
        <li class="html">HTML</li>
        <li class="css">CSS</li>
        <li class="css3">CSS3</li>
        <li class="javascript">JavaScript</li>
        <li class="jquery">jQuery</li>
        ... <li class="blank">portfolio</li> == $0
      </ul>
      <p id="portfolio">portfolio</p>
    </div>
  </body>
</html>
```

### 1-3-10) html()

html() 명령어는 자바스크립트의 innerHTML과 같은 기능으로, 선택 집합의 내부 요소들을 HTML 문자로 나타냅니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery21.html



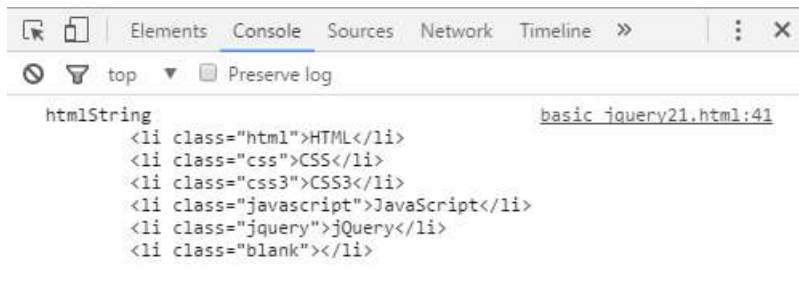
#### JavaScript

```
38 <script>
39 $(function(){
40     var htmlString=$("#tutorial").html();
41     console.log("htmlString", htmlString);
42
43     $(".blank").html('<p class="portfolio">portfolio</p>');
44 });
45 </script>
```

40행 : `var htmlString=$("#tutorial").html();`

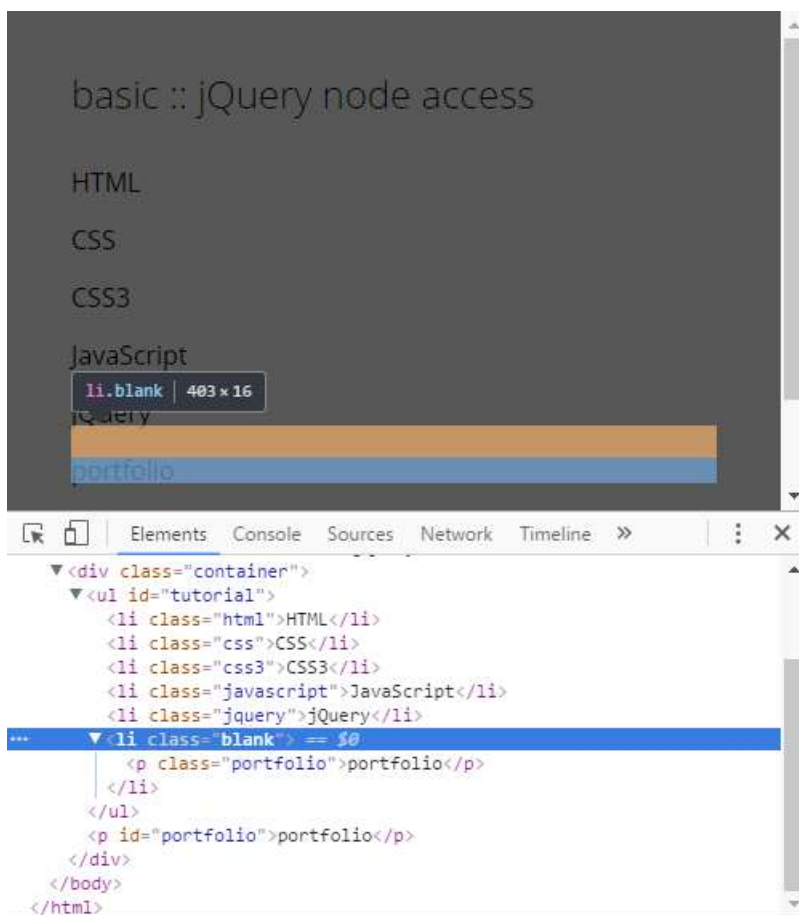
tutorial id 요소의 내부 html 내용을 htmlString 변수에 대입합니다.

아래의 그림은 htmlString 변수를 로그로 확인해 본 결과입니다.



```
htmlString basic jquery21.html:41
<li class="html">HTML</li>
<li class="css">CSS</li>
<li class="css3">CSS3</li>
<li class="javascript">JavaScript</li>
<li class="jquery">jQuery</li>
<li class="blank"></li>
```

43행 : \$(".blank").html('<p class="portfolio">portfolio</p>');  
blank 클래스에 해당 HTML 노드를 작성합니다.



basic :: jQuery node access

HTML

CSS

CSS3

JavaScript

jQuery

portfolio

```
<div class="container">
  <ul id="tutorial">
    <li class="html">HTML</li>
    <li class="css">CSS</li>
    <li class="css3">CSS3</li>
    <li class="javascript">JavaScript</li>
    <li class="jquery">jQuery</li>
    <li class="blank"> == $0
      <p class="portfolio">portfolio</p>
    </li>
  </ul>
  <p id="portfolio">portfolio</p>
</div>
</body>
</html>
```

1-3-11) after()

선택된 요소 뒤에 새로운 요소를 추가합니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery22.html

basic :: jQuery node access

HTML

CSS

CSS3

JavaScript

jQuery

XML

portfolio

#### JavaScript

```
38     <script>
39     $(function(){
40         $(".jquery").after('<li class="xml">XML</li>');
41     });
42     </script>
```

40행 : \$(".jquery").after('<li class="xml">XML</li>');

jquery 클래스 다음 영역에 해당 HTML 노드를 추가합니다.



### 1-3-12) insertAfter()

작성된 jQuery 객체를 target 뒤에 추가합니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery23.html

basic :: jQuery node access

HTML

CSS

CSS3

JavaScript

jQuery

XML

portfolio

#### JavaScript

```
38     <script>
39     $(function(){
40         $('<li class="xml">XML</li>').insertAfter(".jquery")
41     });
42     </script>
```

40행 : \$('<li class="xml">XML</li>').insertAfter(".jquery")  
jquery 클래스 다음 영역에 해당 HTML 노드를 추가합니다.

after() 명령어로 추가되는 코드와 비교해 보세요.

```
$(".jquery").after('<li class="xml">XML</li>');
```

1-3-13) before()

선택된 요소 앞에 새로운 요소를 추가합니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery24.html

basic :: jQuery node access

Basic Study

HTML

CSS

CSS3

JavaScript

jQuery

portfolio

#### JavaScript

```
38     <script>
39     $(function(){
40         $(".html").before('<li class="basic">Basic Study</li>');
41     });
42     </script>
```

40행 : \$(".html").before('<li class="basic">Basic Study</li>');

html 클래스 이전 영역에 해당 HTML 노드를 추가합니다.

1-3-14) insertBefore()

작성된 jQuery 객체를 target 앞에 추가합니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery25.html

basic :: jQuery node access

Basic Study

HTML

CSS

CSS3

JavaScript

jQuery

portfolio

#### JavaScript

```
38    <script>
39    $(function(){
40        $('<li class="basic">Basic Study</li>').insertBefore(".html");
41    });
42    </script>
```

40행 : \$('<li class="basic">Basic Study</li>').insertBefore(".html")

html 클래스 이전 영역에 해당 HTML 노드를 추가합니다.

before() 명령어로 추가되는 코드와 비교해 보세요.

```
$(".html").before('<li class="basic">Basic Study</li>');
```

### 1-3-15) empty()

선택한 요소의 내용을 제거합니다. 단 선택 노드는 삭제되지 않는다는 것이 remove() 명령어와의 차이점입니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery26.html

basic :: jQuery node access

CSS

CSS3

JavaScript

jQuery

portfolio

#### JavaScript

```
38 <script>
39 $(function(){
40     $(".container li:first").empty();
41 });
42 </script>
```

40행 : \$(".container li:first").empty();

container 클래스에 첫 번째 li 노드의 하위 텍스트 ('HTML')를 제거합니다.

1-3-16) remove()

선택한 요소를 제거합니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery27.html

basic :: jQuery node access

CSS

CSS3

JavaScript

jQuery

portfolio

#### JavaScript

```
38 <script>
39 $(function(){
40     $(".container li:first").remove();
41 });
42 </script>
```

40행 : \$(".container li:first").remove();

container 클래스에 첫 번째 li 노드를 제거합니다.

### 1-3-17) detach()

선택한 요소를 제거합니다. 단 remove() 명령어와 다른 점은 제거된 요소들은 임시로 보관할 수 있다는 것입니다. 이 명령어는 DOM에서 제거했다가 다시 추가할 때 유용하게 사용됩니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery28.html

basic :: jQuery node access

HTML

CSS

CSS3

JavaScript

portfolio

#### JavaScript

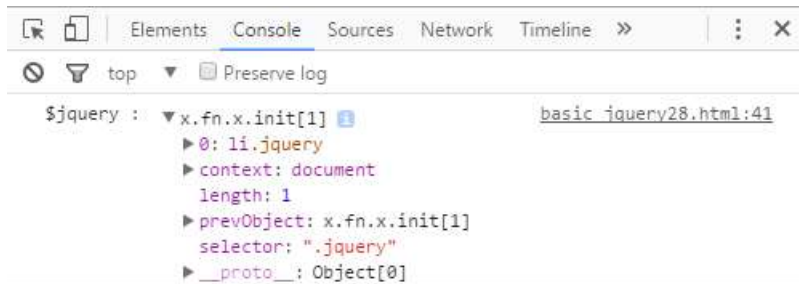
```
38 <script>
39 $(function(){
40     var $jquery=$(".jquery").detach();
41     console.log("$jquery : ", $jquery);
42
43     $("#portfolio").click(function(){
44         if($jquery != ""){
45             $("#tutorial").append($jquery);
46             $jquery="";
47         }else{
48             $jquery=$(".jquery").detach();
49         }
50     });
51 });
52 </script>
```

40행 : var \$jquery=\$(".jquery").detach();

jquery 클래스를 제거합니다. \$jquery 변수에 제거된 내용을 임시로 저장합니다.

41행 : `console.log("$.jquery", $.jquery);`

`$.jquery` 변수의 내용을 `console.log()` 명령어로 확인해 봅시다. 뭔가 저장된 데이터를 확인할 수 있습니다.



43행 : `$("#portfolio").click(function(){`

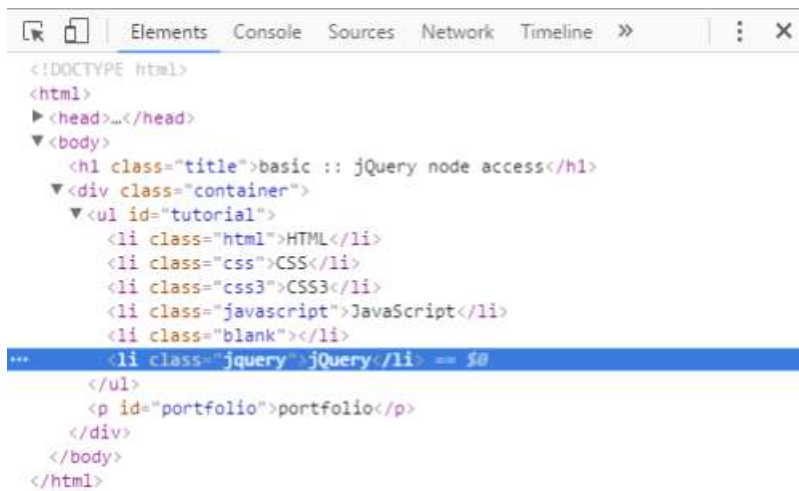
`portfolio` 아이디를 클릭할 때 발생하는 이벤트입니다. 이벤트에 대한 내용은 다음 내용에서 설명하겠습니다.

44행 : `if($.jquery != ""){`

`$.jquery` 변수가 빈 문자가 아닐 경우에 해당하는 조건으로, `jquery` 클래스가 추가되는 상황입니다.

45행 : `$("#tutorial").append($.jquery);`

`tutorial` 아이디에 미리 저장한 데이터를 추가합니다.



46행 : `$.jquery="";`

`$.jquery` 변수를 빈 문자열로 바꾸어 줍니다.

47행 : }else{

\$jquery 변수가 빈 문자열일 경우로 jquery 클래스가 추가된 상황입니다.

48행 : \$jquery=\$("jquery").detach();

다시 jquery 클래스를 제거합니다. \$jquery 변수에 제거된 내용을 임시로 저장합니다.



### 1-3-18) replaceAll()

조건에 맞는 요소들을 target 요소들로 대체합니다.

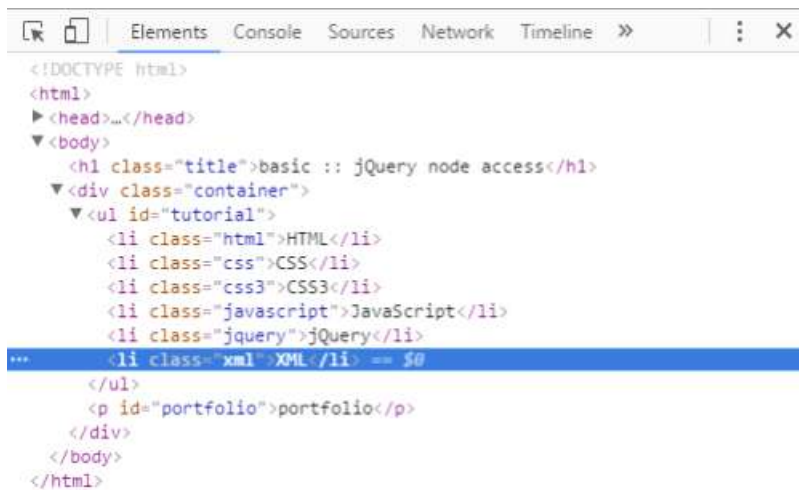
#### 시작 파일

sample/basic\_jquery/start/basic\_jquery29.html

#### JavaScript

```
38    <script>
39    $(function(){
40        $('<li class="xml">XML</li>').replaceAll(".blank");
41    });
42    </script>
```

40행 : \$('<li class="xml">XML</li>').replaceAll(".blank");  
blank 클래스 영역을 해당 노드 (.xml)로 바꾸어 줍니다.



### 1-3-19) replaceWith()

조건에 맞는 요소들을 매개변수로 넘긴 새로운 HTML로 대체합니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery30.html

basic :: jQuery node access

HTML

CSS

CSS3

JavaScript

jQuery

XML

portfolio

#### JavaScript

```
38     <script>
39     $(function(){
40         $(".blank").replaceWith('<li class="xml">XML</li>');
41     });
42     </script>
```

40행 : \$(".blank").replaceWith('<li class="xml">XML</li>');

blank 클래스 영역을 해당 노드 (.xml)로 바꾸어 줍니다.

#### 1-4) 속성(Attributes)

jQuery는 HTML 노드의 속성을 제어할 수 있습니다.

명령어	설명
attr()	HTML 노드 속성을 가져오거나, 바꿔줄 수 있습니다.
prop()	HTML 노드의 속성을 가져오거나, 바꿔줄 수 있습니다. attr()과 거의 유사하지만, prop()는 주로 JavaScript의 속성에 관련된 명령어입니다.
removeAttr()	HTML 노드 속성의 값을 제거합니다.
removeProp()	removeAttr()과 거의 유사하지만, removeProp() 명령어는 속성 값만 제거합니다. 예제로 비교해 봅시다.

##### 1-4-1) attr()

HTML 노드 속성을 가져오거나, 바꿔줄 수 있습니다.

###### 시작 파일

sample/basic\_jquery/start/basic\_jquery31.html

basic :: jQuery node access

HTML

CSS

CSS3

JavaScript

jQuery

portfolio

###### JavaScript

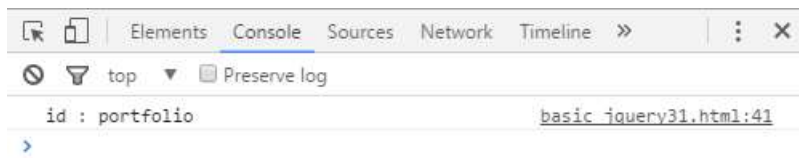
38

<script>

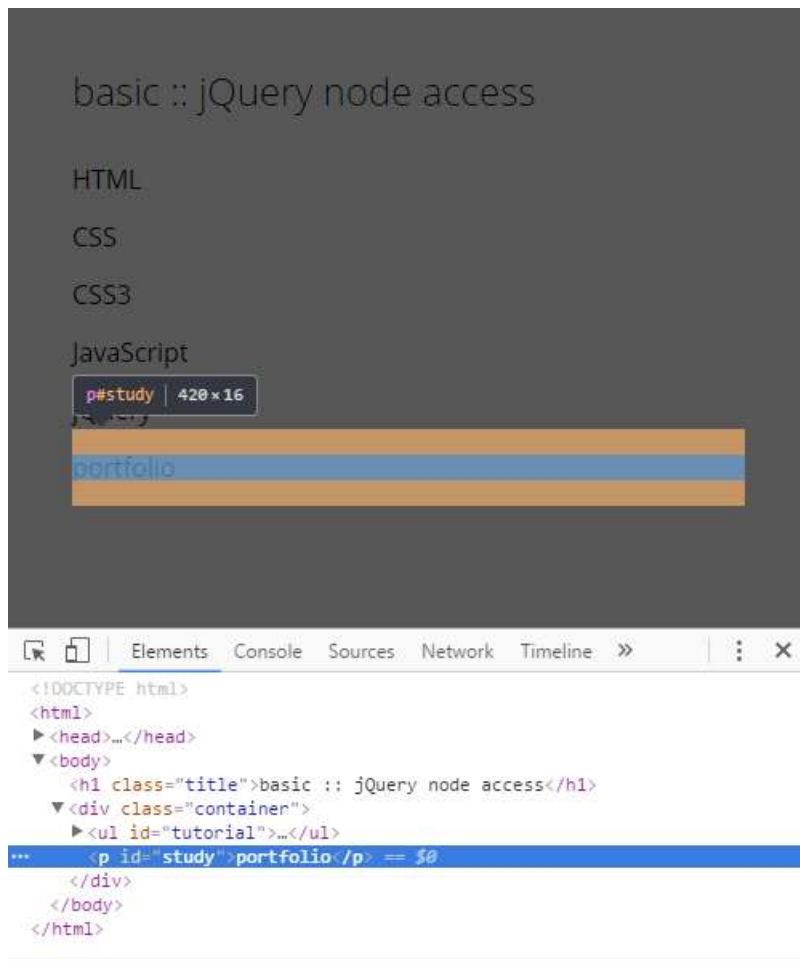
```
39 $(function(){
40     var attr=$("#portfolio").attr("id");
41     console.log("id :", attr);
42
43     $("#portfolio").attr("id", "study");
44 });
45 </script>
```

40행 : `var attr=$("#portfolio").attr("id");`  
portfolio id 요소의 id 속성 값을 attr 변수에 대입합니다.

41행 : `console.log("id :", attr);`  
attr 변수의 값을 console.log()로 확인해 본 결과입니다.



43행 : \$("#portfolio").attr("id", "study");  
portfolio 아이디의 id 속성 값을 변경합니다.



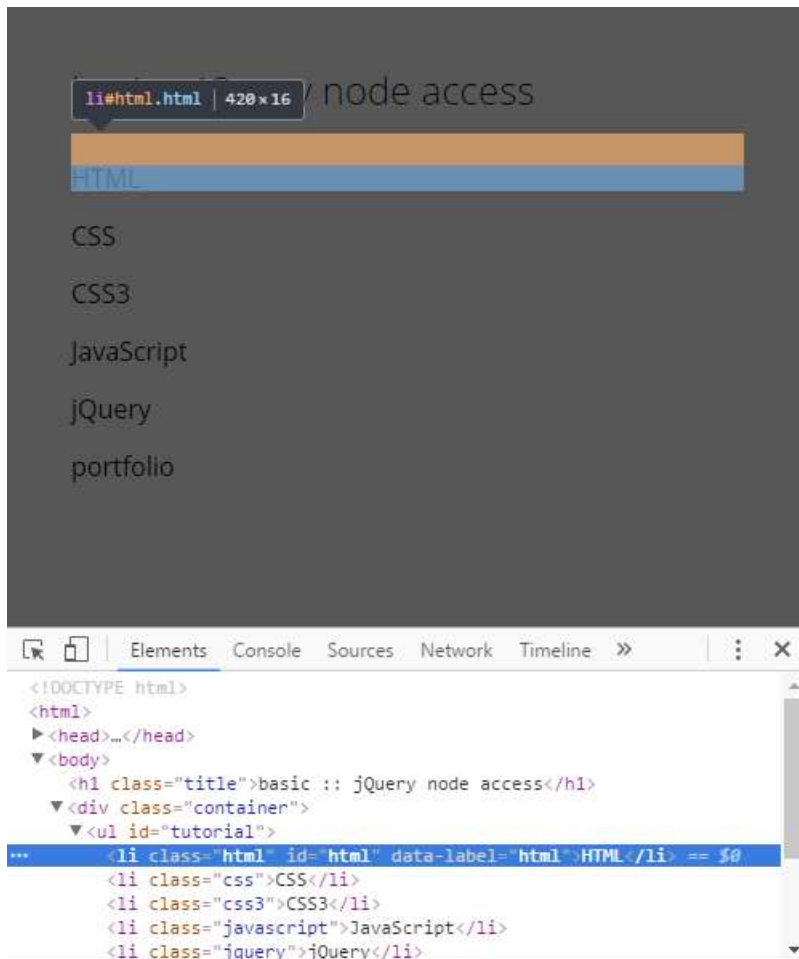
attr() 명령어로 두 개의 속성을 추가해 봅시다.

#### JavaScript

```
38 <script>
39 $(function(){
40     $(".html").attr({ id: "html", "data-label": "html" });
41 });
42 </script>
```

40행 : \$(".html").attr({ id: "html", "data-label": "html" });

container 클래스의 두 개 이상의 속성을 추가합니다. attr() 명령어는 여러 개의 속성을 한 번에 바꾸어 줄 수 있습니다.



## 1-4-2) prop()

HTML 노드의 속성을 가져오거나, 바꿔줄 수 있습니다. attr()과 거의 유사하지만 prop()는 주로 JavaScript의 속성에 관련된 명령어입니다.

예제로 비교해 봅시다.

### 시작 파일

sample/basic\_jquery/start/basic\_jquery32.html

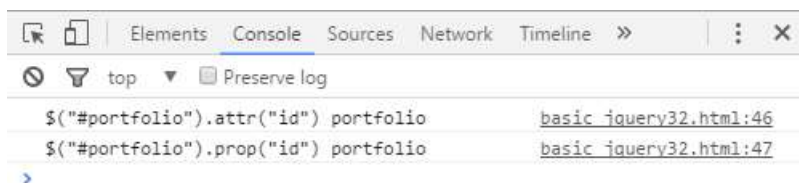
### JavaScript

```
38 <script>
39 $(function(){
40     $("body").append('<a id="link" href="#page1">page1 </a>');
41
42     console.log($("#portfolio").attr("id"), $("#portfolio").attr("id"));
43     console.log($("#portfolio").prop("id"), $("#portfolio").prop("id"));
44 });
45 </script>
```

40행 : `$("#body").append('<a id="link" href="#page1">page1 </a>');`  
body 요소에 원하는 노드를 추가합니다.

42행 : `console.log($("#portfolio").attr("id"), $("#portfolio").attr("id"));`  
portfolio id 요소의 `attr("id")`를 로그로 확인해 봅시다. 'portfolio'가 출력됩니다.

43행 : `console.log($("#portfolio").prop("id"), $("#portfolio").prop("id"));`  
portfolio 아이디의 `prop("id")`를 로그로 확인해 봅시다. 마찬가지로 'portfolio'가 출력됩니다.  
jQuery 요소에 대해서는 `attr()` 명령어와 `prop()` 명령어는 거의 용도가 비슷합니다.



예제의 코드를 바꾸어 보겠습니다.

#### JavaScript

```
38 <script>
39 $(function(){
40     $("body").append('<a id="link" href="#page1">page1 </a>');
41
42     console.log($("#link").attr("href"), $("#link").attr("href"));
43     console.log($("#link").prop("href"), $("#link").prop("href"));
44 });
45 </script>
```

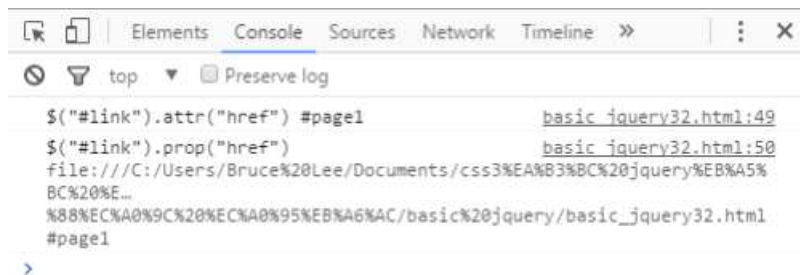
42행 : `console.log($("#link").attr("href"), $("#link").attr("href"));`

추가된 link id 요소의 `attr("href")`를 로그로 확인해 봅시다. '#page1'이 출력됩니다.

43행 : `console.log($("#link").prop("href"), $("#link").prop("href"));`

추가된 link id 요소의 `prop("href")`를 로그로 확인해 봅시다. 긴 URL을 출력 결과로 확인할 수 있습니다.

DOM으로서의 고유한 속성일 경우에 `attr()` 명령어와 `prop()` 명령어는 다릅니다.



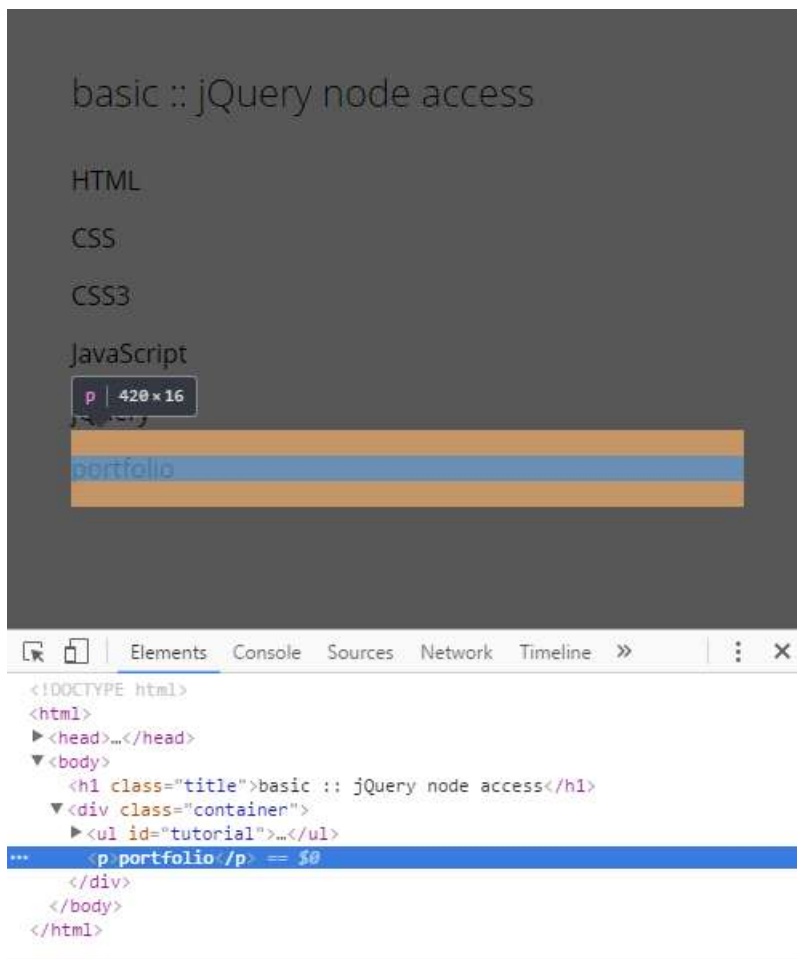


### 1-4-3) removerAttr()

속성의 값을 제거합니다. removerAttr() 명령어는 속성 값만 제거하는 것이 아니라 속성 자체를 삭제해 줍니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery33.html



#### JavaScript

```
38 <script>
39 $(function(){
40     $("#portfolio").removeAttr("id");
41 });
42 </script>
```

40행 : `$("#portfolio").removeAttr("id");`  
portfolio id 요소의 id 속성을 제거합니다.

#### 1-4-4) removeProp()

removeProp() 명령어는 속성 값만 제거합니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery34.html

#### JavaScript

```
38     <script>
39     $(function(){
40         $("#portfolio").removeProp("id");
41     });
42     </script>
```

40행 : `$("#portfolio").removeProp("id");`  
portfolio id 요소의 id 속성 값도 제거합니다. id 값은 'undefined'로 지정됩니다.



## 1-5) 치수(Dimensions)

요소에 대한 위치와 크기를 읽고 정하는 명령어입니다.

| 명령어                  | 설명                                                                                     |
|----------------------|----------------------------------------------------------------------------------------|
| width()              | 요소의 가로 크기를 확인합니다.                                                                      |
| height()             | 요소의 세로 크기를 확인합니다.                                                                      |
| innerWidth()         | padding 값을 포함한 가로 크기를 확인합니다.                                                           |
| innerHeight()        | padding 값을 포함한 세로 크기를 확인합니다.                                                           |
| outerWidth(boolean)  | 매개변수 Boolean의 값이 true일 경우 요소의 margin을 포함한 가로 크기를, false일 경우에는 margin을 제외한 가로 값을 확인합니다. |
| outerHeight(boolean) | 매개변수 Boolean의 값이 true일 경우 요소의 margin을 포함한 세로 크기를, false일 경우에는 margin을 제외한 세로 값을 확인합니다. |

### 시작 파일

sample/basic\_jquery/start/basic\_jquery35.html

```
$("#body").width() : 420px  
  
$("#body").height() : 256px  
  
$("#body").innerWidth() : 460px  
  
$("#body").innerHeight() : 296px  
  
$("#body").outerWidth(true) : 500px  
  
$("#body").outerHeight(true) : 336px
```

## JavaScript

```
38 <script>
39 $(function(){
40     $(window).resize(function(){
41         $("body").html(
42             ' $("body").width() : '+$("body").width()+"px"+
43             "<br><br><br>" +
44             ' $("body").height() : '+$("body").height()+"px"+
45             "<br><br><br>" +
46             ' $("body").innerWidth() : '+$("body").innerWidth()+"px"+
47             "<br><br><br>" +
48             ' $("body").innerHeight() : '+$("body").innerHeight()+"px"+
49             "<br><br><br>" +
50             ' $("body").outerWidth(true) : '+$("body").outerWidth(true)+"px"+
51             "<br><br><br>" +
52             ' $("body").outerHeight(true) : '+$("body").outerHeight(true)+"px"
53         );
54     });
55     $(window).trigger("resize");
56 });
57 </script>
```

40행 : `$(window).resize(function(){`  
브라우저를 크기 조정하면 발생하는 이벤트입니다.

41행 : `$("body").html(`  
body 요소의 내용을 새로 작성합니다.

43행 : `' $("body").width() : '+$("body").width()+"px"+`  
현재 body 요소의 가로 크기를 확인합니다.

만일 `$("body").width()`가 360px이라면  
`$("body").innerWidth()`은 400px, `$("body").outerWidth(true)`는 440px이 될 것입니다.

이유를 설명한다면,  
`$("body").width()`는 margin과 padding이 포함되지 않은 실제 영역 크기입니다.  
`$("body").innerWidth()`는 padding이 포함된 가로 크기 영역입니다.  
`$("body").outerWidth(true)`는 margin과 padding이 포함된 가로 크기 영역입니다.

body 요소의 margin과 padding은 각각 20px입니다.

따라서 `$("#body").width()`가 360px이라면,  
`$("#body").innerWidth()`은 400px, `$("#body").outerWidth(true)`는 440px이 될 것입니다.

`$("#body").outerWidth(false)`는 padding만 포함된 가로 크기 영역이며, margin은 포함되지 않습니다.

55행 : `$(window).trigger("resize");`

페이지가 로딩이 완료되면 `resize` 이벤트가 발생되지 않습니다. 페이지가 로딩이 완료되었을 때에 `resize` 이벤트가 발생될 수 있도록 이벤트를 트리거 (`trigger()`)합니다.

이벤트 트리거에 대한 내용은 이벤트 부분에서 다시 공부하기로 하겠습니다.

## 1-6) 스타일 지정(CSS)

CSS를 통해서 스타일을 조작하는 것처럼 jQuery는 CSS을 제어할 수 있습니다. 사실 이전부터 여러 번 확인했던 명령어입니다.

아래의 표는 CSS를 통해서 스타일을 지정하는 방식입니다.

```
.html {  
    background-color: #666;  
}
```

jQuery에서도 CSS 속성을 적용할 수 있습니다. 용법은 아래와 같습니다.

1가지 스타일을 적용할 경우입니다.

```
$(".html").css("background-color", "#666");
```

2가지 이상의 스타일을 적용할 경우입니다.

```
$(".html").css({ "background-color": "#666", color: "#8ac007" });
```

스타일을 지정하는 jQuery 명령어들은 아래와 같습니다.

| 명령어           | 설명                        |
|---------------|---------------------------|
| css()         | css() 요소 값을 알아내거나 지정합니다.  |
| addClass()    | 특정한 클래스 요소를 추가합니다.        |
| hasClass()    | 특정한 클래스를 있는지를 찾습니다.       |
| removeClass() | 특정한 클래스를 요소에서 제거합니다.      |
| toggleClass() | 특정한 클래스의 추가 제거를 한 번에 합니다. |

### 1-6-1) css()

Query를 통해 CSS의 속성을 알아내거나 정해줍니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery36.html

basic :: jQuery style

HTML

CSS

CSS3

JavaScript

jQuery

portfolio

#### JavaScript

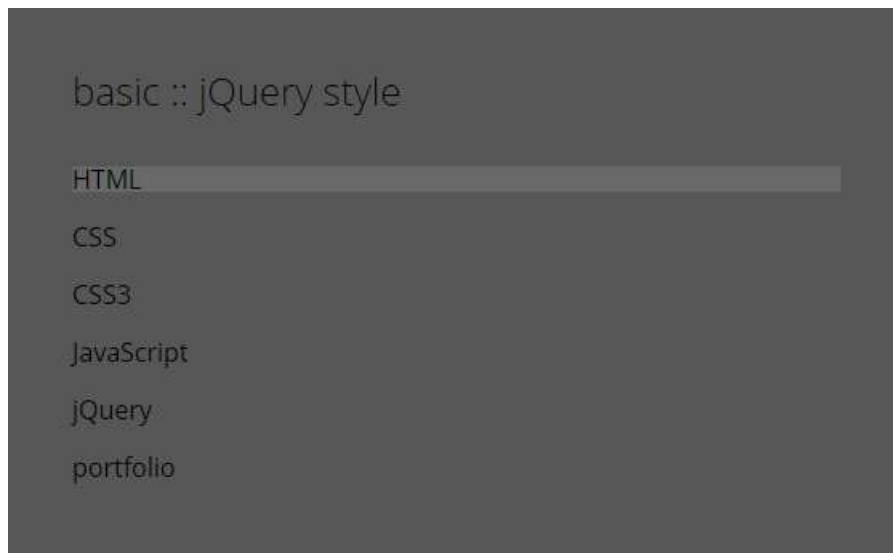
```
38 <script>
39 $(function(){
40     console.log($(".html").css("background-color") :', $(".html").css("background-color"));
41 });
42 </script>
```

40행 : `console.log($(".html").css("background-color") :', $(".html").css("background-color"));`

html 클래스의 바탕 색상 속성 값을 확인합니다. 'rgba(0, 0, 0, 0)'은 색상 값이 적용되기 전의 값입니다.



예제의 코드를 바꾸어 보겠습니다.



#### JavaScript

```
38 <script>
39 $(function(){
40     $(".html").css("background-color", "#666");
41 });
42 </script>
```

40행 : \$(".html").css("background-color", "#666");

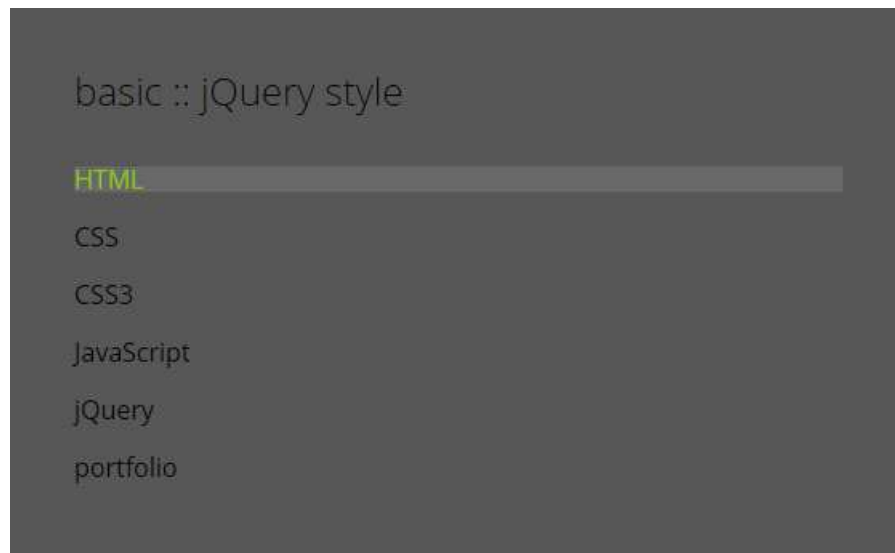
html 클래스의 바탕 색상을 #666 색상으로 바꾸어 줍니다.

이전에도 언급된 바가 있었지만 속성이 두 개의 단어가 결합된 경우는 반드시 작은따옴표나 큰따옴표를 이용해서 문자열 형태로 지정해 주어야 합니다.

두 개의 단어가 결합된 속성에는 background-color, background-position, background-repeat과 같은 것들이 있습니다.



css() 명령어를 사용해서 2개의 스타일을 변경해 보겠습니다.

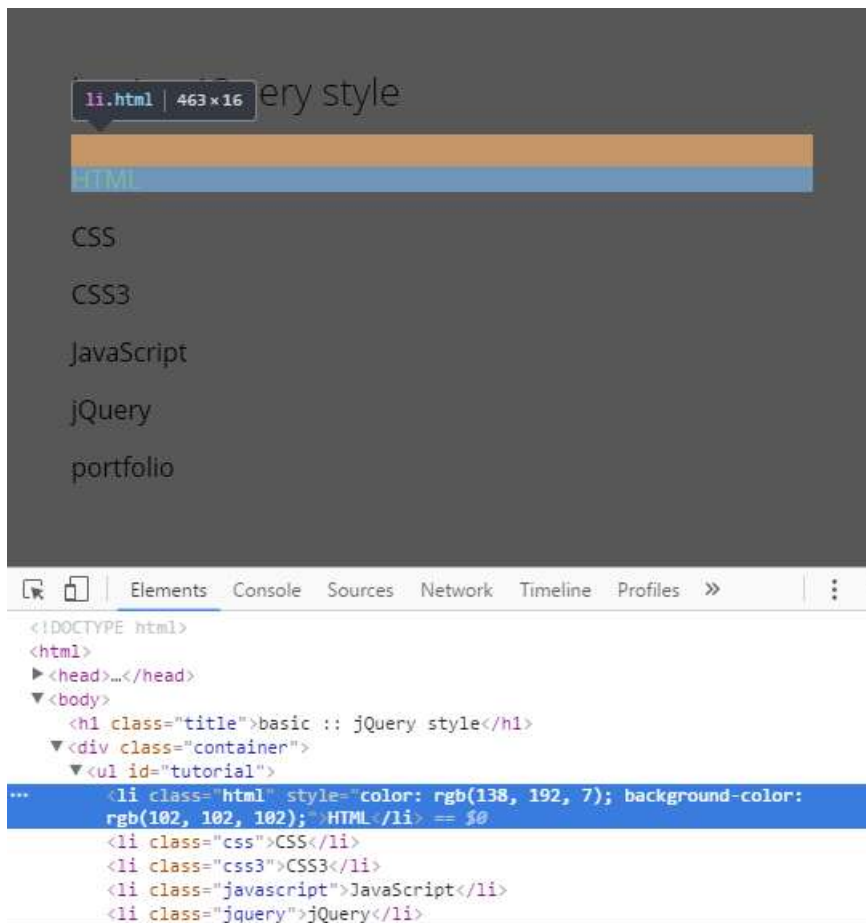


#### JavaScript

```
38 <script>
39 $(function(){
40     $(".html").css({ "background-color": "#666", color: "#8ac007" });
41 });
42 </script>
```

40행 : \$(".html").css({ "background-color": "#666", color: "#8ac007" });  
html 클래스의 바탕 색상과 글자 색상에 대한 스타일을 변경합니다.

css() 명령어를 통해서 적용된 스타일은 Inline Style 방식으로 적용되는 것을 확인할 수 있습니다.

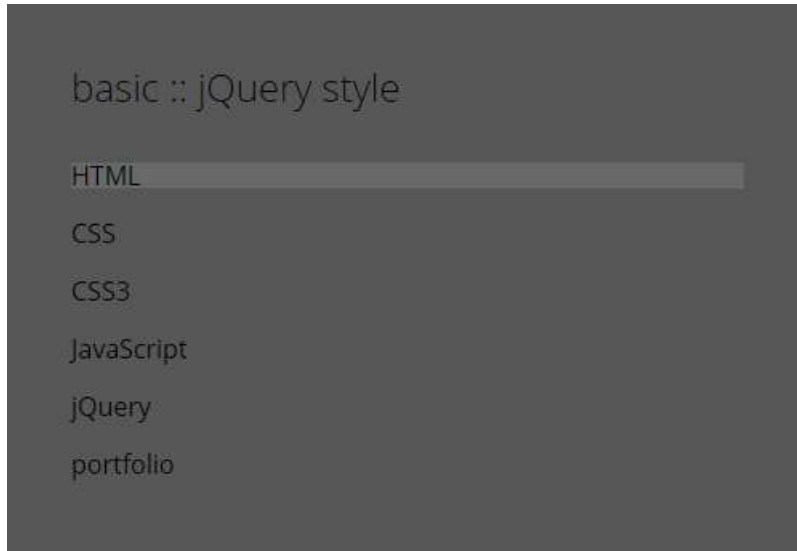


jQuery로 추가된 스타일은 어떤 CSS보다 우선 적용 :

jQuery의 `css()` 명령어를 통해 적용된 스타일은 일반적으로 적용된 스타일보다 항상 우선 적용됩니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery37.html



#### HTML

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <meta http-equiv="X-UA-Compatible" content="IE=edge">
6 <meta name="viewport" content="width=device-width, initial-scale=1.0,
  minimum-scale=1.0">
7 <title>basic :: jQuery Basic Study</title>
8 <link rel="stylesheet"
  href="http://fonts.googleapis.com/css?family=Open+Sans:400,300,300italic,400italic,600,6
  00italic,700,700italic,800,800italic">
9 <style>
10 body {
11     margin: 20px;
12     padding: 20px;
13     line-height: 1;
14     font-family: "Open Sans", sans-serif;
15     font-size: 1em;
```

```
16         background-color: #555;
17         color: #000;
18     }
19     ul, li {
20         margin: 0;
21         padding: 0;
22         list-style: none;
23     }
24     li {
25         margin-top: 20px;
26     }
27     .title {
28         margin: 0;
29         padding: 0;
30         font-size: 1.5em;
31         font-weight: 300;
32     }
33     .container {
34         margin-top: 35px;
35     }
36     .selected {
37         background-color: #666;
38     }
39     .html {
40         background-color: #8ac007;
41     }
42     </style>
43     <script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
44     <script>
45     $(function(){
46         $(".html").css("background-color", "#666");
47     });
48     </script>
49     </head>
50
51     <body>
52     <h1 class="title">basic :: jQuery node Access</h1>
53     <div class="container">
54         <ul id="tutorial">
55             <li class="html">HTML</li>
56             <li class="css">CSS</li>
57             <li class="css3">CSS3</li>
58             <li class="javascript">JavaScript</li>
59             <li class="jquery">jQuery</li>
```

```

60         <li class="blank"></li>
61     </ul>
62     <p id="portfolio">portfolio</p>
63 </div>
64 </body>
65 </html>

```

40행 : `background-color: #8ac007;`

CSS를 통해서 html 클래스에 대한 바탕 색상을 바꾸어줍니다.

46행 : `$(".html").css("background-color", "#666");`

jQuery를 통해서 html 클래스에 대한 바탕 색상을 바꾸어줍니다.

jQuery의 `css()` 명령어를 통해 적용된 스타일은 일반적으로 적용된 스타일보다 항상 우선 적용됩니다. 이는 Inline Style로 추가되어서 CSS로 스타일이 된 내용이 의미가 없어질 수도 있습니다.

이번 예제에서는 CSS로 적용된 스타일은 무의미한 내용이 됩니다.

```

.html {
    background: #8ac007;
}

```

대신 jQuery `css()` 명령어로 추가된 스타일이 적용되는 것입니다. 아래의 그림은 스크립트가 적용된 후의 브라우저의 Debugger 창입니다. `rgb(102, 102, 102)`는 `#666`과 같은 색상입니다.



실무 작업을 하면서도, CSS로 스타일하는 경우와 jQuery `css()` 명령어를 사용하는 경우 스타일이 상충하는 경우가 생길 수 있습니다.

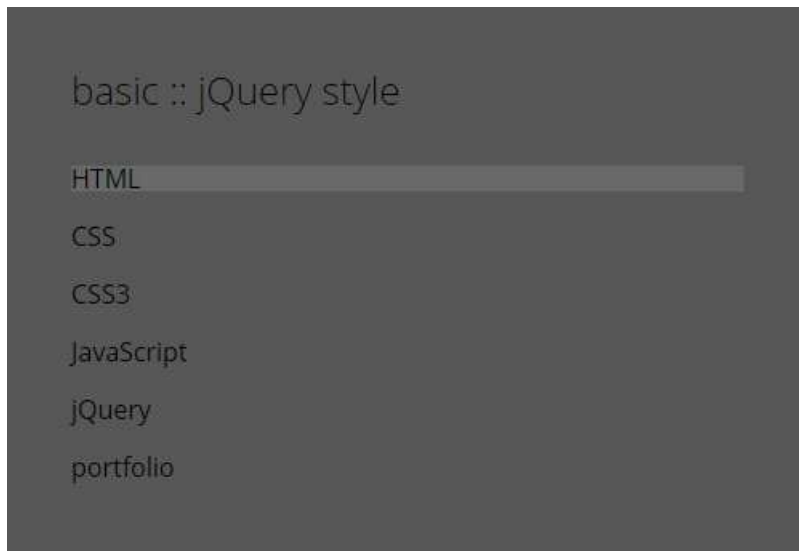
이러한 CSS의 오류를 막기 위해서 클래스를 추가, 제거하는 방식 (`addClass()`, `removeClass()`)을 추천합니다. 클래스를 추가하는 방식에 대해서는 다음 과정에서 진행됩니다.

## 1-6-2) addClass()

원하는 대상 요소에 class 속성을 추가합니다.

### 시작 파일

sample/basic\_jquery/start/basic\_jquery38.html



### JavaScript

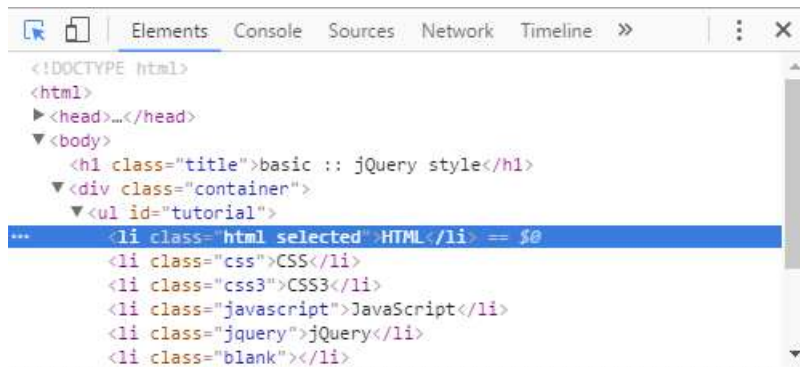
```
38 <script>
39 $(function(){
40     $(".html").addClass("selected");
41 });
42 </script>
```

40행 : \$(".html").addClass("selected");

html 클래스 요소에 selected 클래스를 추가합니다.

selected 클래스의 내용은 아래와 같습니다.

```
.selected {  
    background: #666;  
}
```

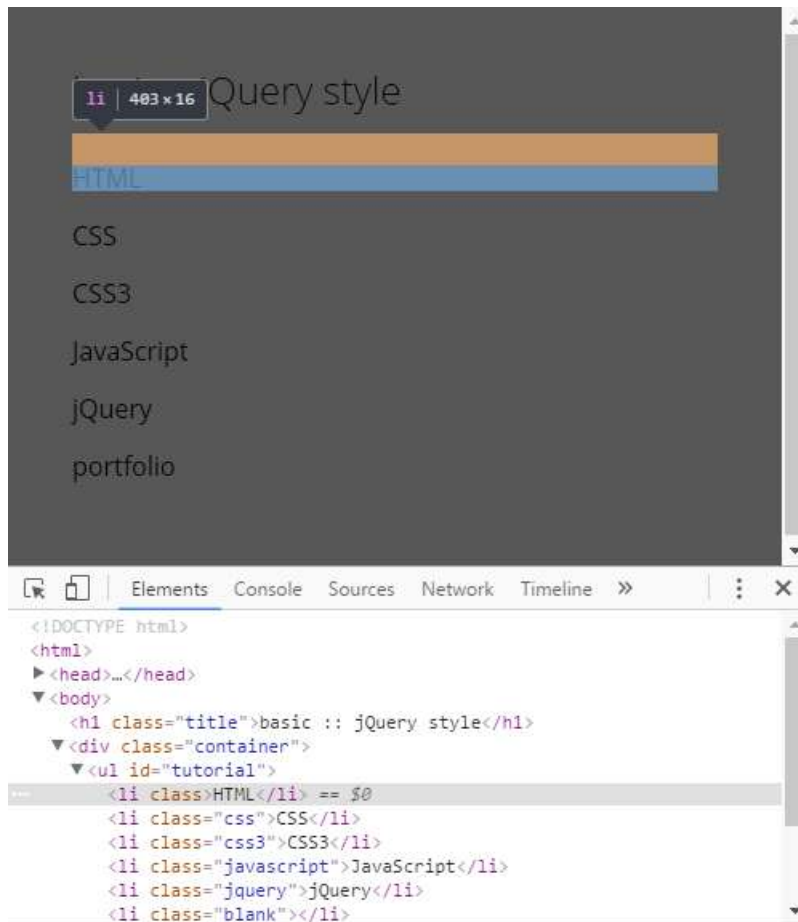


### 1-6-3) removeClass()

원하는 요소에 class 속성을 제거합니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery39.html



#### JavaScript

```
38 <script>
39 $(function(){
40     $(".html").removeClass("html");
41 });
42 </script>
```

40행 : \$(".html").removeClass("html");  
html 클래스 요소에 대해 html 클래스를 제거합니다.



Chrome 브라우저에서 [Elements] 탭을 눌러보면, class 속성에 대한 속성 값이 제거된 것을 확인할 수 있습니다. class 속성 자체를 제거할 경우에는 removeAttr() 명령어를 사용합니다.

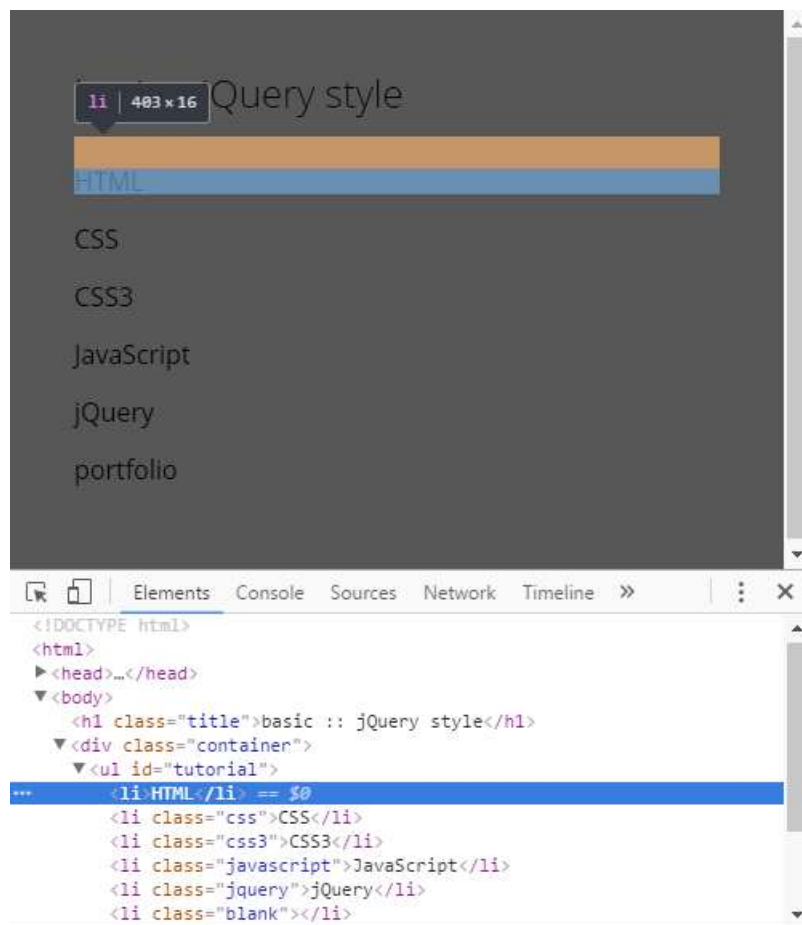
예제의 코드를 바꾸어 보겠습니다.

#### JavaScript

```
38 <script>
39 $(function(){
40     $(".html").removeAttr("class");
41 });
42 </script>
```

40행 : \$(".html").removeAttr("class");

html 클래스 요소에 대해 class 속성을 제거합니다.

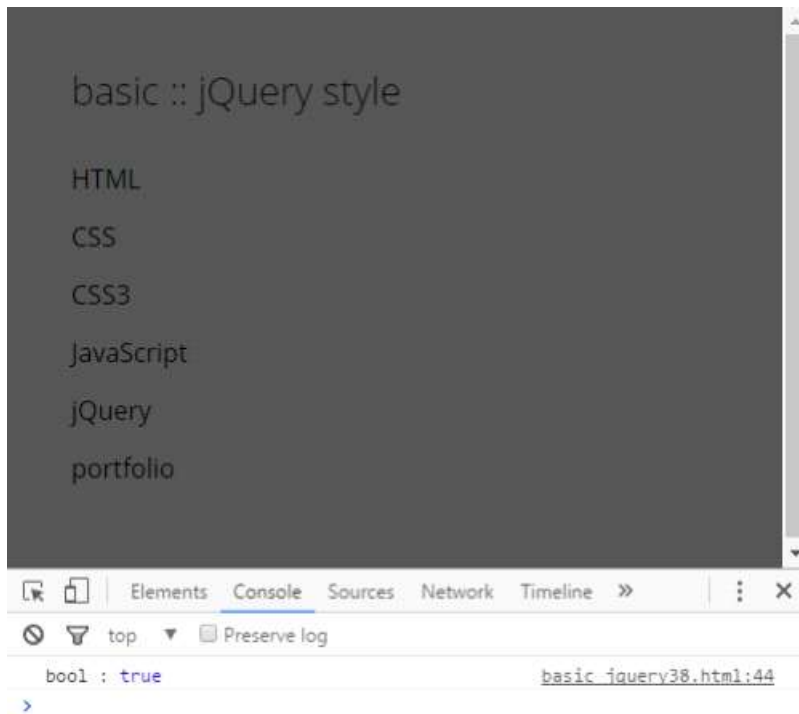


#### 1-6-4) hasClass()

클래스가 있는지 없는지를 확인합니다.

##### 시작 파일

sample/basic\_jquery/start/basic\_jquery40.html



##### JavaScript

```
38 <script>
39 $(function(){
40     var bool=$(".html").hasClass("html");
41     console.log("bool :", bool);
42 });
43 </script>
```

40행 : `var bool=$(".html").hasClass("html");`

html 클래스가 있는지 없는지 여부를 bool 변수에 대입합니다.

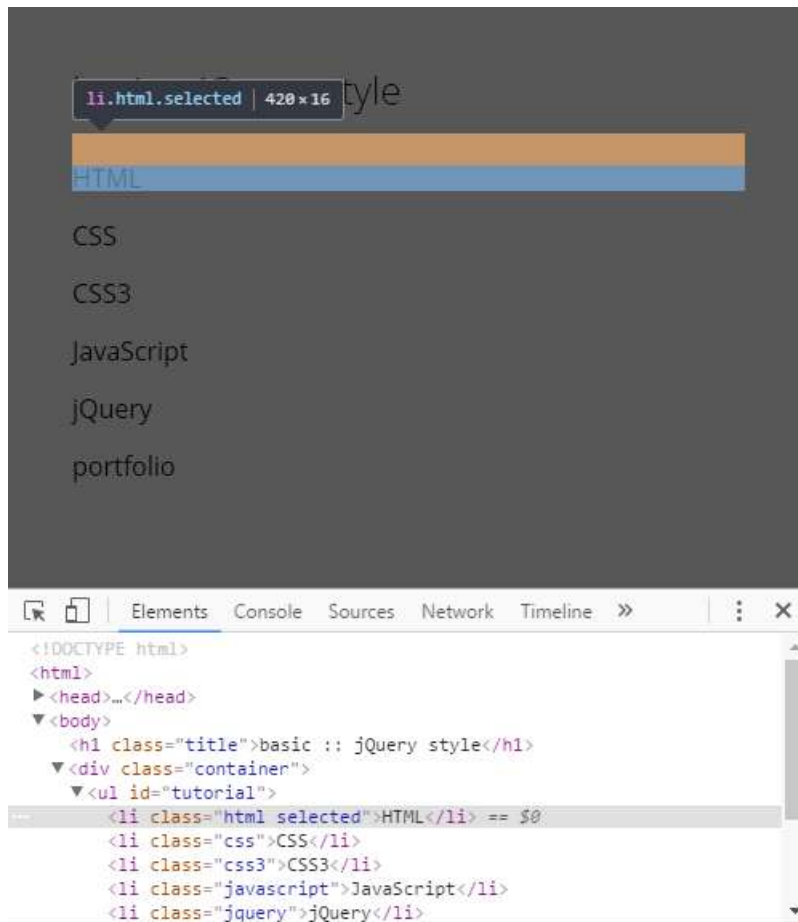
bool 변수의 값은 참(true)으로 로그로써 확인됩니다.

1-6-5) toggleClass() :

특정한 클래스의 추가와 제거를 토글 방식으로 적용합니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery41.html



#### JavaScript

```
38 <script>
39 $(function(){
40     $(".html").click(function(){
41         $(this).toggleClass("selected");
42     });
43 });
44 </script>
```

40행 : \$(".html").click(function(){

html 클래스 요소를 클릭할 때 발생하는 이벤트입니다.

이벤트에 대한 내용은 다음 과정에서 배울 수 있습니다.

41행 : \$(this).toggleClass("selected");

\$(this)는 현재 이벤트 대상으로 html 클래스를 의미합니다. 이벤트가 발생된 대상을 지정할 때 사용하는 참조 방식입니다.

클릭 이벤트가 발생되면, selected 클래스가 있는 경우는 selected 클래스가 제거되며 없는 경우는 selected 클래스가 추가됩니다.

## 1-7) 이벤트(Event)

이벤트 (Event)란 웹 페이지를 방문한 사용자가 반응하는 동작을 의미합니다.

이벤트 핸들러 (Event Handler)는 이벤트가 발생할 경우 처리되는 실제 내용을 담고 있는 명령어입니다.

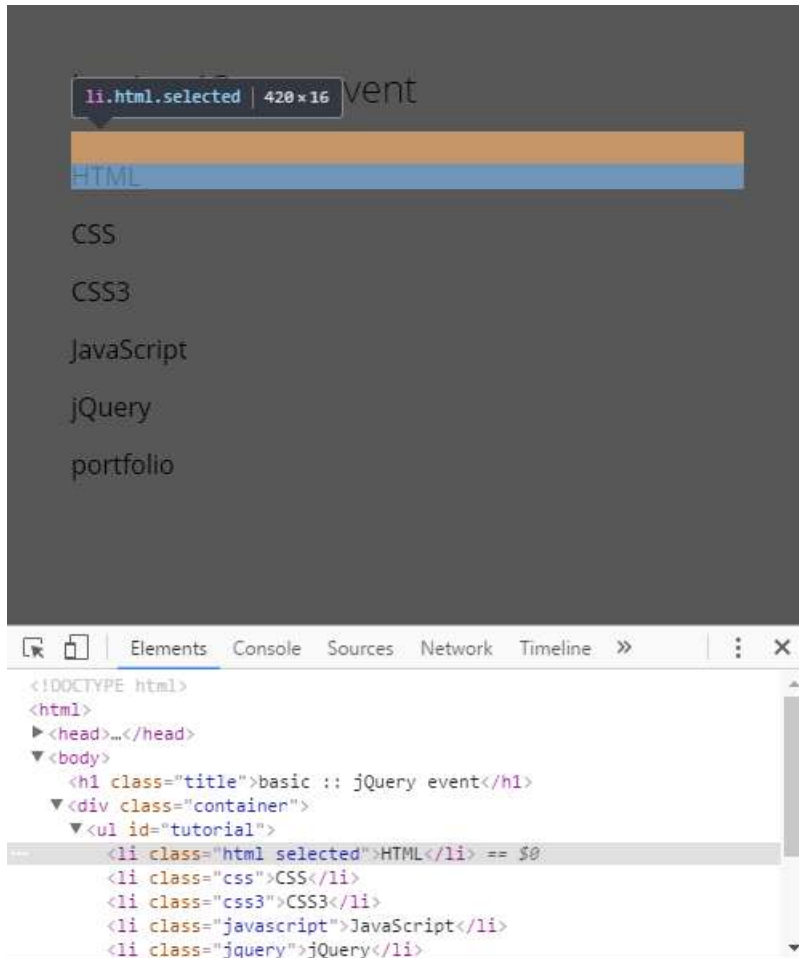
### 1-7-1) 이벤트 핸들러

아래의 표는 jQuery에서 사용되는 이벤트 핸들러 명령어들에 관련된 내용입니다.

| 명령어               | 설명                                                                     |
|-------------------|------------------------------------------------------------------------|
| .bind()           | 이벤트를 생성합니다.                                                            |
| .unbind()         | 이벤트를 제거합니다.                                                            |
| .on()             | 이벤트를 생성합니다.                                                            |
| .off()            | on() 명령어로 생성한 이벤트를 제거합니다.                                              |
| .one()            | 기본적인 기능은 bind()와 같지만, 이벤트가 발생하면 스스로 발생된 이벤트를 제거합니다. 즉 한 번만 이벤트가 발생합니다. |
| .trigger()        | 이벤트를 수동으로 호출합니다.                                                       |
| .triggerHandler() | 이벤트를 수동으로 호출합니다. trigger() 명령어와 유사합니다.                                 |

## 시작 파일

sample/basic\_jquery/start/basic\_jquery42.html



## JavaScript

```
38 <script>
39 $(function(){
40     $(".html").bind("click", function(){
41         $(this).addClass("selected");
42         $(this).unbind("click");
43     });
44 });
45 </script>
```

43행 : \$(".html").bind("click", function(){  
html 클래스에 click 이벤트를 추가합니다.

44행 : `$(this).addClass("selected");`

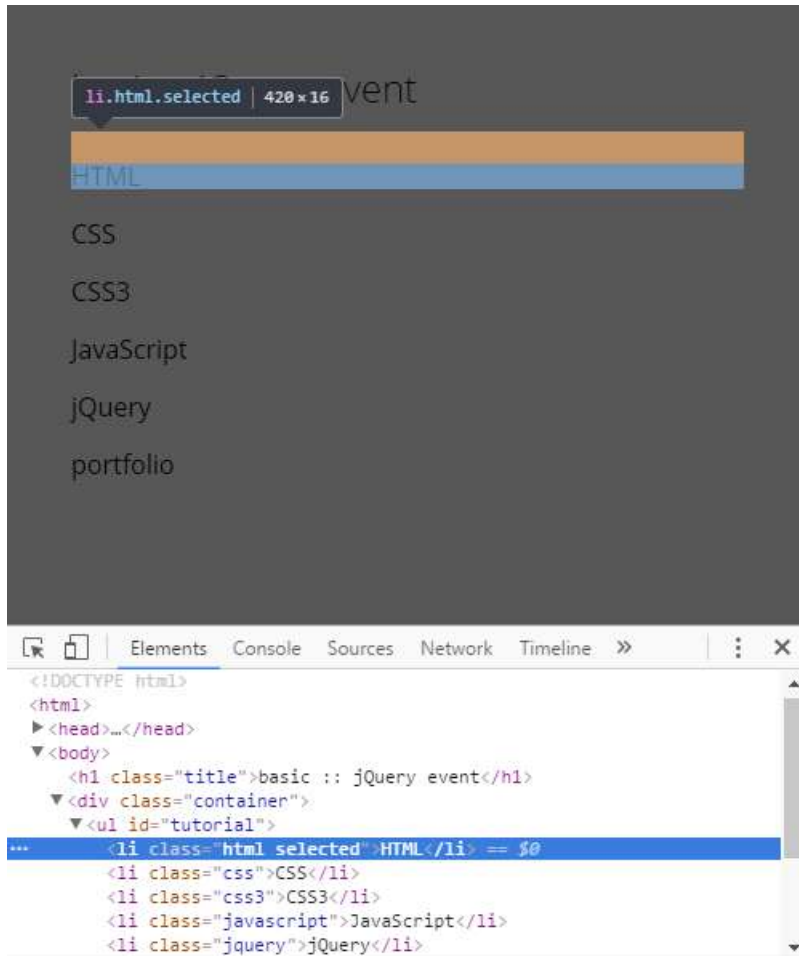
현재 대상 요소에 `selected` 클래스를 추가합니다.

45행 : `$(this).unbind("click");`

현재 대상 요소에 등록된 `click` 이벤트를 제거합니다. 즉 한 번만 이벤트가 발생되고 더 이상은 `click` 이벤트가 발생되지 않습니다.

## 시작 파일

sample/basic\_jquery/start/basic\_jquery43.html



## JavaScript

```
38 <script>
39 $(function(){
40     $(".html").on("click", function(){
41         $(this).toggleClass("selected");
42     });
43 });
44 </script>
```

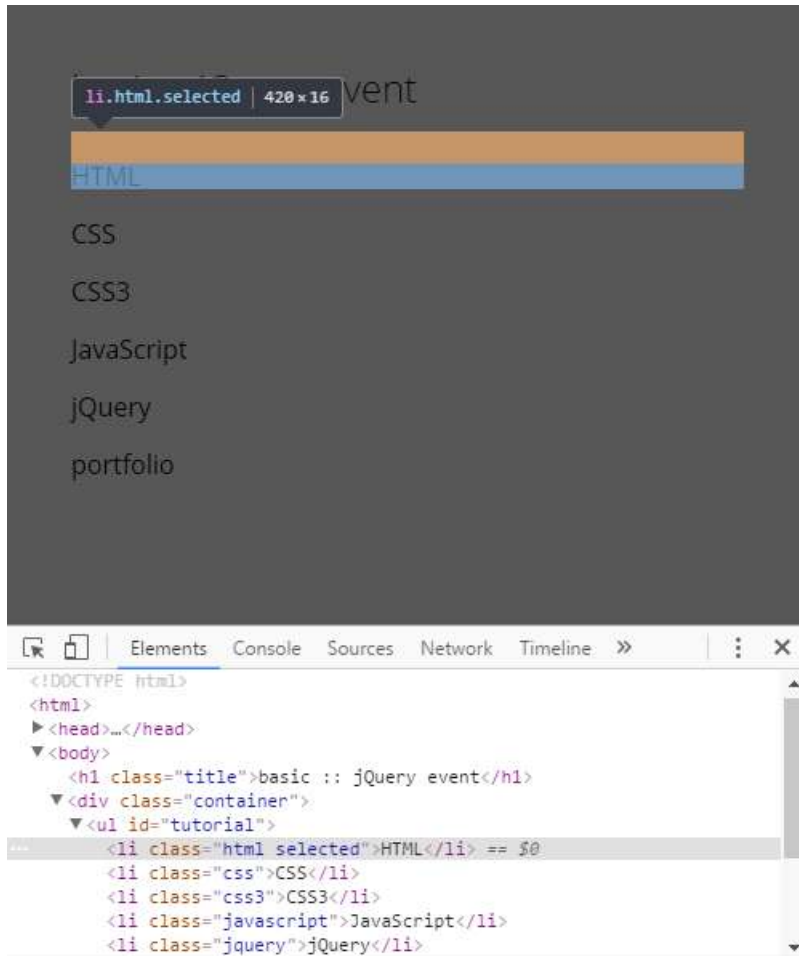
43행 : \$(".html").on("click", function(){

on() 명령어는 bind() 명령어와 마찬가지로 이벤트를 추가합니다.



## 시작 파일

sample/basic\_jquery/start/basic\_jquery44.html



## JavaScript

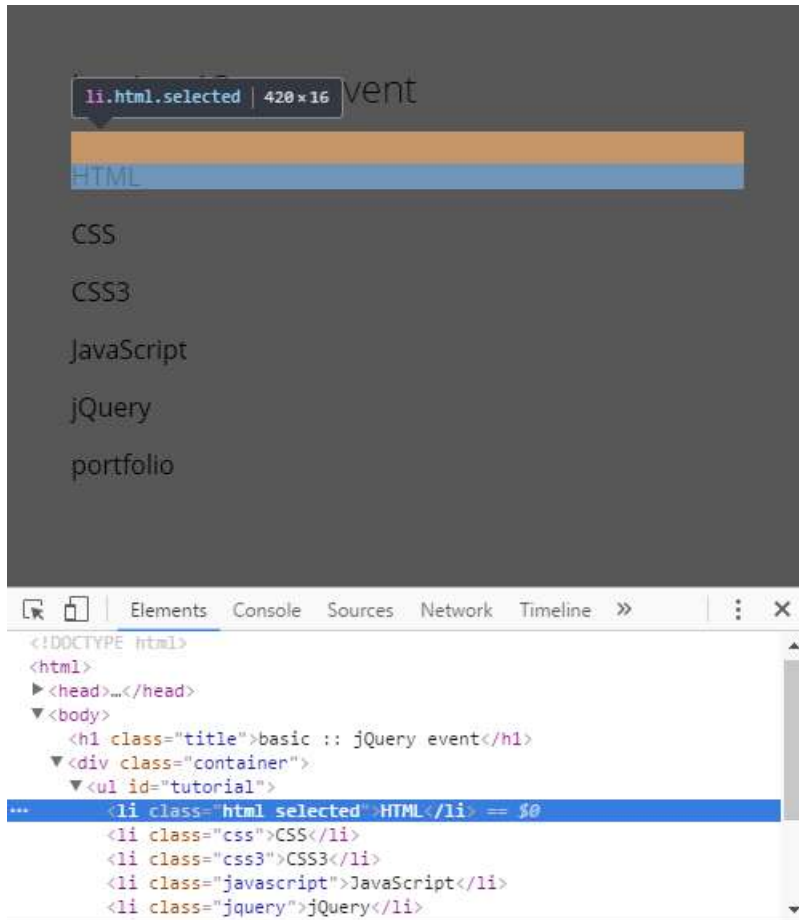
```
38 <script>
39 $(function(){
40     $(".html").on("click", function(){
41         $(this).addClass("selected");
42         $(this).off("click");
43     });
44 });
45 </script>
```

45행 : \$(this).off("click");

off() 명령어는 unbind() 명령어와 마찬가지로 이벤트를 제거합니다.

## 시작 파일

sample/basic\_jquery/start/basic\_jquery45.html



## JavaScript

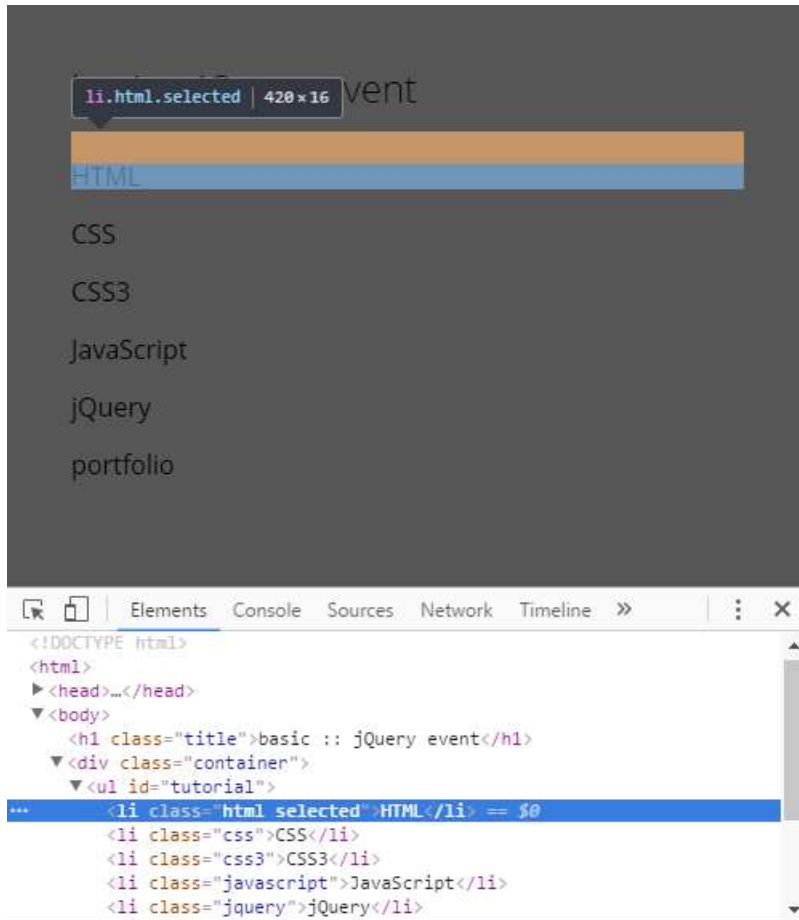
```
38 <script>
39 $(function(){
40     $(".html").one("click", function(){
41         $(this).addClass("selected");
42     });
43 });
44 </script>
```

43행 : \$(".html").one("click", function(){

click 이벤트가 한 번만 발생합니다. html 클래스를 두 번 이상 눌러도 이벤트는 발생되지 않습니다.

## 시작 파일

sample/basic\_jquery/start/basic\_jquery46.html



## JavaScript

```
41 <script>
42 $(function(){
43     $(".html").on("click", function(){
44         $(this).toggleClass("selected");
45     });
46     $(".css").on("click", function(){
47         $(".html").trigger("click");
48     });
49 });
50 </script>
```

47행 : \$(".html").trigger("click");

css 클래스를 클릭하면, html 클래스의 click 이벤트를 트리거 (trigger)합니다.

trigger() 명령어는 이벤트 조건이 맞춰지지 않아도 강제로 이벤트를 실행시키는 명령어입니다.

즉 css 클래스를 클릭하게 되면, html 클래스 대상을 클릭한 것과 같은 결과가 확인됩니다.

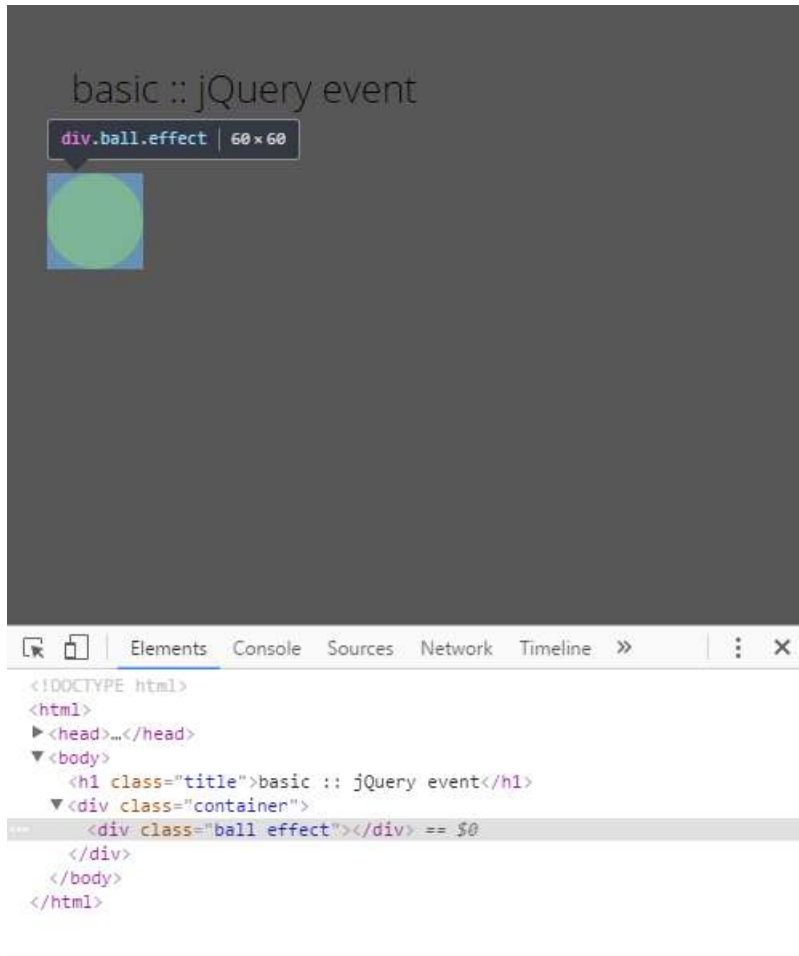
### 1-7-2) 마우스 이벤트

아래의 표는 마우스와 관련된 내용입니다.

| 명령어          | 설명                                            |
|--------------|-----------------------------------------------|
| click()      | 요소에 마우스 포인터로 눌렀다가 떴을 때에 발생합니다.                |
| dblclick()   | 요소를 더블 클릭했을 때 발생합니다.                          |
| hover()      | mouseenter와 mouseleave 이벤트를 한 번에 지정하는 이벤트입니다. |
| mousedown()  | 마우스를 눌렀다가 떴을 때에 발생합니다.                        |
| mouseenter() | 요소에 마우스가 진입했을 때 발생합니다.                        |
| mouseleave() | 마우스가 요소에서 벗어났을 때에 발생합니다.                      |
| mouseover()  | 요소 영역에 마우스를 올려놓았을 때에 발생합니다.                   |
| mouseout()   | 요소에서 마우스 포인터가 떠났을 때 발생합니다.                    |
| mouseup()    | 마우스 포인터를 노드에 올려놓고 마우스 버튼을 눌렀다 떴을 때 발생합니다.     |
| toggle()     | click 이벤트가 발생될 때마다 실행될 함수들을 순차적으로 실행합니다.      |

## 시작 파일

sample/basic\_jquery/start/basic\_jquery47.html



## JavaScript

```
49 $(function(){
50     $(".ball").click(function(){
51         $(this).toggleClass("effect");
52     });
53 });
54 </script>
```

51행 : \$(".ball").click(function(){  
ball 클래스를 클릭하면 이벤트가 발생합니다.

더블 클릭했을 때의 이벤트 발생은 아래와 같습니다.

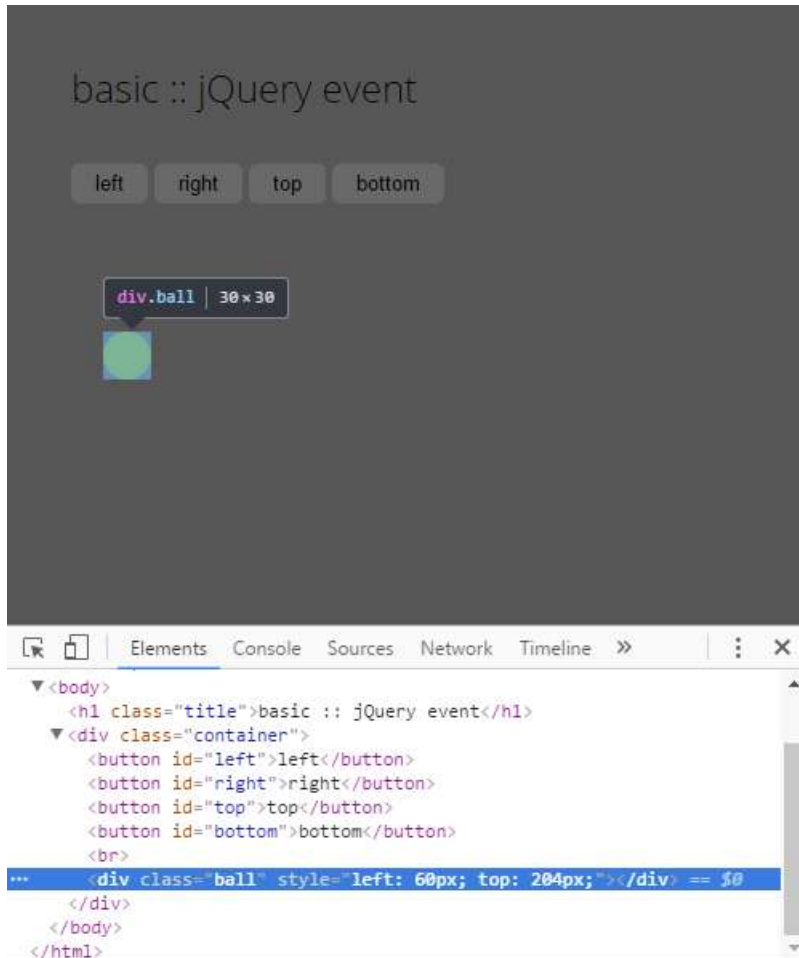
```
$(".ball").dblclick(function(){  
    $(this).toggleClass("effect");  
});
```

ball 클래스를 클릭하면 effect 클래스가 추가됩니다. effect 클래스의 내용은 아래와 같습니다.

```
.effect {  
    transform: scale(2);  
}
```

## 시작 파일

sample/basic\_jquery/start/basic\_jquery48.html



```
53 <script>
54 $(function(){
55     $("#left").click(function(){
56         $(".ball").css({ left: "-=10px" });
57     });
58     $("#right").click(function(){
59         $(".ball").css({ left: "+=10px" });
60     });
61     $("#top").click(function(){
62         $(".ball").css({ top: "-=10px" });
63     });
64     $("#bottom").click(function(){
```



```
65         $(".ball").css({ top: "+=10px" });
66     });
67 });
68 </script>
```

56행 : \$(".ball").css({ left: "-=10px" });

ball 클래스의 left 스타일을 변경합니다. '-=' 혹은 '+=' 방식으로 상대 좌표를 적용할 수 있습니다.

ball 클래스의 left 속성을 절대 좌표 위치인 10px로 위치합니다.

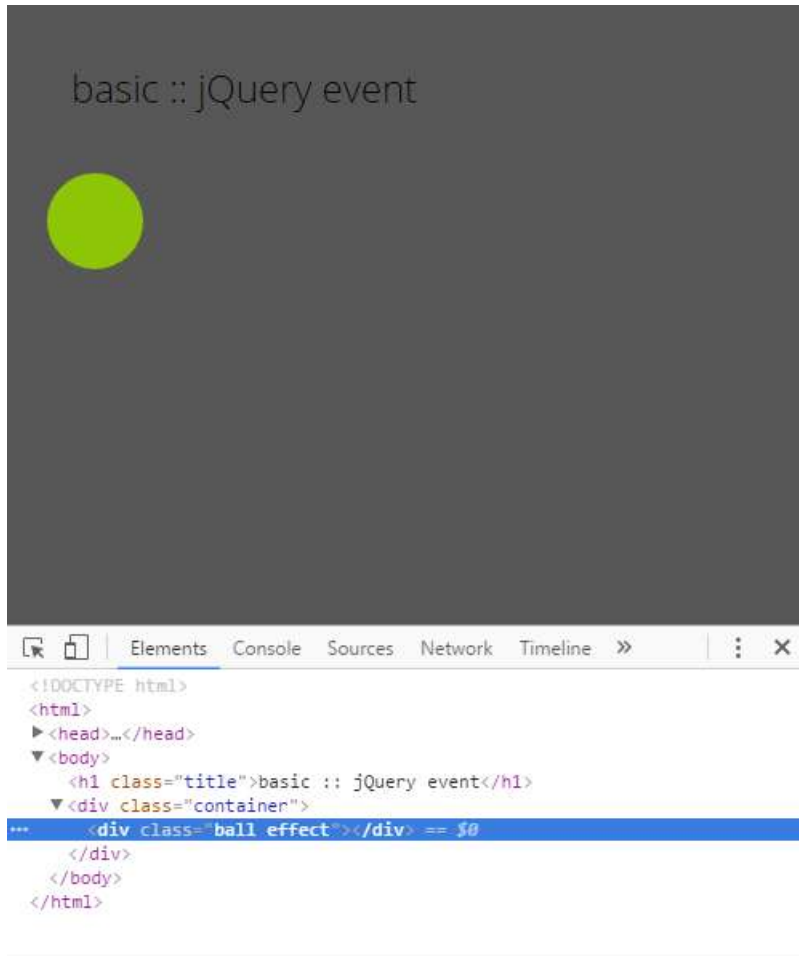
```
$(".ball").css({ left: "10px" });
```

ball 클래스의 위치를 우측으로 10px 이동합니다. 현재 위치에서 상대 위치를 기준하여 우측으로 10px 이동하는 것입니다.

```
$(".ball").css({ left: "+=10px" });
```

## 시작 파일

sample/basic\_jquery/start/basic\_jquery49.html



## JavaScript

```
49  <script>
50  $(function(){
51      $(".ball").mouseenter(function(){
52          $(this).addClass("effect");
53      });
54      $(".ball").mouseleave(function(){
55          $(this).removeClass("effect");
56      });
57  });
58  </script>
```

51행 : \$(".ball").mouseenter(function(){

ball 클래스에 마우스가 진입했을 때 이벤트가 발생합니다.

54행 : `$(".ball").mouseleave(function(){`

ball 클래스에 마우스가 벗어났을 때 이벤트가 발생합니다.

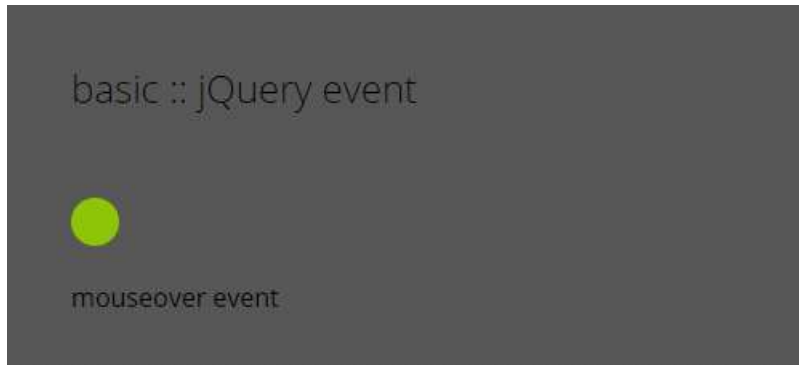
mouseenter, mouseleave 이벤트는 hover 이벤트로 바꿀 수 있습니다.

본문 예제를 아래와 같이 바꿀 수 있습니다.

```
$(".ball").hover(  
    function(){  
        $(this).addClass("effect");  
    },  
    function(){  
        $(this).removeClass("effect");  
    }  
);
```

## 시작 파일

sample/basic\_jquery/start/basic\_jquery50.html



## JavaScript

```
49 <script>
50 $(function(){
51     $("body").append('<div class="status">waiting event!</div>');
52
53     $(".ball").mouseover(function(){
54         $(".status").text("mouseover event!");
55     });
56     $(".ball").mouseout(function(){
57         $(".status").text("mouseout event!");
58     });
59     $(".ball").mouseup(function(){
60         $(".status").text("mouseup event!");
61     });
62 });
63 </script>
```

51행 : `$("body").append('<div class="status">waiting event!</div>');`

body 요소에 status 클래스를 추가합니다. 추가된 status 클래스 요소를 통해 ball 클래스에 적용된 이벤트를 확인할 수 있습니다.

53행 : `$(".ball").mouseover(function(){`

ball 클래스에 마우스를 올려놓았을 때에 이벤트가 발생합니다.

55행 : `$(".ball").mouseout(function(){`

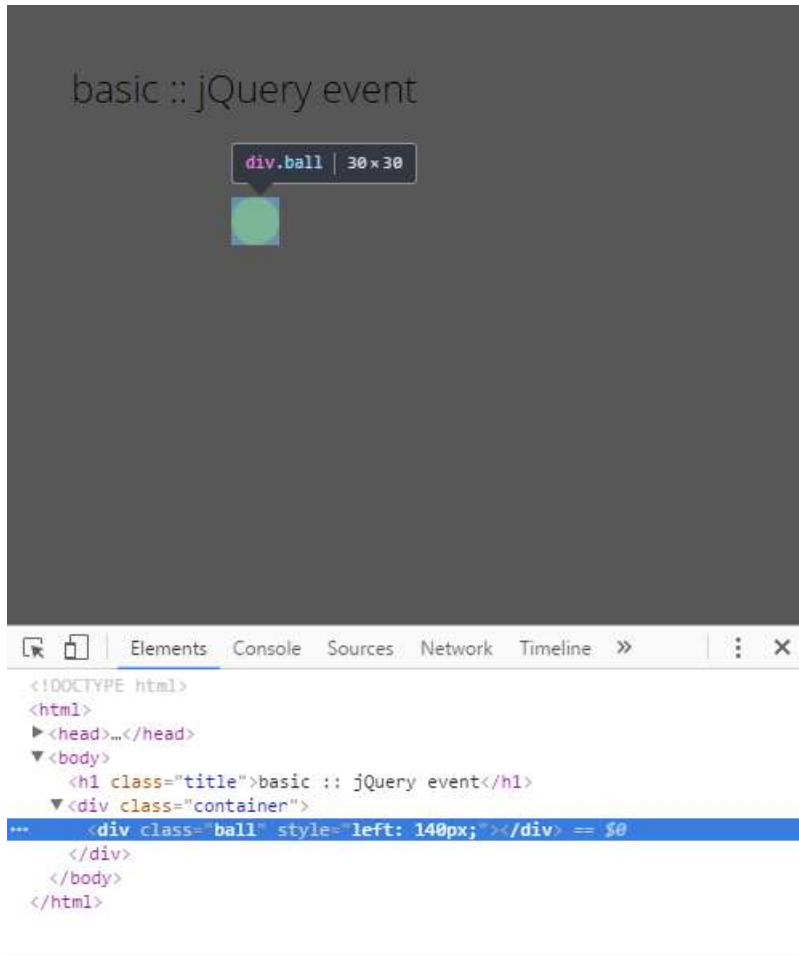
ball 클래스에서 마우스 포인터가 떠났을 때 이벤트가 발생합니다.

57행 : \$(".ball").mouseup(function(){

ball 클래스에 마우스를 올려놓을 상태에서, 눌렀다 떼었을 때 이벤트가 발생합니다.

## 시작 파일

sample/basic\_jquery/start/basic\_jquery51.html



## JavaScript

```
53 <script>
54 $(function(){
55     $(".ball").toggle(
56         function(){ $(this).css("left", "140px"); },
57         function(){ $(this).css("left", "240px"); },
58         function(){ $(this).css("left", "340px"); },
59         function(){ $(this).css("left", "40px"); }
60     );
61 });
62 </script>
```

55행 : \$(".ball").toggle(

ball 클래스를 누르는 click 이벤트가 발생할 때마다, 순서대로 함수를 실행합니다.

toggle() 이벤트는 jQuery 1.9 이상의 버전에서 없어졌기 때문에, 굳이 이 toggle() 이벤트를 사용해야 한다면 jQuery 1.8 버전을 사용해야 합니다.

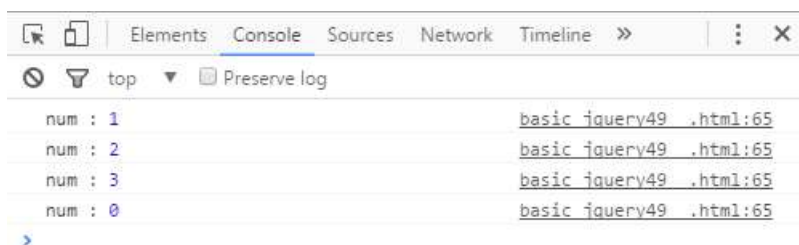
```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8/jquery.min.js"> </script>
```

jQuery 1.9 이상의 버전에서 toggle() 이벤트를 활용하려면, 분기식을 작성하면 될 것입니다.

toggle() 이벤트를 대신해서 작성하기 위해 아래와 같이 코드를 수정했습니다.

#### JavaScript

```
53 <script>
54 $(function(){
55     var num=0;
56
57     $(".ball").click(function(){
58         switch(num){
59             case 0 : $(this).css("left", "140px"); break;
60             case 1 : $(this).css("left", "240px"); break;
61             case 2 : $(this).css("left", "340px"); break;
62             case 3 : $(this).css("left", "40px"); break;
63         }
64         num=(num+1)%4;
65         console.log("num :", num);
66     });
67 }
68 </script>
```



55행 : var num=0;

클릭한 횟수를 기억할 변수를 선언합니다.

57행 : \$(".ball").click(function(){

toggle() 이벤트를 대신해서 click() 이벤트를 적용합니다.

58행 : switch(num){

switch() 구문을 통해서 num 변수 값에 따라서 다른 명령어를 실행합니다.

59행 : case 0 : \$(this).css("left", "140px"); break;

num 변수가 0일 경우에는 ball 클래스의 위치를 140px로 옮겨줍니다.

60행 : case 1 : \$(this).css("left", "240px"); break;

num 변수가 1일 경우에는 ball 클래스의 위치를 240px로 옮겨줍니다.

61행 : case 2 : \$(this).css("left", "340px"); break;

num 변수가 2일 경우에는 ball 클래스의 위치를 340px로 옮겨줍니다.

62행 : case 3 : \$(this).css("left", "40px"); break;

num 변수가 3일 경우에는 ball 클래스의 위치를 40px로 옮겨줍니다.

64행 : num=(num+1)%4;

나눔 연산자는 몫 (4)으로 나누었을 때에 나머집니다. 수의 범위는 0, 1, 2, 3입니다.



### 1-7-3) 문서 로딩 이벤트

아래의 표는 문서가 로딩되거나, 페이지를 빠져나갈 때와 관련된 내용입니다.

| 명령어    | 설명                      |
|--------|-------------------------|
| ready  | 해당 페이지가 로딩되었을 때에 발생합니다. |
| unload | 해당 페이지가 빠져나갈 때에 발생합니다.  |

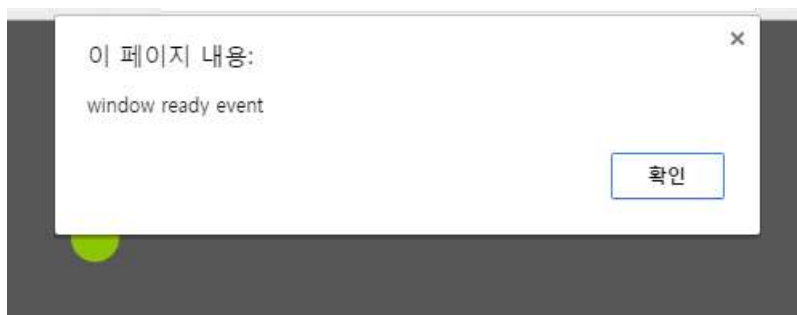
#### 시작 파일

sample/basic\_jquery/start/basic\_jquery50.html

#### JavaScript

```
57 <script>
58 $(window).on("ready", function(e){
59     alert("window ready event!");
60 });
61 $(window).on("unload", function(){
62     alert("document unload event!");
63 });
64 </script>
```

50행 : `$(window).on("ready", function(e){`  
해당 페이지가 로딩되었을 때에 이벤트가 발생합니다.



53행 : `$(window).on("unload", function(){`  
해당 페이지를 빠져나갈 때에 이벤트가 발생합니다.  
IE 브라우저에서만 이 경고창을 확인할 수 있습니다.

문서가 다 로드 되는 시점을 점검하기 위해 아래와 같은 코드를 작성할 수도 있습니다.

```
$("#document").ready(function(){  
});
```

또는 아래와 같이 작성될 수도 있습니다.

```
$(function(){  
});
```

위와 같이 작성된 코드는 Internet Explorer에서는 정상적인 경고 창을 볼 수 있습니다. 하지만 Chrome 브라우저에서는 경고 창을 볼 수 없습니다.

만일 Chrome 브라우저에서 unload 이벤트를 발생하려고 한다면 beforeunload 이벤트를 window에 등록합니다.

```
<script>  
$(window).on("beforeunload", function(){  
    return "window beforeload event!";  
});  
</script>
```

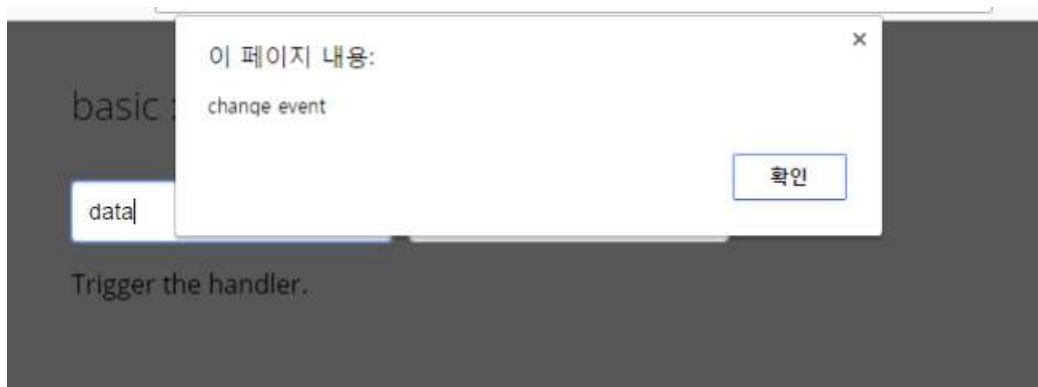
#### 1-7-4) 폼 이벤트

아래의 표는 폼 요소 이벤트와 관련된 내용입니다.

| 명령어    | 설명                      |
|--------|-------------------------|
| blur   | 요소에서 포커스가 떠날 때에 발생합니다.  |
| change | 요소의 값이 변경될 때에 발생합니다.    |
| focus  | 요소가 포커스 되었을 때에 발생합니다.   |
| select | 사용자가 텍스트를 선택했을 때 발생합니다. |
| submit | 폼의 내용을 전송할 때에 발생합니다.    |

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery53.html



#### JavaScript

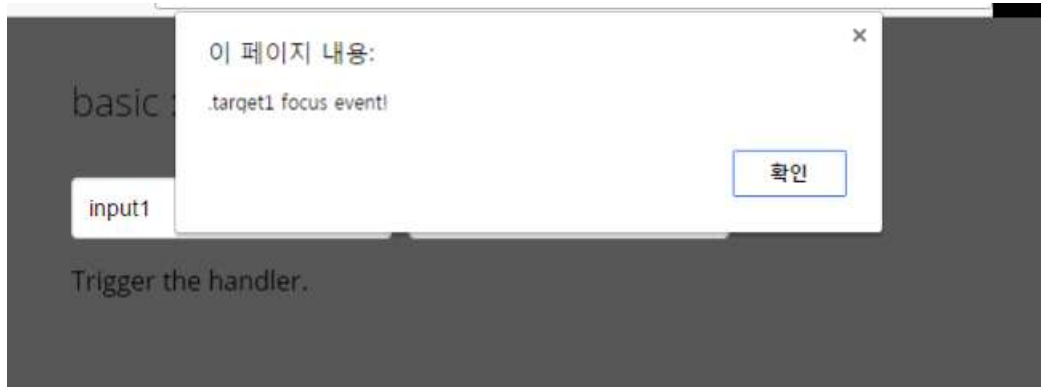
```
43 <script>
44 $(function(){
45     $(".target1").change(function(){
46         alert("change event!");
47     });
48     $(".trigger").click(function(){
49         $(".target1").trigger("change");
50     });
51 });
52 </script>
```

45행 : \$(".target1").change(function(){  
.target1 input 요소의 내용이 바뀔 경우에 이벤트가 발생합니다.

48행 : \$(".trigger").click(function(){  
trigger 클래스를 누르면 강제로 .target1 input의 change 이벤트가 발생합니다.

### 시작 파일

sample/basic\_jquery/start/basic\_jquery54.html



### JavaScript

```
43 <script>
44 $(function(){
45     $(".target1").blur(function(){
46         alert(".target1 blur event!");
47     });
48     $(".target1").focus(function(){
49         alert(".target1 focus event!");
50     });
51     $(".trigger").click(function(){
52         $(".target1").focus();
53     });
54 });
55 </script>
```

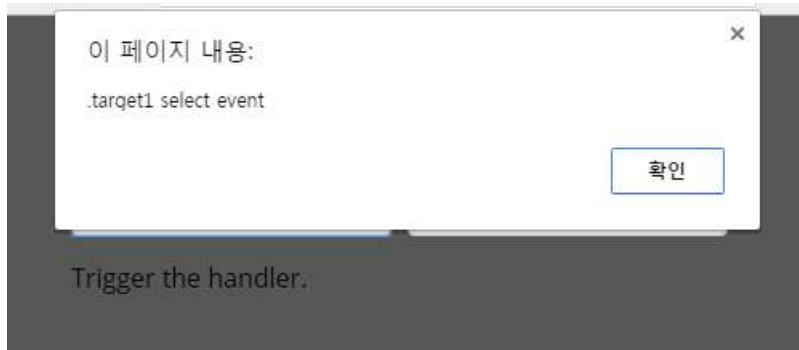
45행 : \$(".target1").blur(function(){  
.target1 input 요소에서 포커스가 떠날 때 이벤트가 발생합니다.

48행 : \$(".target1").focus(function(){  
.target1 input 요소에 대한 포커스가 될 때 이벤트가 발생합니다.

51행 : \$(".trigger").click(function(){  
trigger 클래스를 클릭하면 강제로 .target1 input 요소에 대한 focus() 이벤트가 발생합니다.

### 시작 파일

sample/basic\_jquery/start/basic\_jquery55.html



### JavaScript

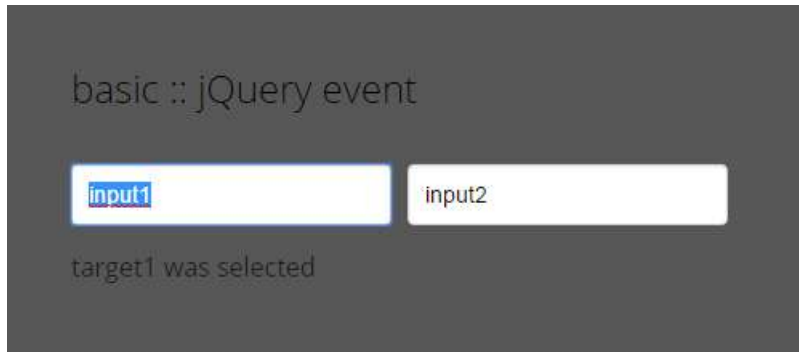
```
43     <script>
44     $(function(){
45         $(".target1").select(function(){
46             alert(".target1 select event");
47         });
48         $(".trigger").click(function(){
49             $(".target1").select();
50         });
51     });
52     </script>
```

45행 : \$(".target1").select(function(){  
.target1 input 요소를 선택할 때 이벤트가 발생합니다.

48행 : \$(".trigger").click(function(){  
trigger 클래스를 클릭하면 강제로 .target1 input 대상의 select() 이벤트가 발생합니다.

## 시작 파일

sample/basic\_jquery/start/basic\_jquery55.html



## JavaScript

```
43
44     <script>
45     $(function(){
46         $("input").select(function(){
47             $(".trigger").text($(this).attr("class")+" was selected").show().fadeOut(1000);
48         });
49     });
50     </script>
```

45행 : `$("input").select(function(){`  
input 요소를 선택할 때 발생하는 이벤트입니다.

46행 : `$(".trigger").text($(this).attr("class")+" was selected").show().fadeOut(1000);`  
trigger 클래스 영역에 현재 선택된 input 요소의 클래스 이름이 작성됩니다.  
그리고 나머지 애니메이션이 발생합니다.

아직 설명이 되지 않은 부분이지만, 아래의 표로만으로도 충분히 이해가 될 것입니다.

| 구문                                                                        | 설명                                     |
|---------------------------------------------------------------------------|----------------------------------------|
| <code>\$(".trigger").text(\$(this).attr("class")+" was selected");</code> | trigger 클래스에 텍스트가 작성됩니다.               |
| <code>\$(".trigger").show();</code>                                       | trigger 클래스가 보이도록 설정됩니다.               |
| <code>\$(".trigger").fadeOut(1000);</code>                                | trigger 클래스를 1초 동안 사라지도록 애니메이션을 구현합니다. |

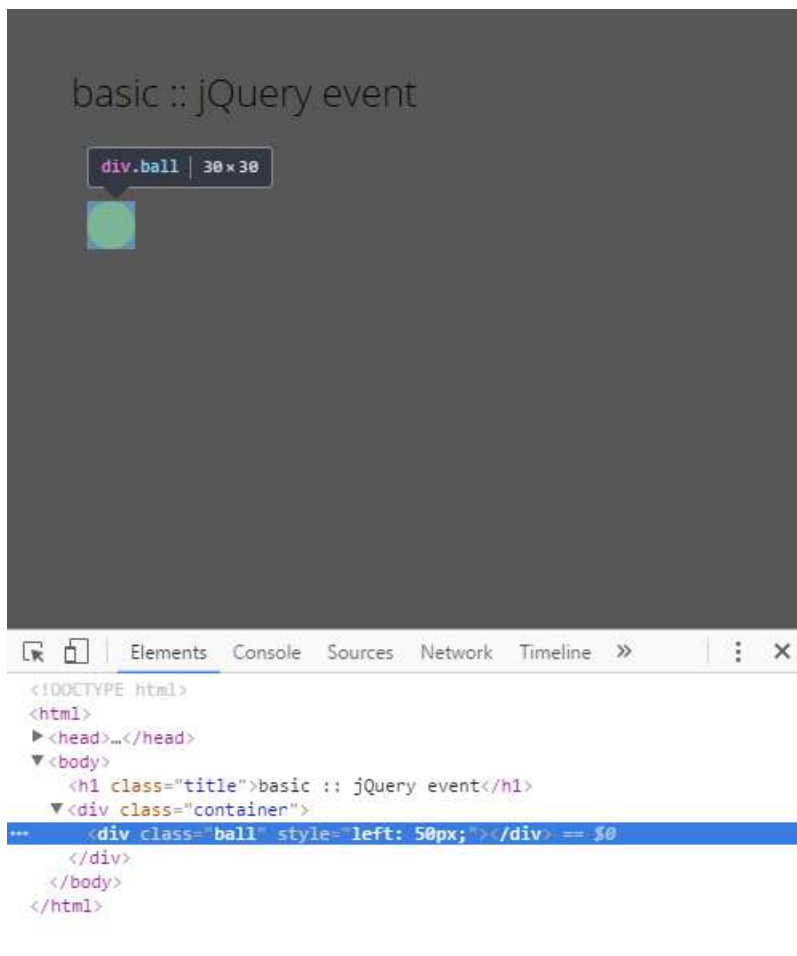
#### 1-7-5) 키보드 이벤트

아래의 표는 사용자의 키보드 이벤트와 관련된 내용입니다.

| 명령어      | 설명                               |
|----------|----------------------------------|
| keydown  | 해당 영역에서 키보드를 눌렀을 때에 발생합니다.       |
| keypress | 해당 영역에서 키보드를 계속 누르고 있을 때에 발생합니다. |
| keyup    | 해당 영역에서 키보드를 눌렀다가 떼었을 때에 발생합니다.  |

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery56.html





## JavaScript

```
49 <script>
50 $(function(){
51     $("body").keydown(function(e){
52         console.log("e.keyCode :", e.keyCode);
53
54         if(e.keyCode == 39){
55             $(".ball").css({ left: "+=10px" });
56         }else if(e.keyCode == 37){
57             $(".ball").css({ left: "-=10px" });
58         }else if(e.keyCode == 38){
59             $(".ball").css({ top: "-=10px" });
60         }else if(e.keyCode == 40){
61             $(".ball").css({ top: "+=10px" });
62         }
63     });
64 });
65 </script>
```

52행 : `$("body").keydown(function(e){`

body에 `keydown()` 이벤트를 등록합니다. `keydown()` 이벤트는 이벤트가 등록된 영역 body 요소에서 키보드를 눌렀을 때에 발생합니다.

53행 : `console.log("e.keyCode :", e.keyCode);`

`keydown()` 이벤트가 발생할 때, 누른 키보드 키는 `keyCode` 속성을 통해 알 수 있습니다.

아래의 표는 예제에서 사용된 키코드 (`keyCode`)와 관련된 내용입니다.

| keyCode | 설명         |
|---------|------------|
| 37      | 왼쪽 방향 키보드  |
| 38      | 위쪽 방향 키보드  |
| 39      | 오른쪽 방향 키보드 |
| 40      | 아래쪽 방향 키보드 |

54행 : `if(e.keyCode == 39){`

누른 키보드가 오른쪽 방향 버튼일 때에 조건입니다.

55행 : \$(".ball").css({ left: "+=10px" });

ball 클래스의 위치를 오른쪽으로 10px 이동합니다.

56행 : }else if(e.keyCode == 37){

누른 키보드가 왼쪽 방향 버튼일 때에 조건입니다.

57행 : \$(".ball").css({ left: "-=10px" });

ball 클래스의 위치를 왼쪽으로 10px 이동합니다.

58행 : }else if(e.keyCode == 38){

누른 키보드가 위쪽 방향 버튼일 때에 조건입니다.

59행 : \$(".ball").css({ top: "-=10px" });

ball 클래스의 위치를 위쪽으로 10px 이동합니다.

60행 : }else if(e.keyCode == 40){

누른 키보드가 아래쪽 방향 버튼일 때에 조건입니다.

61행 : \$(".ball").css({ top: "+=10px" });

ball 클래스의 위치를 아래쪽으로 10px 이동합니다.

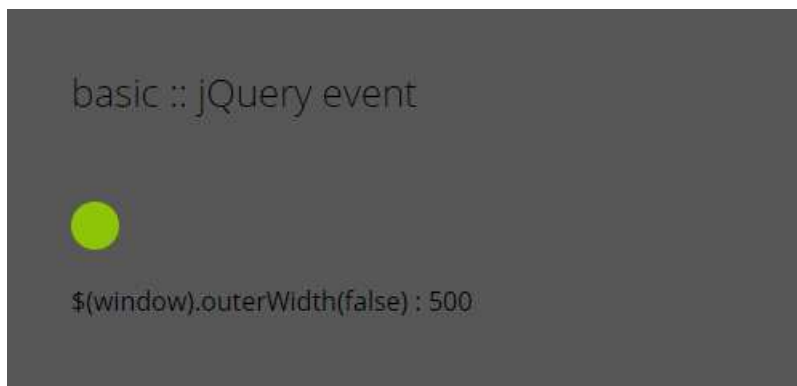
#### 1-7-6) 웹 브라우저 이벤트

아래의 표는 웹 브라우저의 크기를 재조정하거나 스크롤하는 것과 같은 브라우저와 관련된 내용입니다.

| 명령어    | 설명                               |
|--------|----------------------------------|
| resize | 웹 브라우저 윈도우 사이즈의 변화가 있을 때에 발생합니다. |
| scroll | 스크롤이 움직일 때 발생합니다.                |

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery57.html



#### JavaScript

```
49 <script>
50 $(function(){
51     $("body").append('<div class="status">$(window).outerWidth(false) :</div>');
52
53     $(window).resize(function(){
54         $(".status").text("$(window).outerWidth(false) : "+$(window).outerWidth(false));
55     });
56     $(window).trigger("resize");
57 });
58 </script>
```

51행 : \$("body").append('<div class="status">\$(window).outerWidth(false) :</div>');

body 요소에 status 클래스를 추가합니다.

resize() 이벤트가 발생할 때 윈도우 사이즈를 확인할 텍스트 영역입니다.

53행 : `$(window).resize(function(){`

`resize()` 이벤트는 윈도우 사이즈가 변경될 때 발생하는 이벤트입니다.

54행 : `$(".status").text("$(window).outerWidth(false) : "+$(window).outerWidth(false));`

`outerWidth()` 명령어는 매개인자 값이 생략되어 있거나, `false`이면, `padding`과 `border` 값은 크기에 포함되지 않습니다.

56행 : `$(window).trigger("resize");`

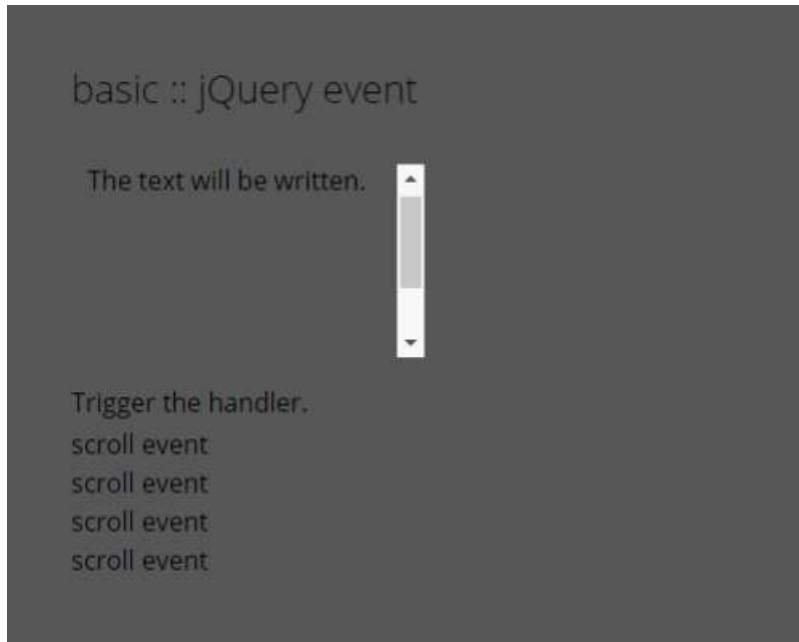
문서가 처음 로딩될 때, `resize()` 이벤트 명령어 내용이 실행됩니다.

`resize()` 이벤트는 초기에 실행되지 않고, 윈도우 크기를 재조정해야만 이벤트가 발생합니다.

따라서 초기에 `resize()` 이벤트를 실행하기 위해 `resize()` 이벤트를 트리거합니다.

## 시작 파일

sample/basic\_jquery/start/basic\_jquery58.html



## JavaScript

```
54 <script>
55 $(function(){
56     $(".text_area").scroll(function(){
57         $(".log").append("<div>scroll event</div>");
58     });
59     $(".trigger").click(function(){
60         $(".text_area").trigger("scroll");
61     });
62 });
63 </script>
```

56행 : \$(".text\_area").scroll(function(){  
text\_area 클래스 영역을 스크롤할 때 이벤트가 발생합니다.

57행 : \$(".log").append("<div>scroll event</div>");  
scroll() 이벤트가 발생되면 log 클래스에 노드가 추가됩니다.

추가되는 노드는 아래와 같습니다.

```
<div>scroll event</div>
```

59행 : \$(".trigger").click(function(){

trigger 클래스를 클릭하면 text\_area 클래스 영역을 스크롤할 때 발생하는 이벤트가 트리거됩니다.

### 1-7-7) 이벤트 버블링

자바스크립트의 이벤트 처리는 생각보다 어렵습니다.

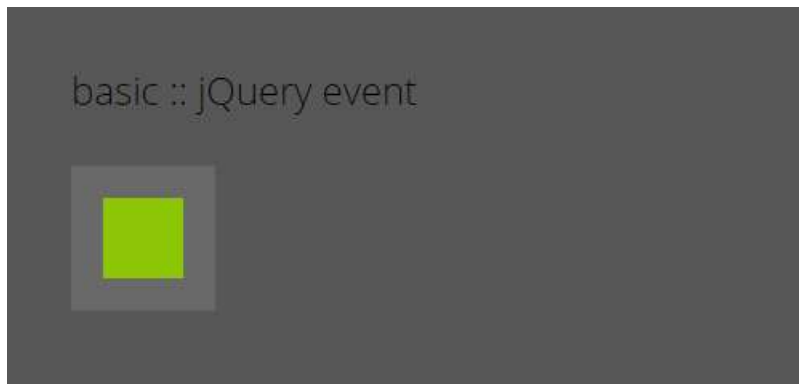
어떤 경우에는 이벤트의 발생이 원하지 않는 방향으로 흐를 때가 있습니다.

특히 마우스 이벤트 처리 시에 이벤트는 까다롭습니다.

간단하게 테스트할 만한 예제를 준비해 보았습니다.

#### 시작 파일

[sample/basic\\_jquery/start/basic\\_jquery59.html](sample/basic_jquery/start/basic_jquery59.html)



#### HTML

```
1      <!DOCTYPE html>
2      <html>
3      <head>
4      <meta charset="utf-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0,
7      minimum-scale=1.0">
8      <title>basic :: jQuery Basic Study</title>
9      <link rel="stylesheet"
10      href="http://fonts.googleapis.com/css?family=Open+Sans:400,300,300italic,400italic,600,6
11      00italic,700,700italic,800,800italic">
12      <style>
13      body {
14      margin: 20px;
15      padding: 20px;
16      line-height: 1;
17      font-family: "Open Sans", sans-serif;
```

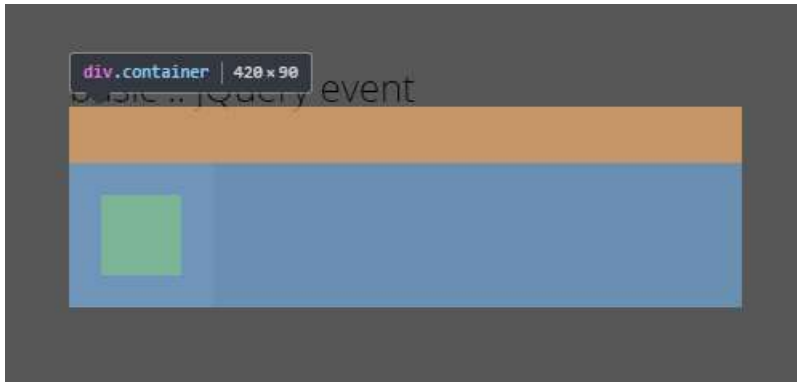
```
15     font-size: 1em;
16     background: #555;
17     color: #000;
18 }
19 .title {
20     margin: 0;
21     padding: 0;
22     font-size: 1.5em;
23     font-weight: 300;
24 }
25 .container {
26     margin-top: 35px;
27 }
28 .wrapper {
29     padding: 20px;
30     width: 50px;
31     height: 50px;
32     background: #666;
33     cursor: pointer;
34 }
35 .wrapper > a {
36     display: block;
37     width: 50px;
38     height: 50px;
39     background: #8ac007;
40     cursor: pointer;
41 }
42 </style>
43 <script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
44 <script>
45 $(function(){
46     $(".wrapper").click(function(e){
47         $(this).css({ background: "rgba(255,255,0,.5)" });
48     });
49 });
50 </script>
51 </head>
52
53 <body>
54 <h1 class="title">basic :: jQuery event</h1>
55 <div class="container">
56     <div class="wrapper">
57         <a href="#"></a>
58     </div>
```



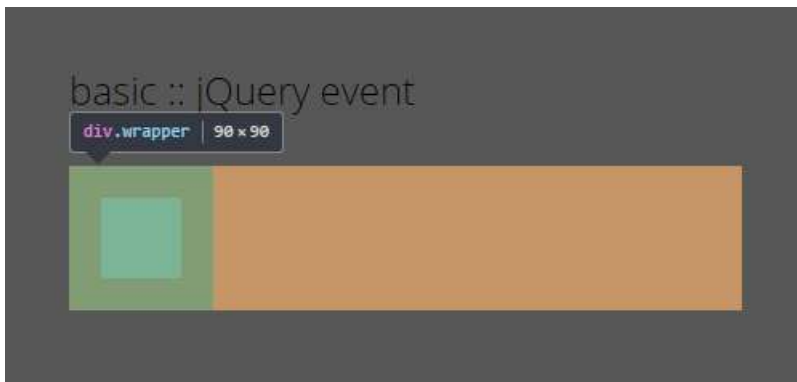
```
59     </div>
60   </body>
61 </html>
```

wrapper 클래스 요소 하위에 a 요소를 배치하였습니다.

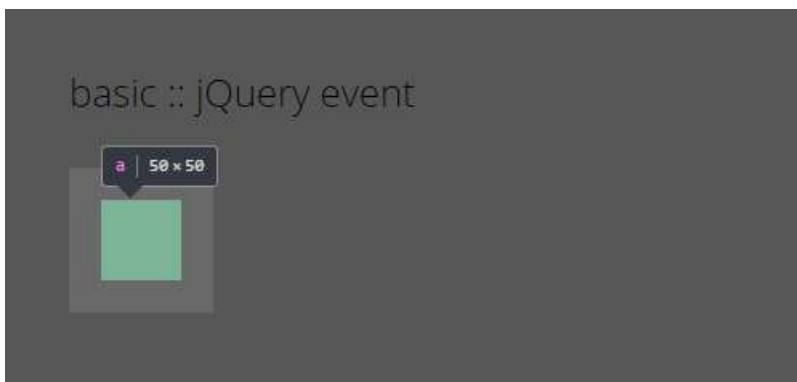
<div class="container"> 영역입니다.



<div class="wrapper"> 영역입니다.



<a href="#"></a> 영역입니다.



이벤트 핸들러 (Event Handler)는 대상에 이벤트가 발생할 때 실행되는 명령어입니다.

더 깊이 있게 설명한다면, 이벤트 핸들러는 캡처 핸들러 (Capture Handler)와 버블 핸들러 (Bubble Handler)로 구분할 수 있습니다.

이벤트 캡처링 (Event Capturing)은 상위 HTML 노드부터 이벤트 발생 영역까지 아래로 내려가며 이벤트를 참조합니다. 이 경우에 캡처 핸들러가 실행됩니다.

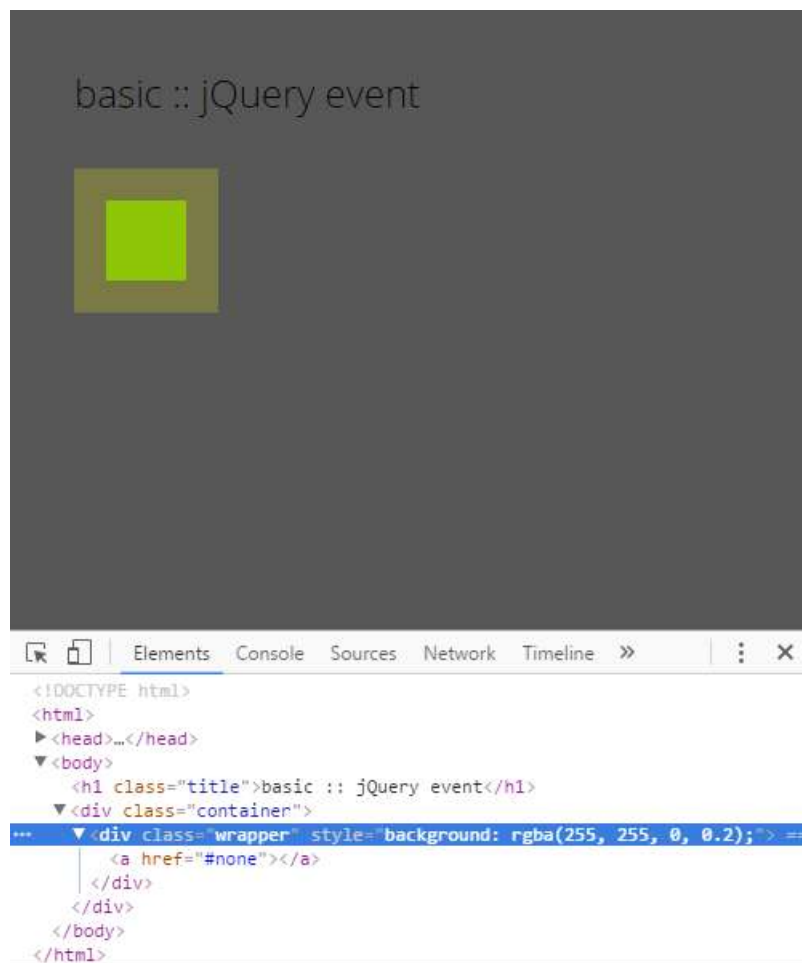
이벤트 발생 요소까지 도달했다면, 다시 이벤트 요소로부터 상위 부모 요소로 올라가면서 이벤트를 참조합니다. 이를 이벤트 버블링 (Event Bubbling)이라고 합니다.

이벤트 버블링이 발생할 때 버블링 핸들러가 실행됩니다.

참고로 Internet Explorer는 이벤트 캡처링을 지원하지 않습니다.

46행 : `$(".wrapper").click(function(e){`

wrapper 클래스를 클릭하면, 이벤트 대상의 배경색을 바꾸어 줍니다.



예제의 코드를 바꾸어 보겠습니다.

#### JavaScript

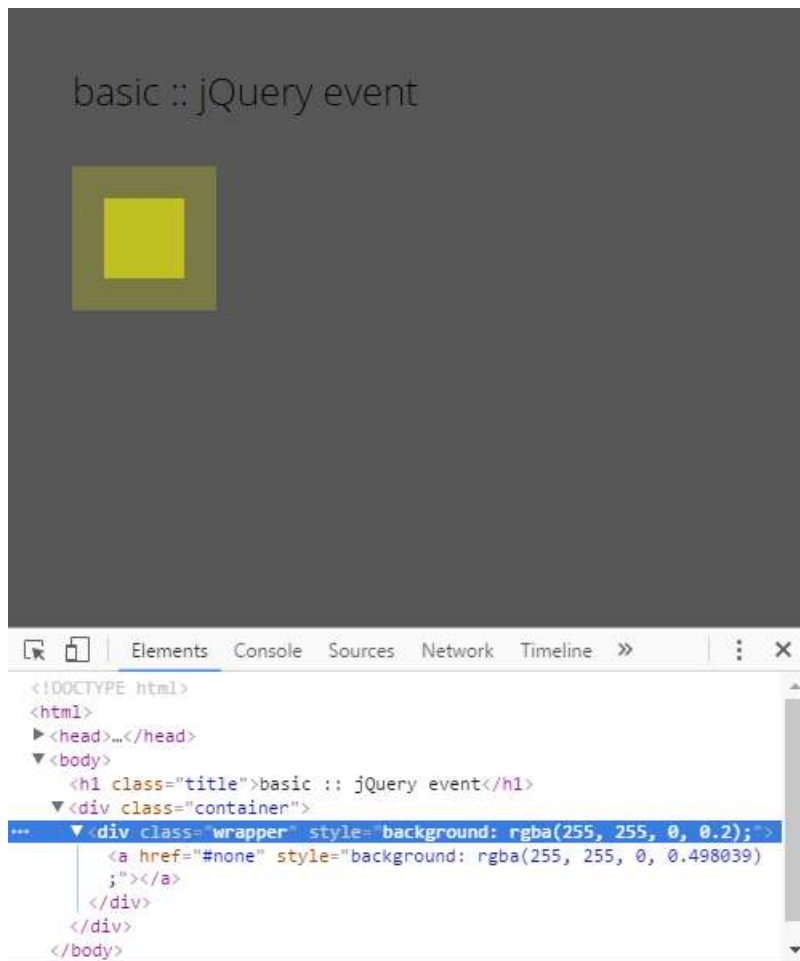
```
44 <script>
45 $(function(){
46     $(".wrapper").click(function(e){
47         $(this).css({ background: "rgba(255,255,0,.2)" });
48     });
49
50     $(".wrapper > a").click(function(e){
51         $(this).css({background:"rgba(255,255,0,.5)"});
52     });
53 });
54 </script>
```

46행 : \$(".wrapper > a").click(function(e){  
wrapper 클래스 하위의 자식 요소인 a 요소를 클릭할 때의 이벤트입니다.

47행 : \$(this).css({ background: "rgba(255,255,0,.5)" });  
이벤트 대상인 a 요소의 색상을 변경합니다.

분명 a 요소만 색상이 변경되는 것이 옳다고 생각되지만, 실제로 a 요소를 클릭하면 상위 wrapper 클래스 영역도 색상이 변경됩니다.

이벤트 버블링이 발생되며 wrapper 클래스의 click 이벤트도 실행되는 것입니다.



일단 a 요소를 클릭하면 a 요소의 이벤트가 발생합니다.(Event Capturing)

다음 단계로 wrapper 클래스로 이벤트 영역이 올라가게 됩니다.(Event Bubbling)

이런 이유로 a 요소를 클릭하면 wrapper 클래스의 click 이벤트도 발생하는 것입니다.

버블링 이벤트를 막아주기 위해서는 stopPropagation() 명령어를 사용합니다. propagation은 번식이란 의미를 지니고 있습니다.

예제의 코드를 바꾸어 보겠습니다.

#### JavaScript

```
44 <script>
45 $(function(){
46     $(".wrapper").click(function(e){
47         $(this).css({ background: "rgba(255,255,0,.2)" });
48     });
49
50     $(".wrapper > a").click(function(e){
51         $(this).css({ background: "rgba(255,255,0,.5)" });
52         e.stopPropagation();
53     });
54 });
55 </script>
```

52행 : `e.stopPropagation();`

a 요소 영역을 클릭하면, 이벤트 대상만이 클릭 이벤트 핸들러가 발생합니다. 버블링 이벤트가 실행되지 않습니다.

basic :: jQuery event



```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <h1 class="title">basic :: jQuery event</h1>
    <div class="container">
      <div class="wrapper">
        <a href="#none" style="background: rgba(255, 255, 0, 0.498039)
        ;"></a> == $0
      </div>
    </div>
  </body>
```

#### 1-7-8) 모션 효과

모션과 애니메이션을 표현할 수 있는 효과입니다.

명령어	설명
hide()	해당하는 요소를 숨깁니다.
show()	해당하는 요소를 보여줍니다.
toggle()	show()와 hide()를 번갈아 실행합니다.

#### hide() :

hide() 명령어는 요소의 스타일을 display: none;으로 변경합니다.

hide() 명령어에 대한 용법입니다. '[]'는 생략이 가능한 인자입니다.

```
.hide([duration] [, easing] [, callback]);
```

- duration : 효과가 지속되는 시간을 의미합니다.  
1/1000초인 밀리 초 단위로서 정해진 옵션으로는 'slow', 'normal', 'fast'가 있습니다.
- easing : 효과의 속도감을 조절하는 부분입니다.
- callback : callback 함수는 스스로 호출되는 함수의 의미로 애니메이션 효과가 완료되면 자동으로 호출되는 함수를 의미합니다.

show() :

요소의 스타일을 display: block; 또는 inline;으로 변경합니다.

```
.show([duration] [, easing] [, callback]);
```

toggle() :

show()와 hide()를 번갈아 가며 실행시켜주는 명령어입니다.

```
.toggle([duration] [, easing] [, callback]);
```

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery60.html

#### JavaScript

```
51 <script>
52 $(function(){
53     $("#show").click(function(){
54         $(".box").show(1000, completeFn);
55     });
56     $("#hide").click(function(){
57         $(".box").hide(1000, completeFn);
58     });
59     function completeFn(){
60         alert($(".box").css("display") : ' + $(".box").css("display"));
61     }
62 });
63 </script>
```

54행 : \$(".box").show(1000, display\_fn);

show 아이디를 클릭하면 box 클래스 요소가 1초 동안 사라집니다.

동작이 완료되면 completeFn() 함수가 실행됩니다. 시간과 callback 함수는 생략되어도 되는 매개인자입니다.





57행 : `$('.box').hide(1000, display_fn);`

hide 아이디를 클릭하면 box 클래스 요소가 1초 동안 나타납니다.  
동작이 완료되면 completeFn() 함수가 실행됩니다.



59행 : `function display_fn(){`

효과가 완료되었을 경우 실행되는 함수를 선언합니다.

60행 `alert($(".box").css("display") : '+$(".box").css("display");`

show 아이디를 누르면 box 클래스 요소의 display 속성은 'block'으로 지정됩니다.

hide 아이디를 누르면 box 클래스 요소의 display 속성은 'none'으로 지정됩니다.

#### 1-7-9) 불투명도 효과

불투명도 (opacity)를 조절하는 명령어입니다.

명령어	설명
<code>fadeIn()</code>	opacity를 0에서 1로 전환하면서 서서히 나타나게 처리합니다.
<code>fadeOut()</code>	opacity를 1에서 0으로 전환하면서 서서히 사라지게 처리합니다.
<code>fadeToggle()</code>	<code>fadeIn()</code> 과 <code>fadeOut()</code> 을 번갈아 가며 실행합니다.
<code>fadeTo()</code>	불투명도를 지정하여 실행합니다.

#### `fadeIn()` :

opacity 0에서 1로 전환하며, 서서히 나타나게 처리합니다. display 상태는 'block'이나 'inline'으로 지정됩니다.

```
.fadeIn([duration] [, easing] [, callback]);
```

#### `fadeOut()` :

opacity 1에서 0로 전환하며 서서히 나타나게 처리됩니다. display 상태는 'none'으로 지정됩니다.

```
.fadeOut([duration] [, easing] [, callback]);
```

#### `fadeToggle()` :

`fadeIn()`과 `fadeOut()`을 번갈아 가며 실행합니다.

```
.fadeToggle([duration] [, easing] [, callback]);
```

## 시작 파일

sample/basic\_jquery/start/basic\_jquery61.html

## JavaScript

```
51 <script>
52 $(function(){
53     $("#fadeOut").click(function(){
54         $(".box").fadeOut(1000, completeFn);
55     });
56     $("#fadeIn").click(function(){
57         $(".box").fadeIn(1000, completeFn);
58     });
59     $("#fadeToggle").click(function(){
60         $(".box").fadeToggle(1000, completeFn);
61     });
62     function completeFn(){
63         alert($(".box").css("display") : '+$(".box").css("display"));
64     }
65 });
66 </script>
```

54행 : \$(".box").fadeOut(1000, display\_fn);

box 클래스가 1초 동안 서서히 사라집니다. 동작이 완료되면 completeFn() 함수가 호출됩니다.



57행 : `$(".box").fadeIn(1000, display_fn);`

box 클래스가 1초 동안 서서히 나타납니다. 동작이 완료되면 `completeFn()` 함수가 호출됩니다.



60행 : `$(".box").fadeToggle(1000, display_fn);`

box 클래스가 번갈아 가면서 사라지거나 나타납니다.

63행 : `alert($(".box").css("display") : '+($(".box").css("display"));`

동작이 완료되면 경고창을 확인할 수 있습니다.

fadeIn id를 누르면 box 클래스 요소의 display 속성은 'block'으로 지정이 되며,  
fadeOut id를 누르면 box 클래스 요소의 display 속성은 'none'으로 지정됩니다.

fadeTo() :

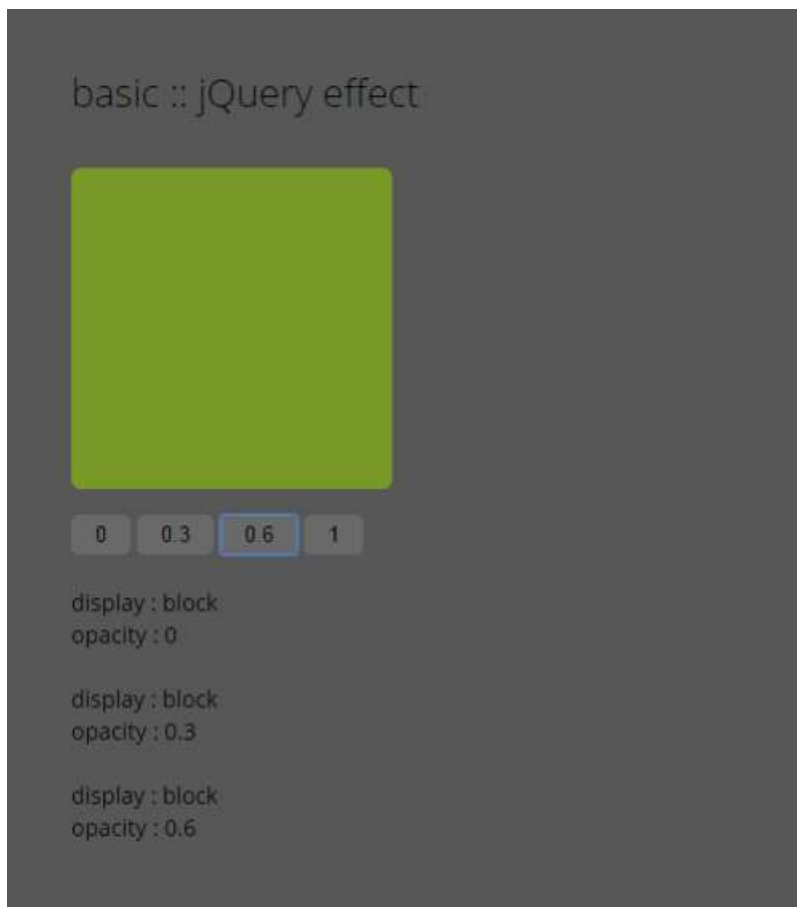
불투명도 (opacity)를 지정하여 값을 조정합니다.

반드시 시간 (duration)을 생략하지 않습니다. fadeOut() 명령어와는 다르게 불투명도가 0으로 되어 있어도 display 상태는 'none'으로 변하지 않습니다.

```
.fadeTo(duration, opacity [, callback]);
```

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery62.html



#### JavaScript

```
56 <script>
57 $(function(){
58     $("body").append('<p class="info"></p>');
59 }
```

```
60     $("button").click(function(){
61         var opacity=$(this).attr("title");
62         $(".box").fadeTo("slow", opacity, function(){
63             $(".info").append("display : "+$(this).css("display")+"<br>");
64             $(".info").append("opacity : "+$(this).css("opacity")+"<br>");
65             $(".info").append("<br>");
66         });
67     });
68 });
69 </script>
```

58행 : `$("body").append('<p class="info"></p>');`

body 요소에 info 클래스를 추가합니다.

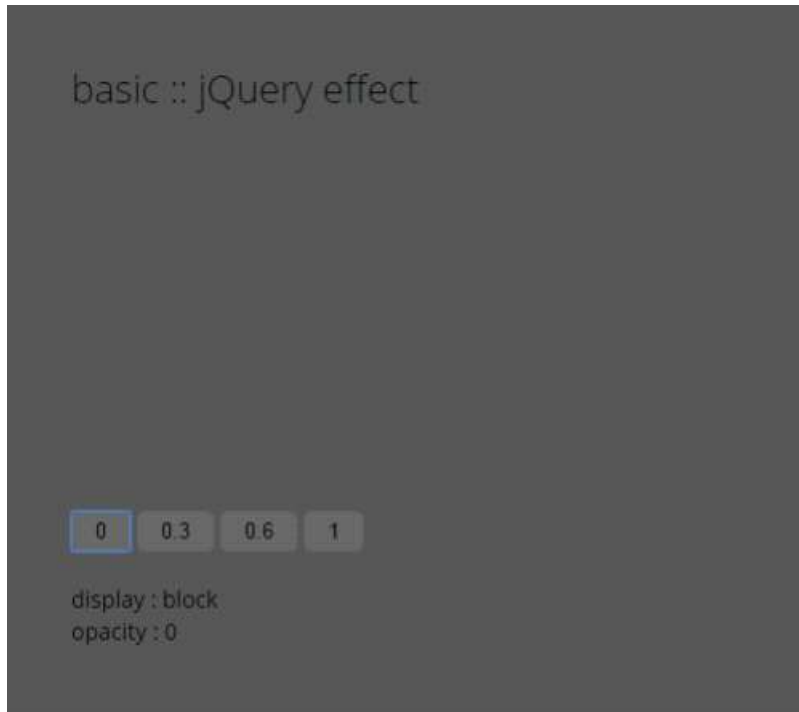
info 클래스는 box 클래스에 대한 display 스타일 속성과 opacity 스타일 속성이 순차적으로 작성될 영역입니다.

61행 : `var opacity=$(this).attr("title");`

클릭된 button 요소의 title 속성을 opacity 변수에 대입합니다.

```
62행 : $(".box").fadeTo("slow", opacity, function(){
```

opacity 변수 값이 0일 경우에는 display 속성은 'block'이고 opacity 속성은 0입니다.



#### 1-7-10) 슬라이딩 효과

sliding 효과는 요소의 높이를 조절하여 접혔다 펴졌다 하는 효과를 만들 수 있습니다.

명령어	설명
slideDown()	슬라이딩 효과로 요소를 보이게 합니다.
slideUp()	슬라이딩 효과로 요소를 숨기게 합니다.
slideToggle()	slideDown()와 slideUp() 명령어를 반복해서 실행합니다.

slideDown(), slideUp(), slideToggle()의 사용 용례입니다.

```
.fadeOut([duration] [, callback]);
```

```
.fadeIn([duration] [, callback]);
```

```
.fadeToggle([duration] [, callback]);
```



## 시작 파일

sample/basic\_jquery/start/basic\_jquery63.html

## JavaScript

```
49 <script>
50 $(function(){
51     $("#slideUp").click(function(){
52         $(".slide").slideUp(1000, completeFn);
53     });
54     $("#slideDown").click(function(){
55         $(".slide").slideDown(1000, completeFn);
56     });
57     $("#slideToggle").click(function(){
58         $(".slide").slideToggle(1000, completeFn);
59     });
60     function completeFn(){
61         alert($(".slide").css("display") : '+' $(".slide").css("display"));
62     }
63 });
64 </script>
```

52행 : \$(".slide").slideUp(1000, completeFn);

slide 클래스가 슬라이딩 효과로 요소를 숨기게 합니다. 동작이 완료되면 completeFn() 함수가 호출됩니다.

basic ::

slideUp

이 페이지 내용:

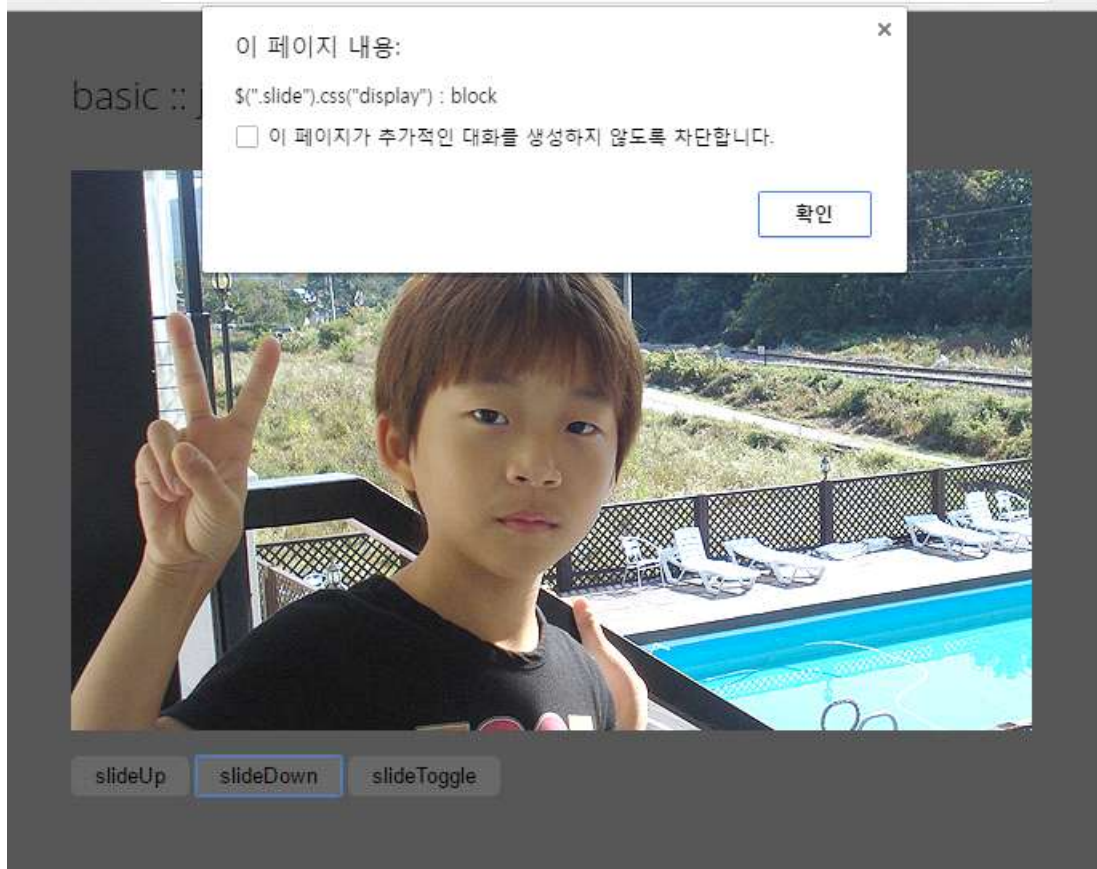
\$(".slide").css("display") : none

×

확인

55행 : `$(".slide").slideToggle(1000, completeFn);`

slide 클래스가 슬라이딩 효과로 요소를 보이게 합니다. 마찬가지로 동작이 완료되면 completeFn() 함수가 호출됩니다.



58행 : `$(".slide").slideToggle(1000, completeFn);`

slide 클래스가 슬라이딩 효과로 요소를 숨기거나 보이게 합니다. 마찬가지로 동작이 완료되면 completeFn() 함수가 호출됩니다.

## 1-7-11) 애니메이션 효과

애니메이션 효과를 직접 만들 수 있습니다.

```
animate(properties [, duration] [, easing] [, complete]);
```

properties 인자는 애니메이션의 속성과 값을 설정합니다.

속성의 내용은 margin, padding, width, height, border 등과 같은 CSS의 속성입니다.  
2개 이상의 단어가 결합된 CSS 속성은 문자열로 만들거나 축약형으로 작성합니다.

```
animate({"margin-left":10px}, 2000);
```

혹은, 아래와 같이 작성되어도 됩니다.

```
animate({marginLeft:10px}, 2000);
```

두 개 이상의 CSS 속성에 애니메이션 효과를 적용할 수 있습니다.

```
animate({width:"200px", height:"200px"}, 2000);
```

속성에 상대 수치를 적용할 수 있습니다.

```
animate({left:"+=20"}, 2000);
```

애니메이션 효과가 완료되면, 완료 함수를 호출할 수도 있습니다.

```
animate({left:"+=20"}, 2000, function(){});
```

## 시작 파일

sample/basic\_jquery/start/basic\_jquery64.html

## JavaScript

```
49 <script>
50 $(function(){
51     $("#slideLeft").click(function(){
52         $(".slide").animate({width:"0px"}, 1000, completeFn);
53     });
54     $("#slideRight").click(function(){
55         $(".slide").animate({width:"600px"}, 1000, completeFn);
56     });
57     function completeFn(){
58         alert($(".slide").css("display") : ' + $(".slide").css("display"));
59     }
60 });
61 </script>
```

52행 : `$(".slide").animate({width:"0"}, 1000, completeFn);`

slideLeft 아이디를 클릭하면 slide 클래스의 가로 크기를 0으로 애니메이션을 적용합니다.

반면 slideUp(), slideDown() 명령어는 수직 방향의 슬라이딩 효과만을 적용합니다.

basic :: jQuery effect

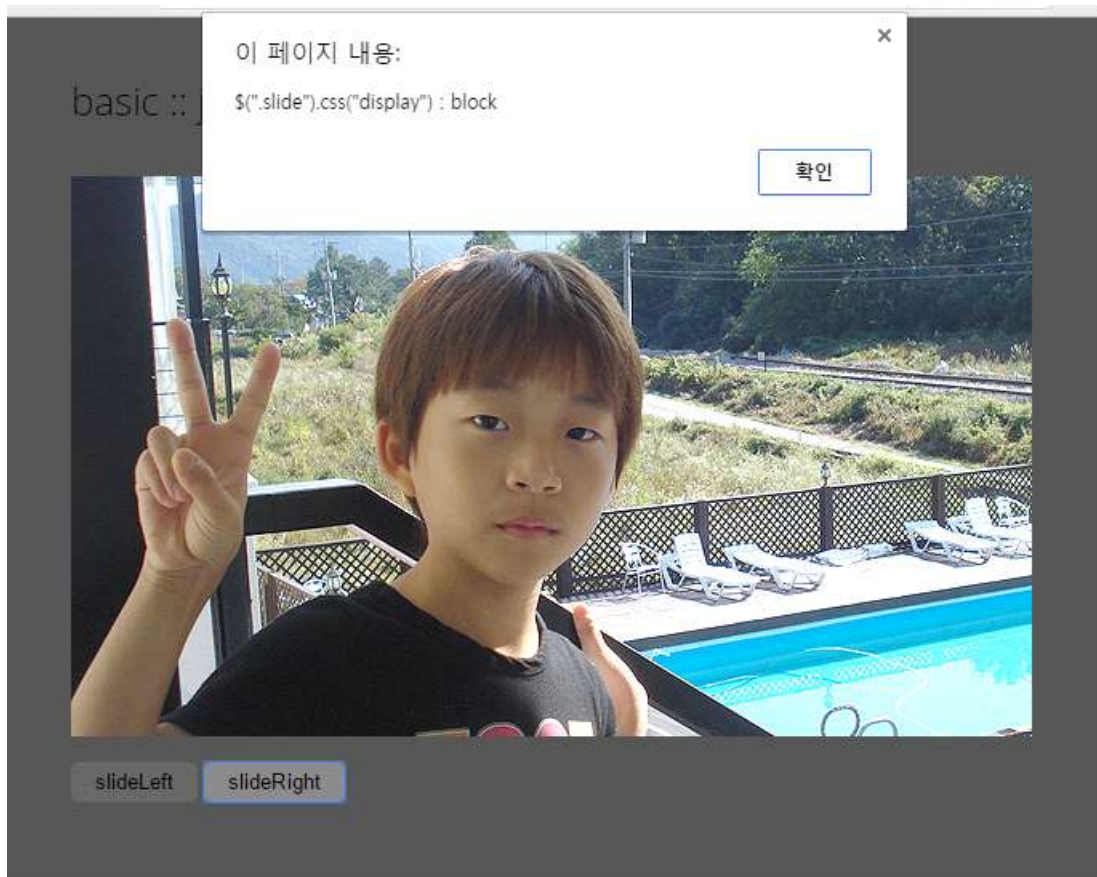


slideLeft

slideRight

55행 : `$(".slide").animate({width:"600px"}, 1000, completeFn);`

slideRight 아이디를 클릭하면 slide 클래스의 가로 크기를 600px로 애니메이션을 적용합니다.

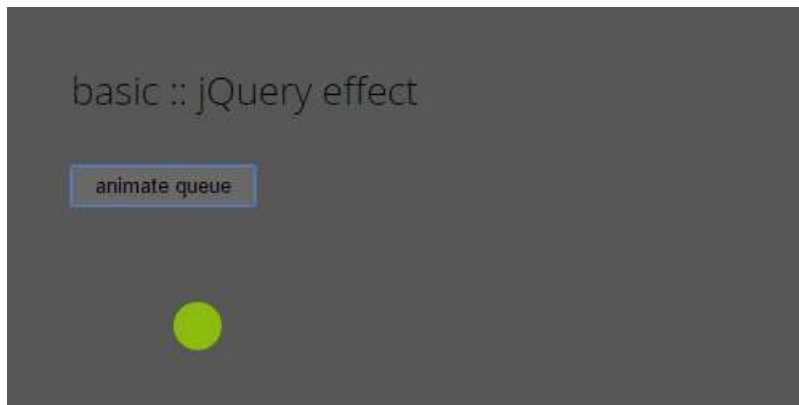


## 1-7-12) 애니메이션 큐(Queue)

여러 개의 애니메이션이 실행을 기다리는 대기열을 의미합니다.

### 시작 파일

sample/basic\_jquery/start/basic\_jquery65.html



### JavaScript

```
53 <script>
54 $(function(){
55     $("#queue").click(function(){
56         $(".ball").animate({"opacity":"0.5", "left":"+=400px"}, 2000)
57         .animate({"top":"-=100px"}, 1000);
58     });
59 });
60 </script>
```

56행 : \$(".ball").animate({"opacity":"0.5", "left":"+=400px"}, 2000)

1단계 애니메이션으로 불투명도를 0.5로 지정하고, 오른쪽으로 400px 위치로 이동됩니다.

57행 : .animate({"top":"-=100px"}, 1000);

2단계 애니메이션으로 위쪽으로 100px 위치로 이동됩니다.



### 1-7-13) 애니메이션 정지

애니메이션을 정지시키는 명령어입니다.

```
.stop([clearQueue] [, jumpToEnd]);
```

- clearQueue : 기본 값은 false이며, 큐에 대기 중인 효과들을 삭제 (true)할 것인지를 여부를 결정합니다. 큐에 대기하고 있는 효과를 삭제하려면 true, 삭제하지 않고 실행하려면 false를 지정합니다.
- jumpToEnd : 기본 값은 false이며, 진행 중인 애니메이션을 완료할 것인지를 결정합니다. true이면 현재 진행 중인 애니메이션의 종료 시점을 이동하고, 그렇지 않고 stop이 수행된 지점에서 멈추고 끝내려면 false로 지정합니다.

stop() 명령어에 인자가 하나도 지정되지 않게 된다면, stop(false, false)와 같습니다.

stop() 명령어를 실행하면, 누른 시점을 기준으로 큐를 삭제하지 않으며 바로 애니메이션이 멈추게 되는 것입니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery66.html

#### JavaScript

```
72 <script>
73 $(function(){
74     $(".start_btn").click(function(){
75         $(this).siblings("div").animate({opacity:0.1, left:"500px"}, 1000)
76         .animate({opacity:0.4, top:0, left:"100px"}, 1000)
77         .slideUp(1000)
78         .slideDown(1000)
79         .animate({left:0}, 1000)
80         .animate({opacity:1}, 2000);
81     });
82
83     $(".stop_btn1").click(function(){
84         $(".box1").stop(true, true);
85     });
86     $(".stop_btn2").click(function(){
87         $(".box2").stop(true, false);
88     });
89     $(".stop_btn3").click(function(){
```

```

90         $(".box3").stop(false, true);
91     });
92     $(".stop_btn4").click(function(){
93         $(".box4").stop(false, false);
94     });
95 });
96 </script>

```

74행 : \$(".startbtn").click(function(){

4개의 startbtn 클래스 버튼을 누를 경우에 실행되는 내용입니다.



75행 : \$(this).siblings("div").animate({opacity:"0.1", left:"500px"}, 1000)

startbtn 클래스 버튼과 같은 경로에 있는 div 요소(.box1 ~ .box4)를 애니메이션 효과를 적용합니다.

단계별로 여러 동작들에 대한 애니메이션 효과를 적용합니다.

84행 : \$(".box1").stop(true, true);

현재 진행되고 있는 효과는 정지되면서 효과의 맨 마지막으로 즉시 이동한 후 큐에 남아 있는 효과는 모두 지우고 끝냅니다.

87행 : \$(".box2").stop(true, false);

현재 진행되고 있는 효과는 즉시 정지되고 큐에 남아 있는 효과는 모두 지우고 끝냅니다.

90행 : \$(".box3").stop(false, true);

현재 진행되고 있는 효과는 정지되면서 효과의 맨 마지막으로 즉시 이동한 후 큐에 남아 있는 효과를 수행합니다.

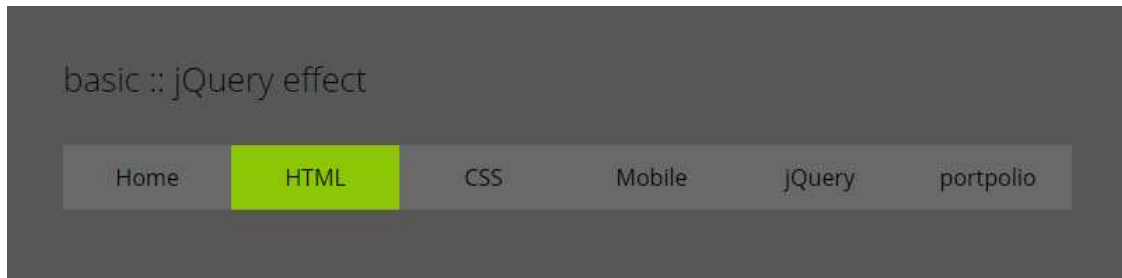
93행 : \$(".box4").stop(false, false);

현재 진행되고 있는 효과는 정지되고 큐에 남아 있는 효과를 수행합니다. 매개인자가 모두 없는 경우와 같습니다.

stop() 명령어를 활용이 필요한 간단한 1Depth 메뉴 예제를 만들어 보시다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery67.html



#### HTML

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <meta http-equiv="X-UA-Compatible" content="IE=edge">
6 <meta name="viewport" content="width=device-width, initial-scale=1.0,
  minimum-scale=1.0">
7 <title>basic :: jQuery Basic Study</title>
8 <link rel="stylesheet"
  href="http://fonts.googleapis.com/css?family=Open+Sans:400,300,300italic,400italic,600,6
  00italic,700,700italic,800,800italic">
9 <style>
10 body {
11     margin: 20px;
12     padding: 20px;
13     line-height: 1;
14     font-family: "Open Sans", sans-serif;
15     font-size: 1em;
16     background: #555;
17     color: #000;
18 }
19 ul, li {
20     margin: 0;
21     padding: 0;
22     list-style: none;
23 }
24 a {
```

```
25     display: inline-block;
26     margin-bottom: 5px;
27     text-decoration: none;
28     color: #666;
29 }
30 .title {
31     margin: 0;
32     padding: 0;
33     font-size: 1.5em;
34     font-weight: 300;
35 }
36 .container {
37     margin-top: 35px;
38 }
39 .navi {
40     width: 720px;
41 }
42 .navi li {
43     float: left;
44     width: 120px;
45 }
46 .navi a {
47     display: block;
48     padding: 15px 0;
49     width: 120px;
50     text-align: center;
51     text-decoration: none;
52     background: #666;
53     color: #000;
54     -webkit-transition: background 0.3s;
55     transition: background 0.3s;
56 }
57 .navi a.active {
58     background: #8ac007;
59 }
60 @media screen and (max-width: 720px) {
61     .navi {
62         width: 100%;
63     }
64     .navi li {
65         float: none;
66         width: 100%;
67     }
```

```

68     .navi a {
69         width: 100%;
70     }
71 }
72 </style>
73 <script src="http://code.jquery.com/jquery-1.10.1.min.js"> </script>
74 <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.min.js"> </script>
75 <script>
76 $(function(){
77     $(".navi li a").hover(
78         function(){
79             $(this).stop().animate({"background-color":"#8ac007"}, 300);
80         },
81         function(){
82             $(this).stop().animate({"background-color":"#666"}, 300);
83         }
84     );
85 });
86 </script>
87 </head>
88
89 <body>
90 <h1 class="title">basic :: jQuery effect</h1>
91 <div class="container">
92     <ul class="navi">
93         <li><a href="#">Home</a></li>
94         <li><a href="#">HTML</a></li>
95         <li><a href="#">CSS</a></li>
96         <li><a href="#">Mobile</a></li>
97         <li><a href="#">jQuery</a></li>
98         <li><a href="#">portpolio</a></li>
99     </ul>
100 </div>
101 </body>
102 </html>

```

43행 : .navi li {float: left;}

li 요소를 가로 배치로 스타일합니다.

47행 : .navi a {display: block;}

a 요소의 display 속성을 block으로 지정합니다.

55행 : `.navi a {transition: background 0.3s;}`

a 요소에 CSS3의 전환 효과를 적용합니다.

78행 : `$(".navi li a").hover(`

a 요소에 `hover()` 이벤트를 등록합니다. `hover()` 이벤트는 `mouseenter()` 이벤트와 `mouseleave()` 이벤트를 한 번에 적용합니다.

80행 : `$(this).stop().animate({"background-color": "#8ac007"}, 300);`

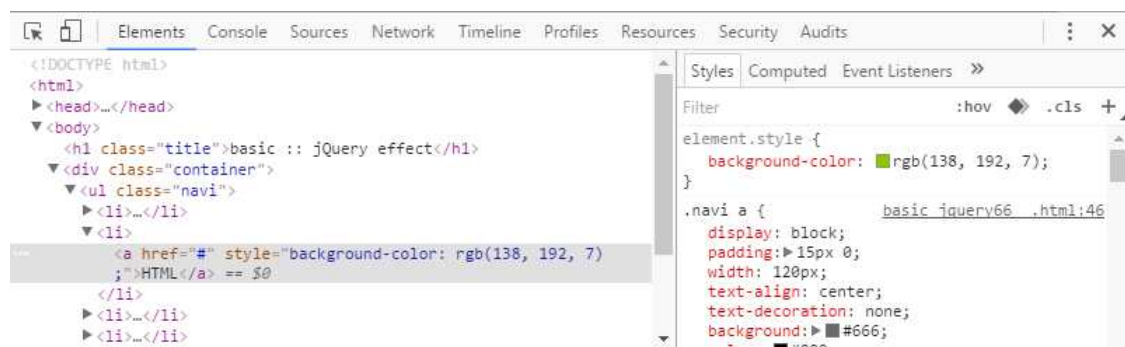
`stop()` 명령어로 이전 애니메이션 효과를 제거합니다.

`stop()` 명령어가 없는 코드로 확인해 보세요.

```
$(function(){
    $(".navi li a").hover(
        function(){
            $(this).animate({"background-color": "#8ac007"}, 300);
        },
        function(){
            $(this).animate({"background-color": "#666"}, 300);
        }
    );
});
```

메뉴 사이를 왔다 갔다 하면 오류를 확인할 수 있습니다. `hover()`와 같이 자주 발생할 수 있는 마우스 이벤트는 `stop()` 명령어로 이전 이벤트를 제거하는 것이 중요합니다.

코드가 적용이 된다면, a 요소에 Inline 방식으로 스타일이 작성되는 것을 확인할 수 있습니다.



83행 : `$(this).stop().animate({"background-color": "#666"}, 300);`

a 요소에서 마우스가 벗어나면 원래 색상대로 바뀌게 됩니다. 마찬가지로 코드가 적용이 된다면, a 요소에 Inline 스타일로 스타일이 지정됩니다.

이처럼 a 요소에 Inline 스타일이 지정되는 것은 다른 CSS 스타일보다 강조되어 스타일의 혼란을 야기할 수 있습니다.

다음에 소개되는 예제 방식은 css() 명령어의 단점을 개선하는 방식입니다.  
기존에 미리 준비되었던 클래스를 추가(addClass())하거나 제거(removeClass())합니다.

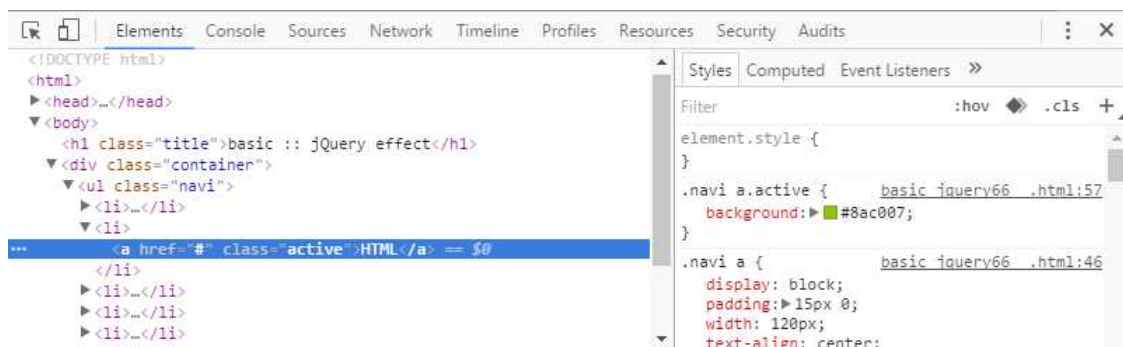
예제의 코드를 바꾸어 보겠습니다.

#### JavaScript

```
76 <script>
77 $(".navi li a").hover(
78     function(){
79         $(this).addClass("active");
80     },
81     function(){
82         $(this).removeClass("active");
83     }
84 );
85 </script>
```

79행 : \$(this).addClass("active");

a 요소에 마우스를 올리면 active 클래스가 추가됩니다.



82행 : \$(this).removeClass("active");

a 요소에 마우스를 내리면 active 클래스가 제거됩니다.

이처럼 클래스를 추가하거나 제거하면, 기존 CSS 스타일과 상충되어서 생기는 혼란을 막을 수 있을 것입니다.

또한 단순한 CSS3의 transition 속성을 사용하는 것이므로 애니메이션에 대한 동작도 가볍게 움직이게



됩니다. JavaScript로 만든 애니메이션 효과보다는 훨씬 가볍게 느껴지죠.

#### 1-7-14) 애니메이션 지연

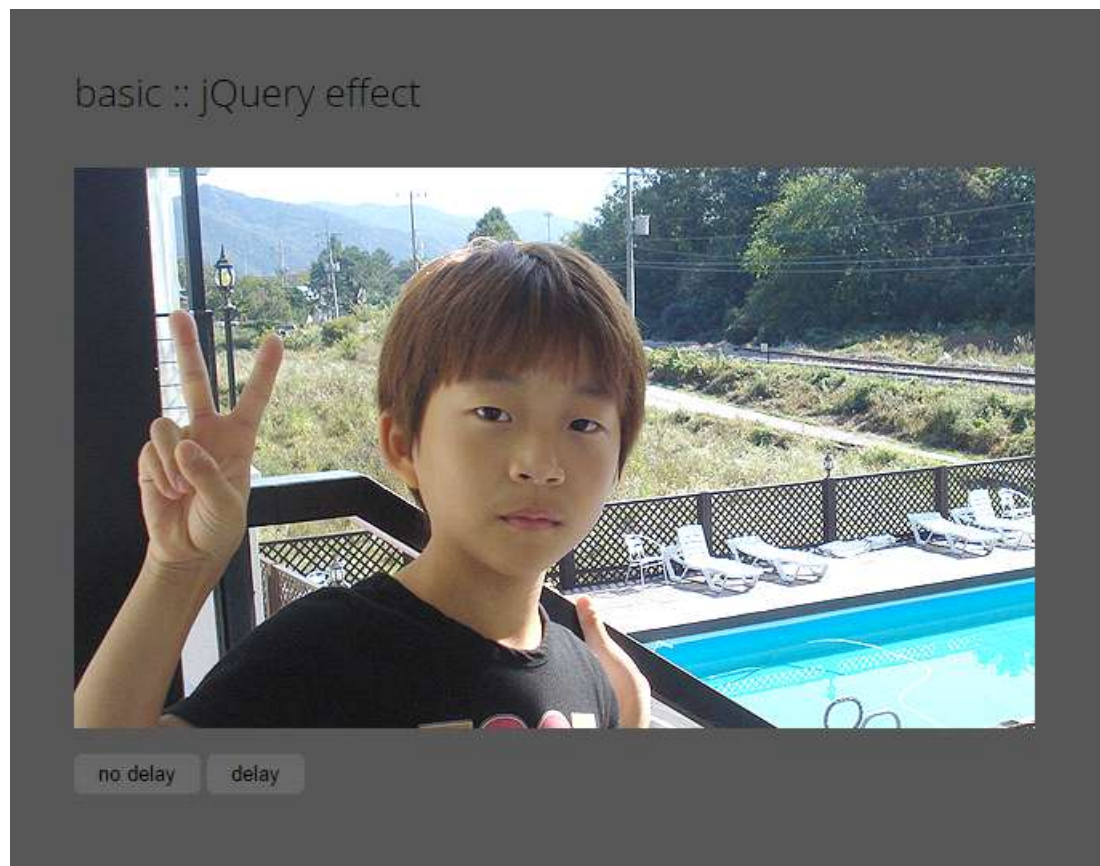
애니메이션 효과를 delay() 명령어로 지연시킬 수 있습니다.

```
.delay(duration [, queueName]);
```

단 매개변수가 없는 show()나 hide()와 같은 경우는 delay() 명령어가 적용되지 않습니다.

#### 시작 파일

sample/basic\_jquery/start/basic\_jquery68.html



#### JavaScript

```
49 <script>
50 $(function(){
51     $("#noDelay").click(function(){
52         $(".slide").slideUp(1000, slideResetFn);
```

```
53     });  
54     $("#delay").click(function(){  
55         $(".slide").delay(2000).slideUp(1000, slideResetFn);  
56     });  
57     function slideResetFn(){  
58         $(".slide").slideDown(1000);  
59     }  
60     });  
61 </script>
```

52행 : \$(".slide").slideUp(1000, slideResetFn);

slide 클래스는 1초 동안 슬라이드 효과로 가려집니다.

55행 : \$(".slide").delay(2000).slideUp(1000, slideResetFn);

slide 클래스는 2초 정도 지연되었다가, 1초 동안 슬라이드 효과로 가려집니다.

58행 : \$(".slide").slideDown(1000);

slideUp() 효과가 완료되면, 1초 정도의 시간으로 slideDown() 효과가 적용됩니다.

## 1-7-15) jQuery UI

jQuery UI는 jQuery 공식 홈페이지에서 무료로 배포한 라이브러리입니다.

다양한 인터페이스와 위젯, 효과를 사용할 수 있게 배포한 jQuery UI를 이용하면 웹에서 좀 더 다양한 효과를 낼 수 있습니다.

### jQuery UI 설정 :

jQuery UI 파일을 jQuery 파일 다음에 연결해 줍니다.

```
<script type="text/javascript" src="js/jquery-1.10.2.min.js">
<script type="text/javascript" src="js/jquery-ui-1.10.3.custom.min.js">
```

구글 CDN 방식으로 파일을 연결한 경우입니다.

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"> </script>
<script src="http://ajax.googleapis.com/ajax/libs/jqueryui/1.10.3/jquery-ui.min.js"> </script>
```

### jQuery UI easing :

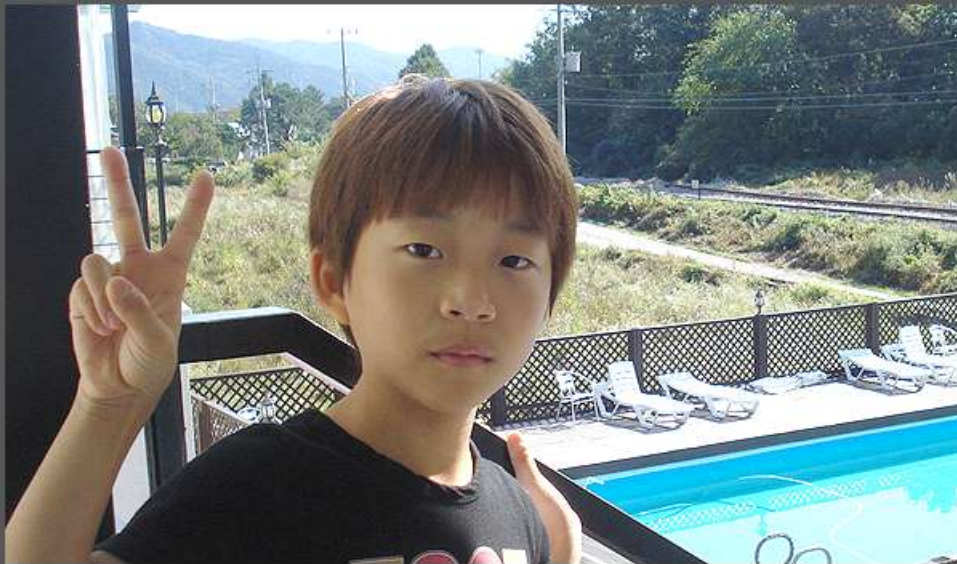
jQuery UI는 jQuery의 다양한 easing 효과를 표현합니다.

easing 효과를 사용하면 애니메이션에 속도감을 부여할 수 있습니다.

## 시작 파일

sample/basic\_jquery/start/basic\_jquery69.html

### basic :: jQuery effect



easeOutQuart

easeInOutElastic

easeOutBack

easeOutBounce

## JavaScript

```
50 <script>
51 $(function(){
52     $("button").click(function() {
53         var easing=$(this).attr("data-ease");
54
55         $(".slide").slideUp(1000, easing, function(){
56             $(this).slideToggle();
57         });
58     });
59 });
60 </script>
```

53행 : `var easing=$(this).attr("data-ease");`

button 요소의 'data-ease' 속성 값을 easing 변수에 대입합니다.

아래와 같이 작성되어도 같은 값을 대입할 수 있습니다.

```
var easing=$(this).text();
```

55행 : `$(".slide").slideUp(1000, easing, function(){`

slide 클래스를 1초 동안 슬라이드 효과로 가려집니다. 이 때 easing 변수 형식대로 easing 효과를 표현할 수 있습니다. 동작이 완료되면 callback 함수가 실행됩니다.

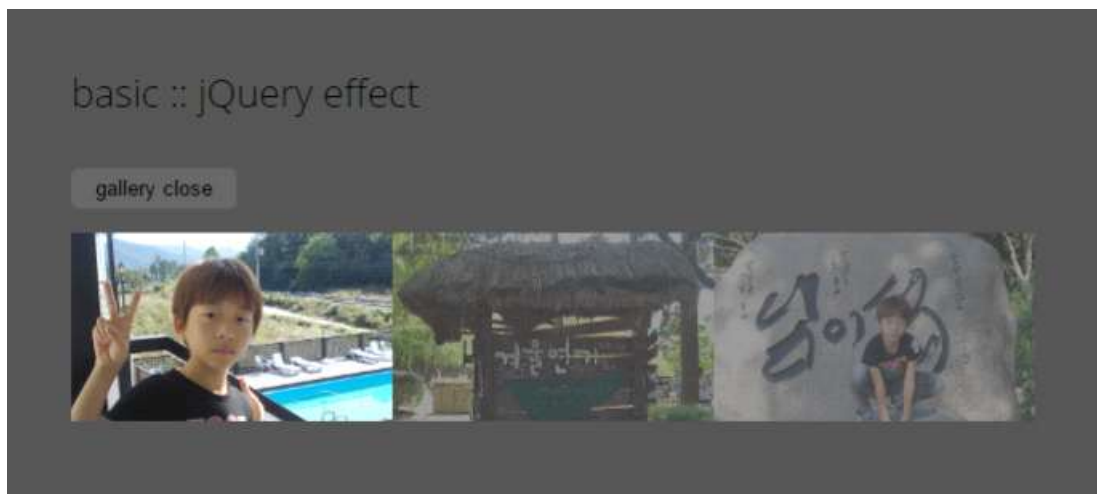
56행 : `$(this).slideToggle();`

슬라이드 효과가 완료되면 다시 슬라이드 효과로 보입니다.

이제 easing 효과를 활용한 간단한 예제를 만들어 봅시다.

#### 시작 파일

basic\_jquery69.html



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <meta http-equiv="X-UA-Compatible" content="IE=edge">
6 <meta name="viewport" content="width=device-width, initial-scale=1.0,
  minimum-scale=1.0">
```

```
7 <title>basic :: jQuery effect</title>
8 <link rel="stylesheet"
  href="http://fonts.googleapis.com/css?family=Open+Sans:400,300,300italic,400italic,600,600italic,700,700italic,800,800italic">
9 <style>
10 body {
11     margin: 20px;
12     padding: 20px;
13     line-height: 1;
14     font-family: "Open Sans", sans-serif;
15     font-size: 1em;
16     background-color: #555;
17     color: #000;
18 }
19 ul, li {
20     margin: 0;
21     padding: 0;
22     list-style: none;
23 }
24 button {
25     padding: 5px 15px;
26     background-color: #666;
27     color: #000;
28     border: none;
29     cursor: pointer;
30     border-radius: 5px;
31     -webkit-transition: all 0.5s;
32     transition: all 0.5s;
33 }
34 button:hover {
35     background-color: #888;
36 }
37 .title {
38     margin: 0;
39     padding: 0;
40     font-size: 1.5em;
41     font-weight: 300;
42 }
43 .container {
44     position: relative;
45     left: 0;
46     top: 0;
47     margin-top: 35px;
48     width: 600px;
```

```
49     height: 200px;
50     overflow: hidden;
51 }
52 .gallery {
53     position: absolute;
54     z-index: 1;
55     left: -600px;
56     top: 40px;
57     width: 600px;
58     height: 200px;
59     list-style: none;
60     -webkit-transition: left 0.5s ease-out;
61     transition: left 0.5s ease-out;
62 }
63 .gallery li {
64     float: left;
65     width: 200px;
66     height: 200px;
67     opacity: 0.4;
68     -webkit-transition: opacity 0.3s;
69     transition: opacity 0.3s;
70 }
71 .gallery li img {
72     width: 200px;
73 }
74 .gallery.active {
75     left: 0;
76 }
77 .gallery li.over {
78     opacity: 1;
79 }
80 </style>
81 <script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
82 <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.min.js"></script>
83 <script>
84 $(function(){
85     var status="closed";
86
87     $("#tab").click(
88         function(){
89             if(status == "closed"){
90                 status="open";
91                 $(this).text("gallery close");
92                 $(".gallery").stop().animate({left:"0px"}, 500, "easeOutBack");
```



```

93         }else if(status == "open"){
94             status="closed";
95             $(this).text("gallery open");
96             $(".gallery").stop().animate({left:"-600px"}, 500);
97         }
98     }
99 );
100 $(".gallery li").hover(
101     function(){
102         $(this).animate({opacity:1}, 300);
103     },
104     function(){
105         $(this).animate({opacity:0.4}, 300);
106     }
107 );
108 });
109 </script>
110 </head>
111
112 <body>
113 <h1 class="title">basic :: jQuery effect</h1>
114 <div class="container">
115     <button id="tab">gallery open</button>
116     <ul class="gallery">
117         <li></li>
118         <li></li>
119         <li></li>
120     </ul>
121 </div>
122 </body>
123 </html>

```

85행 : `var status="closed";`

현재 탭이 열린 상태인지 닫힌 상태 인지를 저장하는 변수를 선언합니다.  
status 변수 값이 'closed'이면 닫힌 상태이고, 'open'이면 열린 상태입니다.

87행 : `$("#tab").click(`

tab 아이디를 클릭할 경우의 이벤트입니다. 토글 기능으로 적용될 toggle() 이벤트를 적용해도 되지만, toggle() 이벤트는 jQuery 1.8 이하 버전에서만 사용이 가능합니다.  
따라서 status라는 변수를 사용해 토글 기능을 구현했습니다.

89행 : `if(status == "closed"){`

status 변수 값이 'closed'인 경우, 즉 닫힌 상태입니다.

91행 : `$(this).text("gallery close");`

버튼의 라벨을 'gallery close'로 바꾸어 줍니다. 현재는 열린 상태이기 때문에 반대의 동작에 관련된 내용을 작성합니다.

92행 : `$(".gallery").stop().animate({left:"0px"}, 500, "easeOutBack");`

gallery 클래스의 left 좌표가 이동됩니다. 'easeOutBack' 방식으로 easing 효과가 적용됩니다.

93행 : `}else if(status == "open"){`

status 변수 값이 'open'인 경우, 즉 열린 상태입니다.

95행 : `$(this).text("gallery open");`

버튼의 라벨을 'gallery open'으로 바꾸어 줍니다. 현재는 닫힌 상태이기 때문에 반대의 동작에 관련된 내용을 작성합니다.

96행 : `$(".gallery").stop().animate({left:"-600px"}, 500);`

gallery 클래스의 left 좌표가 이동됩니다. 'easeOutBack' 방식으로 easing 효과가 적용됩니다.

예제의 코드를 바꾸어 보겠습니다.

easing 효과를 탭 영역을 CSS3의 transition 속성을 사용해서 만들어 봅시다.

이번 예제에서도 바뀐 부분은 animate() 명령어 대신 addClass() 명령어 방식으로 대상 요소에 클래스를 추가하는 방식입니다. CSS3의 transition 효과를 사용하게 되면 훨씬 부드러운 움직임 효과를 확인할 수 있을 것입니다.

#### JavaScript

```
83 <script>
84 $(function(){
85     var status="closed";
86
87     $("#tab").click(
88         function(){
89             if(status=="closed"){
90                 status="open";
91                 $(this).text("gallery close");
92                 $(".gallery").addClass("active");
93             }else if(status=="open"){
94                 status="closed";
95                 $(this).text("gallery open");
96                 $(".gallery").removeClass("active");
97             }
98         }
99     );
100     $(".gallery li").hover(
101         function(){
102             $(this).addClass("over");
103         },
104         function(){
105             $(this).removeClass("over");
106         }
107     );
108 });
109 </script>
```

92행 : \$(".gallery").addClass("active");

아래 코드를 대신합니다.

```
$("#gallery").stop().animate({left:"0"}, 500, "easeOutBack");
```

active 클래스는 아래와 같은 스타일입니다.

```
.gallery {  
  -webkit-transition: left 0.5s ease-out;  
  transition: left 0.5s ease-out;  
}  
.gallery.active {  
  left: 0;  
}
```

102행 : `$(this).addClass("over");`

아래 코드를 대신합니다.

```
$(this).animate({opacity:1}, 300);
```

over 클래스는 아래와 같은 스타일입니다.

```
.gallery li {  
  -webkit-transition: opacity 0.3s;  
  transition: opacity 0.3s;  
}  
.gallery li.over {  
  opacity: 1;  
}
```