1장 jQuery 라이브러리 사용하기

제이쿼리는 "적게 쓰고 더 많이"라는 라는 슬로건 가진 자바스크립트 라이브러리입니다. HTML 문서 안에 클라이언트 자바스크립트의 DOM 객체에 쉽게 접근하고 매우 단순하게 프로그래밍을 할 수 있습니다. 이러한 제이쿼리는 존 레식에 의해, 2006년 뉴욕시 바캠프(Barcamp NYC)에서 공식으로 소개된 이후 지금까지 가장 사랑받는 자바스크립트 라이브러리라 할 수 있습니다.

1-1 라이브러리란

우리는 제이쿼리를 자바스크립트의 '라이브러리'라고 합니다. '라이브러리'이란 무엇일까요?
라이브러리는 보통 프로그램 언어로 만들기에는 많은 시간과 고난도의 기술이 필요한 작업을 쉽고 간편하게 작업을 할 수 있도록 도와주는 명령어 셋트 입니다. 즉 필요한 작업을 명령어로 묶어 놓은 함수의 집단이라고 생각하면 됩니다.

프로그램 라이브러리란?

예를 들어 봅시다. 우리가 알고 있는 자바스크립트로 페이지가 로딩하면 요소 안에 <p>요소가 생성되도록 합시다.

```
10    window.addEventListener("DOMContentLoaded", function(evt) {
11        const CONTENT = document.getElementById("content");
12        const P = document.createElement("p");
13        var TEXT = document.createTextNode("welcome 자바스크립트");
14        P.appendChild(TEXT);
15        CONTENT.appendChild(P);
16    });
```

자바스크립트로 작업을 하면 다음과 같이 작업이 됩니다.
그럼 이번에는 제이쿼리로 작업을 해 봅시다.

```
18    $("document").ready(function() {
19    $("#content2").append("<p>welcome 자바스크립트</p>");
20    });
```

같은 작업을 해도 간단하고 쉽게 작업을 할 수 있습니다.
이렇게 라이브러리로 작업을 하면 자바스크립트로 작업을 하는 것 보다 몇 개의 명령어로 쉽고 간단하게 작업을 할 수 있습니다. 현재 사용되고 있는 라이브러리의 종류는 여러 가지입니다. React, Vue, Ember, Angular, Node 등 여러 종류의 라이브러리가 사용되고 있습니다. 라이브러리는 각자 자신만의 특성화된 파트를 가지고 있습니다. 모든 라이브러리가 같은 기능을 하는 것은 아닙니다. 그래서 우리가 라이브러리를 사용하기 위해서는 그 특성을 먼저 파악하고 적재적

소에 알맞게 사용해야 합니다.

jQuery 이용하기

제이쿼리를 사용하기 위해서는 먼저 제이쿼리 라이브러리 함수가 필요합니다. 이 파일은 제이쿼리에 필요한 명령어들의 실행문들이 정리되어 있어 우리가 작업하기 이전에 먼저 이 명령어 집합의 파일을 연결해야만 합니다. 이 실행문이 있어야 우리가 작업하는 제이쿼리 명령어들과 필요한 실행문들이 실행됩니다.

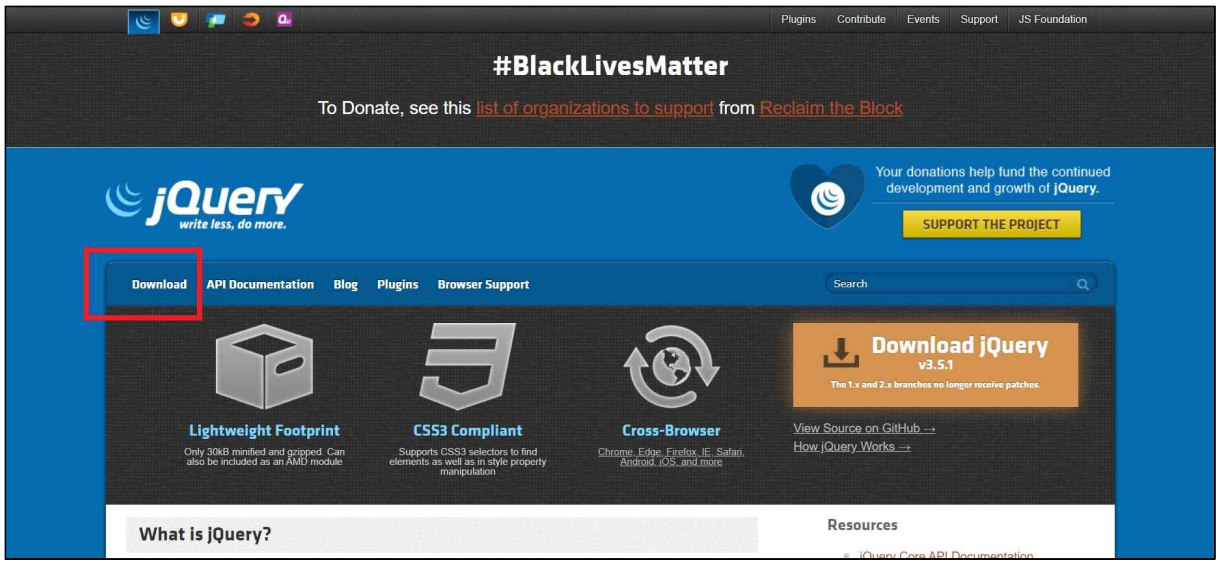
그럼 이 라이브러리 함수부터 연결해 봅시다.

먼저 사이트로 이동합니다.

http://jquery.com/

jquery 함수 다운로드 하기

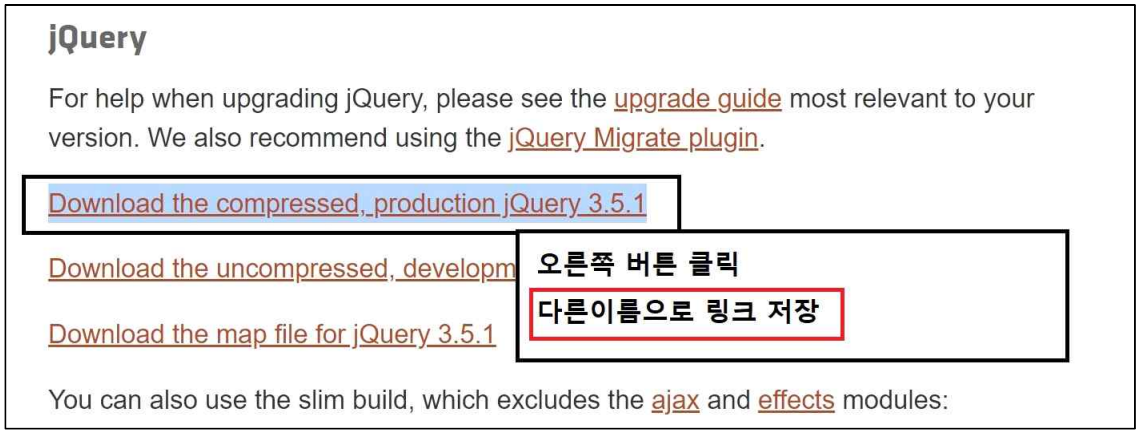
사이트를 방문하면 다운로드 메뉴가 보입니다. 이곳에서 제이쿼리 함수를 내려받을 수 있습니다.



제이쿼리 라이브러리는 두가지 종류입니다.

jquery.js (compressed production jQuery)	제이쿼리 라이브러리 원본으로 개발자용이라고 불립니다. 비압축 버전으로 실제 제이쿼리 개발 파일입니다.
jquery.min.js (uncompressed, development jQuery)	사용자용 파일이라 불리며 실제 원본 파일의 공간문자, 줄 바꿈이 생략된 압축 파일입니다. 원본 파일에 비해 메모리가 작아서 일반적으로 사용됩니다.

우리는 이 두 파일 중에 압축 버전을 사용할 겁니다.



만약에 제이쿼리 함수를 저장하여 사용할 수 없는 경우가 있습니다. 이럴 경우는 CDN사용을 권장합니다. 단 이 CDN을 사용할 때는 꼭 인터넷이 연결된 컴퓨터에서 사용하여야만 됩니다.

CDN 사용하기

CDN은 보통 인터넷 서비스 제공자(ISP, Internet Service Provider)에 직접 연결해 데이터를 전송하는 방식으로 제이쿼리 라이브러리를 연결할 수 있는 서비스를 제공하는 곳의 주소를 이용합니다.

제이쿼리 사이트(모든 버전 가능)

제이쿼리 사이트	https://code.jquery.com/

일반적인 개발자 도움 사이트

Google	https://developers.google.com/speed/libraries#jquery
Microsoft CDN	https://docs.microsoft.com/en-us/aspnet/AJAX/cdn/overview#jQuery_Releases_on_the_CDN_0
CDNJS CDN	https://cdnjs.com/libraries/jquery

제이쿼리 다운로드 페이지에서 연결된 주소로 이동 가능합니다.

Other CDNs

The following CDNs also host compressed and uncompressed versions of jQuery releases. Starting with jQuery 1.9 they may also host sourcemap files; check the site's documentation.

Note that there may be delays between a jQuery release and its availability there. Please be patient, they receive the files at the same time the blog post is made public. Beta and release candidates are not hosted by these CDNs.

- Google CDN
- Microsoft CDN
- CDNJS CDN
- jsDelivr CDN

HTML문서에 제이쿼리 라이브러리 파일 불러들이기

제이쿼리 라이브러리를 사용하기 위해서는 연결하고 제이쿼리 라이브러리를 HTML 문서에 불러들이기를 해야 합니다.

불러들이는 방법은 html문서의 <head>영역에 <script>요소를 이용하여 삽입합니다.

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>제이쿼리 시작하기</title>
    <script src="파일경로/제이쿼리라이브러리파일명.js"></script>
</head>
```

이 책의 실습파일에는 각 해당 폴더에 js폴더를 생성하고 jquery.min.js 파일을 셋팅 해 놓았습니다.


```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>제이쿼리 시작하기</title>
    <script src="js/jquery.min.js"></script>
</head>
```

제이쿼리 라이브러리 버전

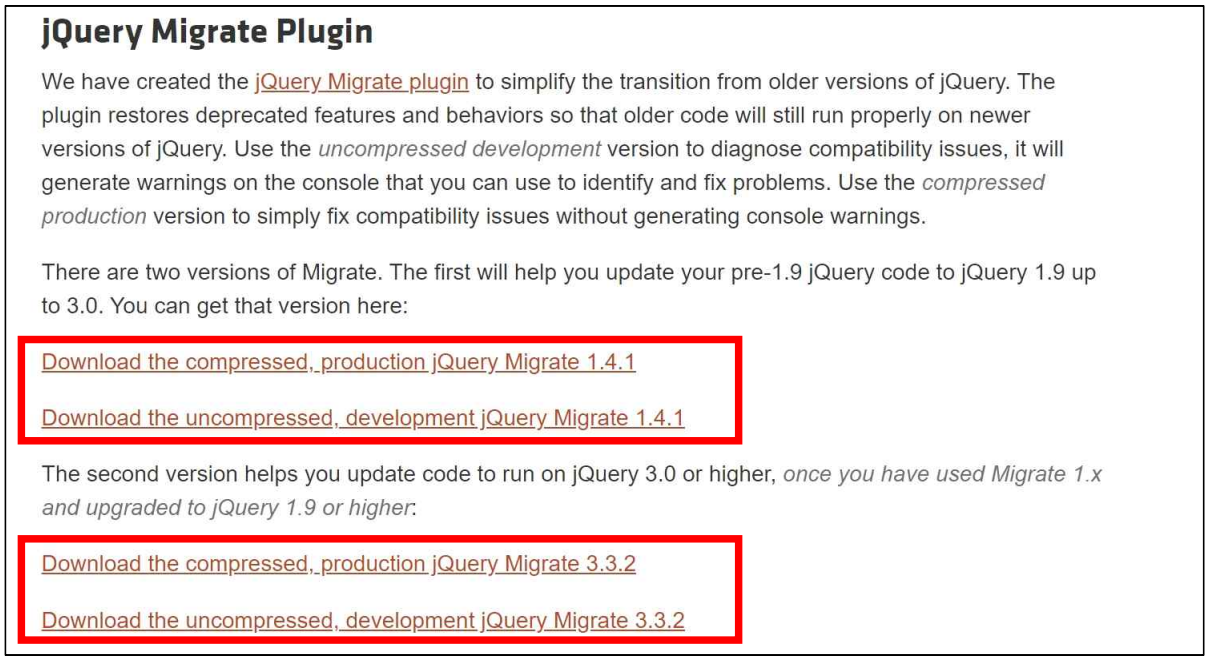
제이쿼리 라이브러리는 자주 업그레이드를 진행합니다. 그래서 이전 버전으로 사용 했던 기능이 삭제되거나 변경되는 일이 종종 발생합니다.

만약에 예전 버전의 명령어를 계속 사용하려면 Migrate plugin을 제리쿼리 라이브러리 다음에

연결해야 합니다.

Migrate plugin은 제이쿼리 라이브러리 다운로드 페이지에서 다운로드 할 수 있습니다.

1. https://jquery.com/download/에서 [Download the compressed, production jQuery Migrate 3.3.2]를 내려 받습니다.



jQuery Migrate Plugin

We have created the jQuery Migrate plugin to simplify the transition from older versions of jQuery. The plugin restores deprecated features and behaviors so that older code will still run properly on newer versions of jQuery. Use the *uncompressed development* version to diagnose compatibility issues, it will generate warnings on the console that you can use to identify and fix problems. Use the *compressed production* version to simply fix compatibility issues without generating console warnings.

There are two versions of Migrate. The first will help you update your pre-1.9 jQuery code to jQuery 1.9 up to 3.0. You can get that version here:

Download the compressed, production jQuery Migrate 1.4.1

Download the uncompressed, development jQuery Migrate 1.4.1

The second version helps you update code to run on jQuery 3.0 or higher, *once you have used Migrate 1.x and upgraded to jQuery 1.9 or higher*:

Download the compressed, production jQuery Migrate 3.3.2

Download the uncompressed, development jQuery Migrate 3.3.2

2. script 요소를 이용하여 먼저 제이쿼리 라이브러리 파일을 연결하고 그다음 migrate plugin 파일을 연결합니다.

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>제이쿼리 시작하기</title>
     <script src="js/jquery.js"></script>
    <script src="js/jquery-migrate-1.4.1.min.js"></script>
</head>
```

1-2 제이쿼리 서식

제이쿼리 라이브러리를 사용하기 위해 제이쿼리 라이브러리의 서식을 알아봅시다.

먼저 제이쿼리는 제이쿼리 라이브러리가 연결 되면 제이쿼리 라이브러리를 불러들이기를 해야 합니다. 그리고 html 문서가 준비되면 DOM의 객체를 이용하여 필요한 요소로 이동하여 작업을 진행합니다. 먼저

1. 제이쿼리 라이브러리 연결하기

```
<script src="../../js/jquery-3.4.1.min.js"></script>
```

2. 스크립트 요소를 이용하여 라이브러리 불러들이기

```
<script>
    jQuery(    );
</script>
```

jQuery(시점(작동))

3. 문서가 준비가 되면 실행

```
<script>
    jQuery( document.ready(      )  );
</script>
```

4. 실행문을 실행합니다. 이때 실행문은 그룹만들어 사용하므로 익명의 함수의 형식을 이용합니다.

```
<script>
    jQuery( document.ready( function ( ){실행문 ;}  )  );
</script>
```

화살표함수, 벡틱 사용 지양 / 결합연산자 위주로 사용

제이쿼리 라이브러에서는 'jquery'를 대신하여 '$'의 기호를 사용합니다. 그래서

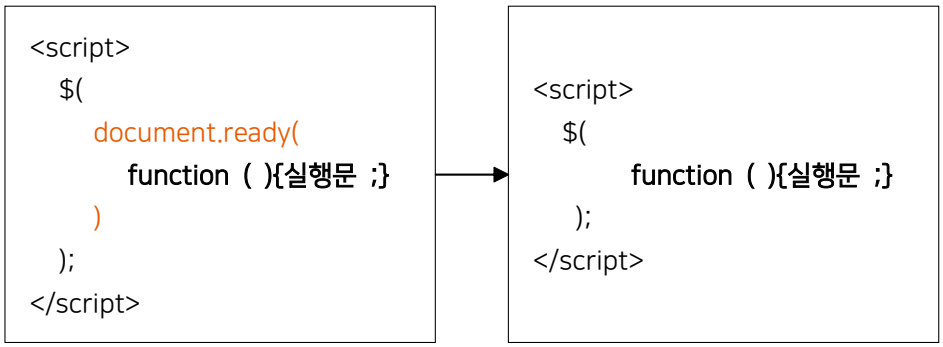
```
<script>
  jQuery(
    document.ready(
      function ( ){실행문 ;}
    )
  );
</script>
```

→

```
<script>
  $(
    document.ready(
      function ( ){실행문 ;}
    )
  );
</script>
```

로 바꾸어 사용 할 수 있습니다.

그리고 제이쿼리 함수를 불러들이고 body 문서를 확인 후 제이쿼리를 작동하므로 우리는 document.ready( )의 명령을 생략하여 사용하기도 합니다.



여기까지 정리를 하면 우리는 제이쿼리를 사용하기 위해서는 먼저 제이쿼리 라이브러리는 연결한 후에 라이브러리를 불러들여야 제이쿼리를 사용할 수 있습니다. 라이브러리를 연결 후 불러들이는 방법은 두 가지 형식으로 사용할 수 있습니다.

```
형식 1:
<script> $( document.ready( function ( ){실행문 ;}  )  ); </script>

형식 2:
<script> $( function ( ){실행문 ;} ); </script>
```

제이쿼리 기본 문장 만들기

제이쿼리는 먼저 DOM 객체에서 필요한 요소로 이동하여 작업을 시작합니다. 이 작업의 시작점을 선택자하고 합니다.

선택자는 '$( )'안에 따옴표(' ')를 사용하여 선택자 이름을 넣으면 됩니다. 이때 사용하는 따옴표는 홑 따옴표(' ')를 사용해도 되고 큰따옴표(" ")를 사용해도 무방합니다. 단지 앞에서 정리한 따옴표의 법칙만 잘 지켜지면 됩니다.

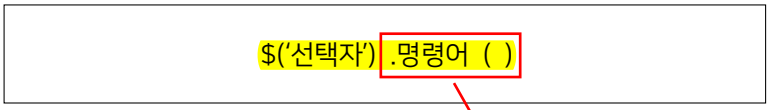
이렇게 먼저 선언한 선택자에 우리가 필요한 명령어를 마침표(.)를 이용하여 사슬 형태로 연결하여 사용할 프로그래밍하면 됩니다.

```
형식 1:
                $('선택자명') .명령어 ( )
형식 2:
          $('선택자명') .명령어 ( )  .명령어 ( )
```

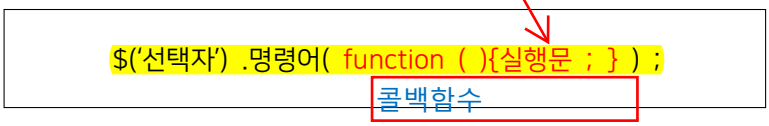
명령어 사용법

제이쿼리 명령어는 크게 두 가지로 나누어서 생각해 볼 수 있습니다.

1. 선택자에서 필요한 작업을 하는 명령어



2. 여러 실행문을 문장의 형식으로 작성해서 사용하는 명령어 이런 명령어는 실행문은 익명의 함수의 형식으로 바꾸어 사용합니다.



실습 파일 '01/01_01.html'파일을 열어 봅시다. (실습 파일: 01/01_01.html. 완성 파일: 01/all/01_01.html)

1. body 요소 안의 html을 확인합니다.

```
14    <button type="button">sample</button>
15    <div> </div>
15    <p></p>
```

2. 그럼 먼저 제이쿼리 라이브러리를 연결합니다.

제이쿼리 라이브러리는 각 폴더의 js 폴더가 만들어져 있고 그 폴더 안에 'jquery.min.js' 파일이 있습니다. head 요소 안에 script 요소를 이용하여 라이브러리를 연결합니다.

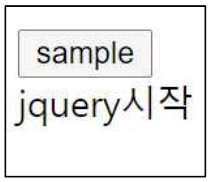
```
6    <!--준비하기-->
7    <script src="js/jquery.min.js"></script>
```

3. body 요소가 끝나기 전에 script 요소를 만들어 제이쿼리 라이브러리를 불러들입니다.

```
16    <script>
17      $(function(){
18
19      });
20    </script>
```

4. 먼저 $('div')선택자에 text('jquery시작') 명령어를 적고 브라우저에서 확인합니다.

```
21    <script>
22      $(function(){
23        $('div').text('jquery시작');
24      });
25    </script>
```



결과 확인하기

5. 이번에는 $('button')선택자에 .click()명령어를 넣고 그 안에 익명의 함수를 넣습니다.

```
17    $('button').click(function(e){
18
19        });
```

익명의 함수 안에 다음과 같이 선택자와 명령어를 넣습니다.

```
17    $('button').click(function(e){
18        $('p').text('welcome').css('color','red');
19        });
```

전체를 확인하면 다음과 같습니다.

```
14      $(function(){
15        $('div').text('jquery시작');
16
17        $('button').click(function(e){
18          $('p').text('welcome').css('color','red');
19        });
20      });
```

마지막으로 브라우저에서 버튼을 클릭하여 결과물을 확인합니다.

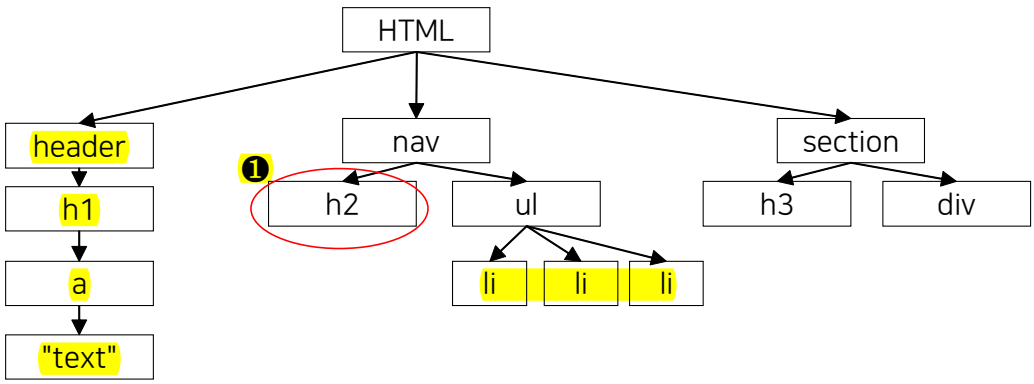
이처럼 제이쿼리에서 사용할 수 있는 명령어는 두가지로 나누어서 기억하면 좋겠습니다.

형식1	$('선택자명') .명령어 ( )
형식2	$('선택자') .명령어(function ( ){실행문 ; }) ;

그럼 이제 선택자에 대해서 알아봅시다.

1-3 선택자

제이쿼리 작업을 시작하기 위해서는 HTML 문서에서 원하는 DOM의 객체 즉 HTML 요소를 선택해야 합니다. 이런 선택의 기능을 선택자(Selection)라 합니다. 선택하는 방법은 직접적인 선택방법과 이미 선택한 선택자를 기준으로 인접해 있는 요소를 선택하는 인접선택방법이 있습니다. 인접선택방법을 HTML 문서를 동적으로 이동하며 선택한다고 해서 Traversing이라고 합니다.
노드트리를 보며 알아봅시다.



❶의 h2를 선택하는 방법은
직접적인 방법을 사용하면 $('nav>h2') 라고 선택할 수 있고, 또 인접선택방법을 사용하면 먼저 $('nav')를 요소를 선택한 후 체인을 이용하여 nav의 자식인 h2라고 하여 '.siblings('h2')'를 붙이는 방법입니다. 형식을 한번 보면

```
직접방법 :          $('nav>h2').css('background-color','red');
인접선택방법:       $('nav').siblings('h2').css('background-color','red');
```

다음과 같이 차이가 있습니다.
이번 장에는 일반적인 직접선택방법을 이용한 선택자에 대해 알아보도록 합시다.

기본선택자

기본선택자는 일반적인 css1, css2, css3버전에서 사용하는 선택자를 제이쿼리를 이용하여 선택하는 방법입니다.
형식은 알아보면

```
형식: $('기본선택자')
```

선택자	사용법	설명
요소 선택자	$('p')	지정한 p 요소모드
아이디 선택자	$('#main')	id속성값 main과 일치하는 요소
클래스 선택자	$('.sub')	class속성값 sub와 일치하는 요소 선택
종속 선택자	$('div p)	div 에 있는 자식또는 자손인 p 요소 선택
자식 선택자	$('div>h2)	div 자식인 h2 요소 선택
이웃 선택자	$('h2+p')	h2 다음에 있는 모든 형제 중 p 요소 선택
형제 선택자	$('h2~article')	h2 바로 뒤에 위치한 형제 article 요소 선택
그룹 선택자	$('#main, .sub,p')	아이디 main 과 클래스 sub 그리고 p 요소 모두 선택
모든 선택자	$('*')	모든 요소 선택

탐색선택자 사용하기

탐색선택자는 기본선택자를 선택한 요소 중 원하는 요소를 필터링하여 한 번 더 선택하는 방법입니다. 배열의 인덱스(index)처럼 위치를 기준으로 선택하는 방법과 선택된 요소의 속성과 text 노드를 검색하여 선택하는 방법이 있습니다. 위치 선택자부터 살펴봅시다.

위치 선택자는 css3 선택자를 그대로 사용하는 경우와 제이쿼리 선택자를 사용하는 경우 두가지로 나누어서 사용할 수 있습니다.

css3 선택자는 index의 순서가 1부터 시작이고, 제이쿼리 선택자는 index의 순서가 0부터 시작합니다. 먼저 css3 선택자를 이용한 선택 방법에 대해 알아봅시다.

1. css3 선택자

```
css 선택자 -> 1, 2, 3 ...
jquery 선택자 -> 0, 1, 2, 3 ...
```

선택자	사용법	설명
:first-child	li:first-child	부모 요소 기준으로 첫 번째 li 요소 선택
:last-child	li:last-child	부모 요소 기준으로 마지막 번째 li 요소 선택
:nth-child(index) :nth-child(n+index)	li:nth-child(2) li:nth-child(3n+2)	부모 요소 기준으로 두 번째 li 요소 선택 부모 요소 기준으로 세 개씩 묶어서 묶음 중 각각 두 번째 li 요소 선택
:nth-last-child(index) :nth-last-child(n+index)	li:nth-last-child(2) li:nth-last-child(3n+1)	부모 요소 기준으로 마지막 요소부터 시작하여 두 번째 li 요소 선택 부모 요소 기준으로 마지막 요소부터 시작하여 세 개씩 묶어서 묶음 중 각각 두 번째 li 요소 선택
:nth-of-type(index) :nth-of-type(n+index)	li:nth-of-type(2) li:nth-of-type(3n+1)	선택한 요소 li 중 두 번째 li 요소 선택 선택한 요소 li 중 세 개씩 묶어서 묶음 중 각각 두 번째 li 요소 선택
:nth-last-of-type(index) :nth-last-of-type(n+index)	li:nth-last-of-type(2) li:nth-last-of-type(3n+1)	선택한 요소 li 중 마지막 요소부터 시작하여 두 번째 li 요소 선택 선택한 요소 li 중 마지막 요소부터 시작하여 세 개씩 묶어서 묶음 중 각각 두 번째 li 요소 선택

index번호 1	index번호 2	index번호 3	index번호 4	index번호 5
li:nth-child(1)	li:nth-child(2)	li:nth-child(3)	li:nth-child(4)	li:nth-child(5)

2. 제이쿼리 선택자

선택자	사용법	설명
:even	li:even	li 중 index 번호가 짝수 번째인 li 선택
:odd	li:odd	li 중 index 번호가 홀수 번째인 li 선택
:eq(index)	li:eq(2)	li 중 index 2번째 li 선택
:gt(index)	li:gt(2)	li 중 index 2번째보다 뒤에 있는 li 모두 선택 2번째 제외 그 이후 것들
:lt(index)	li:lt(2)	li 중 index 2번째보다 앞에 있는 li 모두 선택 2번째 제외 그 이전 것들
:first	li:first	li 중 첫 번째 li 선택 :first-child
:last	li:last	li 중 마지막 li 선택 :last-child
:only-child	span:only-child	부모 요소 안에서 span 요소가 하나인 span 선택

index번호 0	index번호 1	index번호 2	index번호 3	index번호 4
li:eq(0)	li:eq(1)	li:eq(2))	li:eq(3)	li:eq(4)

기타선택자

선택자	기능설명
:has(요소)	특정 요소를 포함하는 요소 선택
:contains('text노드')	특정 text를 포함한 요소 선택
:empty	빈요소 선택 빈공간이라도 하나 있어도 선택되지 않음
:parent	다른 요소를 포함한 요소를 찾는다. 빈칸 하나만 들어 있어도 선택된다.
:header	제목 요소를 찾는다. hn요소 선택
:not(조건)	조건에 해당되지 않는 요소만 선택

속성선택자

요소의 속성의 값으로 선택하는 방법입니다.

선택자	사용법	설명
$('[속성]')	$('a[href]')	a 요소 중 href 속성이 있는 a 요소 선택
$('[속성="값"]')	$(title="main"]')	id속성과 값이 "main"인 요소 선택
$('[속성*="값"]')	$('div[id*="bird"])	id 속성에 값이 "bird"가 포함 되어 있는 요소 선택
$('[속성$="값"]')	$('div[id$="rd"])	id속성의 값이 'rd'으로 끝나는 요소 선택
$('[속성^="값"]')	$('div[id^="bi"])	id속성에 값의 'bi'로 시작되는 요소 선택
$('[속성!="값"]')	$('div[id!="bird"])	id속성에 값이 'bird'가 아닌 div 모두 선택
$('[속성1="값1"] [속성2="값2"]')	$('div[id="bird"] [title^="aclass"])	id 값이 bird이고 title속성의 값이 "aclass"로 시작되는 div 요소선택

10

2장 DOM의 객체 변경하기

2-1 기본 콘텐츠의 변경

하위 text 노드, 요소노드 변경과 가져오기

종류		설명
$(선택자).text();	get	선택자의 하위 text 노드의 값을 읽어오기
$(선택자).text("내용")	set	선택자의 하위 text 노드 변경
$(선택자).html();	get	선택자의 하위 요소 노드의 값을 읽어오기
$(선택자).html("<></>")	set	선택자의 하위 요소 노드 변경

선택자의 text내용을 가져옴

$(선택자).text( )

$(선택자).text("내용")

선택자의 text내용을 바꿈

선택자 안의 하위 요소를 가져옴

$(선택자).html( )

$(선택자).html('<> </>')

선택자 안의 요소를 새로 바꿈

요소의 속성 변경하기

종류	설명
$(선택자).attr("속성명")	선택자의 속성의 해당 값을 가져온다. getAttribute
$(선택자).attr("속성명","속성값")	선택자의 속성의 값을 변경한다. setAttribute
$(선택자).removeAttr("속성명")	선택자의 속성값을 제거한다.

선택자의 src속성의 값 변경

$('div').attr('src','../images/flower.jpg');

$('a').attr('href');

선택자의 href값 가져오기

요소의 스타일 변경하기

종류	설명
$(선택자).css("속성명")	선택자 스타일의 해당 값을 가져온다. get
$(선택자).css("속성명","속성값")	선택자 스타일의 속성의 값을 변경한다. set
$(선택자).css({속성:'값',속성:'값'});	선택자에 여러 스타일 한번에 설정하기

1. 선택자에 여러 스타일 한 번에 설정하기

선택자에 여러 스타일을 한 번에 설정하는 방법은 두 가지 형식으로 사용할 수 있습니다.

```
형식1:
        $(선택자).css({속성:'값' , 속성:'값'});
                $('p').css({backgroundColor:'red', color:'green'});
형식2:
        $(선택자).css({'속성':'값' , '속성':'값'});
                $('p').css({'background-color':'red',color:'green'});
```

css() 명령어에서 속성명이 두 단어인 경우에는

Ⓐ 따옴표 사용할 때는 우리가 알고 있는 css 속성명을 그대로 사용하면 됩니다.

ex) css({ 'background-color' , 'red' });

Ⓑ 따옴표를 사용하지 않을 때는 '-'를 빼고 두 번째 단어를 대문자로 사용해야 합니다.

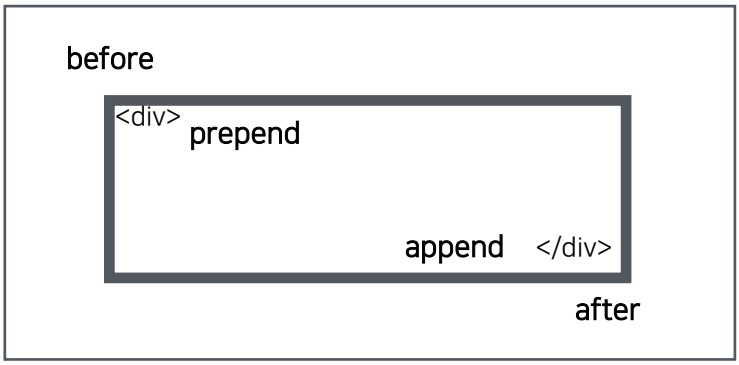
ex) css({ backgroundColor , 'red' });

클래스 사용하기

선택자	설명
$(선택자).addClass()	선택자에 class첨가
$(선택자).hasClass()	선택자에 특정한 class가 있는 지 찾기
$(선택자).removeClass()	선택자에 class제거
$(선택자).toggleClass()	선택자에 class추가 제거 반복

2-3 DOM 객체 변형하기

객체의 추가하기



선택자	설명
$(선택자).prepend("<></>")	선택자의 내부에 맨 앞에 자식 요소 추가 $('장소').위치('콘텐츠')
$(선택자).append("<></>")	선택자의 내부에 맨 뒤에 자식 요소 추가 $('장소').위치('콘텐츠')
$(선택자).prependTo("target")	선택자를 target의 맨 앞에 요소 삽입 $('이동대상').위치('장소')
$(선택자).appendTo("target")	선택자를 target의 맨 뒤에 요소 삽입 $('이동대상').위치('장소')

객체의 이동하기

선택자	설명
$(선택자).before("<></>")	선택자 앞에 노드 첨가 $('장소').위치('콘텐츠')
$(선택자).after("<></>")	선택자 뒤에 노드 첨가 $('장소').위치('콘텐츠')
$(선택자).insertBefore("target")	선택자를 target의 맨 앞에 삽입 $('이동대상').위치('장소')
$(선택자).insertAfter("target")	선택자를 target의 맨 뒤에 삽입 $('이동대상').위치('장소')

요소 묶기

시작요소만 넣어도 됨

7.태그 묶기	
$(선택자).wrap(<>)	선택자 각각을 요소로 감싸기
$(선택자).wrapAll(<>)	선택 집합 전체를 요소로 감싸기 -> DOM의 변화
$(선택자).wrallnner(<>)	선택자 안을 요소로 감싸기
$(선택자).unwrap()	감싸고 있는 태그 제거

객체의 바꾸기

선택자	설명
$(선택자).replaceWith()	선택자를 다른 요소로 변경한다.
$(선택자).replaceAll()	선택자로 요소를 변경한다. $('콘텐츠').replaceAll('선택자')

기타변형하기

선택자	설명
$(선택자).remove()	선택자와 일치하는 노드 제거
$(선택자).detach()	remove()와 같은 형상이지만 메모리에 남아 있기 때문에 다시 사용 할 수 있다. ctrl + x
$(선택자).empty()	선택자와 일치하는 노드 중 자식 노드들 제거
$(선택자).clone()	선택자와 똑같은 노드를 복제하여 기억한다. ctrl + c

03 탐색이동 매서드

이동 매서드의 형식

```
형식:
        $(선택자).이동 매서드().명령어(); ❶
        $(선택자).이동 매서드('selector').명령어();❷
```

❶ 선택자로 선택 후 이동 매서드를 이용하여 원하는 선택자 이동한다.

❷ 선택자로 선택 후 이동 매서드 이용하여 조건에 맞는 selector를 선택한다.

순번 이동 매서드

선택자	의미
.first()	선택한 요소 집합 중 첫 번째 요소를 찾는다.
.last()	선택한 요소 집합 중 마지막 요소를 찾는다.

위치 이동 매서드

명령	의미
.next()	선택한 요소의 바로 다음에 있는 형제 요소를 찾는다.
.nextAll()	선택한 요소 다음에 있는 형제 요소를 모두를 찾는다.
.prev()	선택한 요소의 바로 앞에 있는 형제 요소를 찾는다.
.prevAll()	선택한 요소의 바로 앞에 있는 형제 요소를 모두를 찾는다.

관계 이동 매서드

명령	의미
.children()	선택요소의 자식 요소를 모두 찾는다.
.parent()	선택한 요소의 부모 요소를 찾는다.
.parents()	문서 트리를 거슬러 올라가면서 선택 요소의 조상 요소를 모두 찾는다.
.parentsUntil(selector)	문서 트리를 거슬러 올라가면서 selector를 만날 때까지 조상 요소를 모두 찾는다. selector를 명시하지 않았다면 html요소까지 계속 찾는다.
.siblings()	선택한 요소의 형제 요소를 모두 찾는다.

탐색 기타 이동 매서드

명령	의미
.find(selector)	이미 선택한 요소의 자손 요소를 모두 찾는다. selector 넘겨서받 아서 찾을 자손 요소를 제한할 수도 있다. .children()메서드와 비슷하지만 찾는 범위가 넓으므로 더 느리다.
.not(selector)	선택한 요소 중에서 selector에 해당하는 요소를 제거한다.
.offsetParent()	CSS속성 top, left 등의 기준이 되는 조상 요소를 찾는다. 문서 트리를 거슬러 올라가면서 position 속성이 relative, absolute, fixed로 지정된 가장 가까운 조상 요소를 찾는다.
.selice(start, end)	선택한 요소 중 start +1번째 부터 END번째까지 찾는다. End는 생략 할 수도 있다.
.each(function(){실행문})	모든 요소 각각에 함수를 실행하는 메서드
add()	이미 선택한 요소에 다른 요소를 추가한다.

4장 이벤트:Event

4-1 이벤트 등록하기

형식

형식:

$(선택자). 이벤트명령어(function(e){ 실행문 ;}) ; ❶

$(선택자). 이벤트핸들러('이벤트명',function(e){실행문 ;}) ❷

❶ 이벤트 명령어는 하나의 선택자에 이벤트 하나를 등록하여 실행문을 실행하는 방법입니다.

❷ 이벤트핸들러는 하나의 선택자에 여러 이벤트를 등록하여 실행문을 실행 할 수 있습니다.

```
$('div#box_02>.button').dblclick(function(e){
   $(this).next('p').css('background','yellow');
 });
```

```
$('div#box_02>.button').on('click',function(e){
   $(this).next('p').css('background','yellow');
 });
```

이벤트 등록 매서드 사용하기

on( )	이벤트타입과 이벤트 핸들러를 설정할 수 있다.
one( )	이벤트 발생을 한 번만 실행한다. one("이벤트 타입",function(){ })

❶ 선택자의 이벤트가 발생되면 실행문을 실행합니다.

❷ 선택자의 이벤트가 여러번 발생하여도 한번 만 실행문이 실행하도록 합니다.

형식:

$(선택자). on('이벤트명',function(e){실행문 ;}); ❶

$(선택자). one('이벤트명',function(e){실행문 ;}); ❷

이벤트를 전달하지 못하는 상황이 생길 수 있다

$('영역').on('이벤트','선택자', function(){실행문})

이벤트 삭제하기

off()	이벤트를 삭제하기

❶ 선택자의 이벤트가 실행 되지 않게 합니다.

형식:

$(선택자). off('이벤트명'); ❶

강제로 등록하기

trigger()	선택자에 정의한 이벤트 함수를 강제로 실행하기
triggerHandler()	선택자에 정의한 이벤트 함수를 강제로 실행하지만 단, 첫번째 엘리먼트에만 작동한다. 즉 한번만 실행된다.

❶ 선택자의 이벤트가 실행 되지 않게 합니다.

형식:

$('selector').trigger('선택할 이벤트');

```
17    $('#testEle2').on('click', function() {
18      $('#testspan').trigger('click');
19    });
```

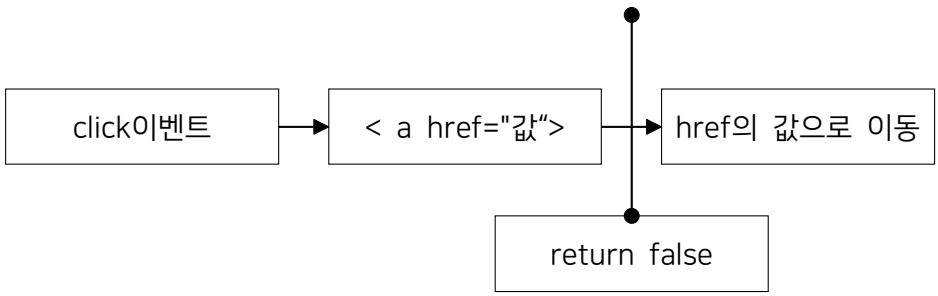
4-2 마우스 이벤트

카멜기법 X

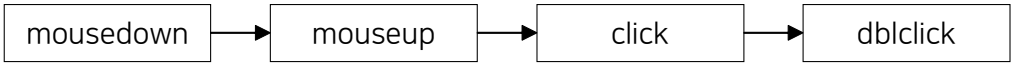
$(선택자).click()	선택자에 마우스 포인터가 눌렀다 떼었을 때
$(선택자).dblclick()	선택자를 더블 클릭하였을 때
$(선택자).mousedown()	선택자에 마우스 버튼을 눌렀을 때
$(선택자).mouseup()	선택자에 마우스 버튼을 떼었을 때
$(선택자).mouseenter()	선택자에 마우스가 진입했을 때
$(선택자).mouseleave()	선택자에 마우스가 벗어났을 때
$(선택자).hover()	mouseenter()와mouseleave()를 한 번에 사용할 때 형식: hover(function(){mouseenter할 일},function(){mouseleave할 일});
$(선택자).move()	마우스커서가 해당 선택자 위에서 움직일 때 실행된다.

<a>요소의 클릭 이벤트

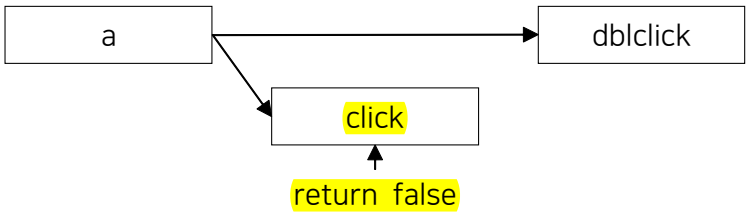
a요소는 클릭이라는 이벤트를 만나면 a 요소 속성중 href의 값으로 이동이 됩니다. 이런 a의 요소에 클릭하여 실행문을 실행하고자 하면 href의 값으로 이동을 막아야 합니다. 이 역할을 이벤트의 실행문이 끝나는 시점에 return false의 값으로 막아주면 됩니다.



마우스의 이벤트는 서로 상관관계를 가지고 있습니다. 예를 들면 click이벤트가 발생하려면 mousrdown 이벤트 먼저 발행되고 그다음 mouseup 이벤트가 발생하여야 click 이벤트가 완성이 됩니다. 또 더블클릭도 click이라는 이벤트가 두 번 발생이 되어야 버블 클릭 이벤트가 발생됩니다.



그럼 이번에는 a 요소에 더블클릭 이벤트가 발생이 되었을 때의 문제점 해결에 대해 알아보도록 합시다.
(실습 파일: 04/04_04.html. 완성 파일: 04/all/04_04.html)



on을 이용한 라이브이벤트 등록

on( ) 명령어를 이용한 이벤트 등록 방법은 이벤트를 등록한 이후에 새로 추가된 요소나 복제된 요소에는 이벤트가 등록되지 않습니다.

이런 경우에는 on( ) 명령어를 라이브방식으로 형식을 변경하면 됩니다.

```
형식:
$('이벤트 대상의 상위 요소').on('이벤트명','이벤트 대상',function(e){실행문 ;});
```

4-3 기타 이벤트

키보드 이벤트

명령어	설명
keydown()	선택한 요소에서 키보드를 눌렀을 대 이벤트가 발생
keyup()	선택한 요소에서 키보드를 키가 올라갈 때 이벤트가 발생
kypress()	선택한 요소에서 키보드를 키가 내려갈 때 이벤트가 발생

윈도우 이벤트

매서드	내용
resize()	브라우저의 가로 사이즈가 변경될 때 이벤트 발생
scroll()	브라우저의 스크롤이 움직일 때 이벤트 발생
load()	소스가 로딩될 때 이벤트 발생
ready()	HTML문서가 로딩이 완료되면 이벤트 발생

05 효과와 애니메이션

05-01 이펙트명령어

이펙트 매서드 사용하기

형식:
$(선택자).이펙트명령어(❶시간, '❷easing',❸콜백 함수function(){실행문;});

❶ 효과가 적용되는 시간을 나타낸다. 적용방법은 두가지로
1. 키워드값: slow, normal, fast
2. 밀리언초: 1초 1,000으로 계산

❷ 가속도를 설정한다. 필요가 없는 경우는 생략할 수 있다.
키워드값: linear(일정 속도로 유지), swing(조금씩 빨라 졌다 느리게 끝냄)

❸ 콜백 함수는 이펙트가 끝난 후에 실행할 실행문을 설정한다. 필요가 없는 경우는 생략할 수 있다. 익명의 함수로 설정해야 한다.

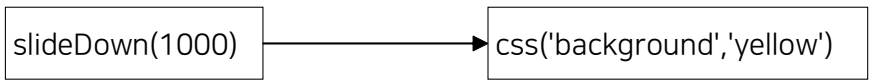
show()	요소를 시간에 맞추어서 나타나게 한다.
hide()	요소를 시간에 맞추어서 안 보이게 한다.
toggle()	show()와 hide() 자동 반복한다.
slideDown()	요소가 아래로 시간에 맞추어서 나타나게 한다.
slideUp()	요소가 위로 시간에 맞추어서 안 보이게 한다.
slideToggle()	sildeDown()과 slideUp()을 자동 반복한다.
fadeIn()	숨겨진 요소가 점점 투명도가 올라가면서 선명해 진다.
fadeOut()	요소가 점점 투명해지면서 숨겨진다.
fadeTp()	요소가 지정한 투명도가 적용된다.
fadeToggle()	fadeIn()와 fadeOut()를 자동 반복

보이기	show()	slideDown()	fadeIn()	display:block
안보이기	hide()	slideUp()	fadeOut()	display:none
토글	toggle	slideToggle	fadeToggle()	
기타			fadeTo()	

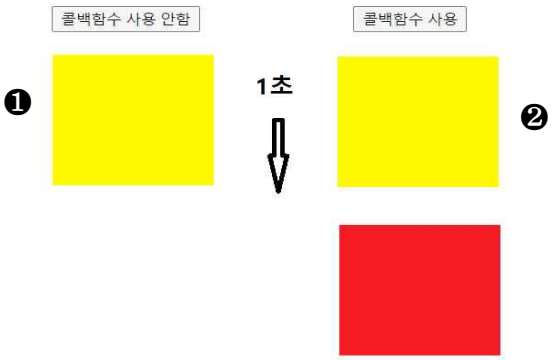
콜백함수

콜백 함수는 이펙트가 끝난 후에 실행할 실행문을 설정한다. 그럼 콜백 함수로 설정된 실행문과 일반적인 실행문과의 차이를 알아봅시다.

선택된 요소가 slideDown() 후에 스타일을 변경하고자 합니다.




```
27    <div id="box01"> ❶
28        <p><button class="btn01">콜백함수 사용 안함</button></p>
29        <p class="back back01"></p>
30    </div>
31    <div id="box02"> ❷
32        <p><button class="btn02">콜백함수 사용</button></p>
33        <p class="back back02"></p>
34    </div>
35    <script>
36        $(function(){
37        $('button.btn01').on('click',function(e){
38          $('.back01').slideDown(500);
39          $('.back01').css('background','yellow');
40        });
41        $('button.btn02').on('click',function(e){
42          $('.back02').slideDown(500,function(){
43              $(this).css('background','yellow');
44          });
45        });
46        });
47        </script>
```



$(:not(:animate)) 선택자 사용하여 광클릭 문제 해결하기

이펙트 명령어를 사용할 때 광클릭과 같이 이벤트가 여러 번 작동 시킬 수 있습니다. 이럴 경우에는 이벤트의 작동 수 만큼 이펙트도 계속 작동되어 문제가 발생됩니다. 이러한 광클릭을 막아주는 방법 중에 선택자로 해결하는 방법을 알아보도록 합시다.

형식:
$('선택자:❶not(❷:animated)).이펙트 명령어( )

❶ ':not(조건)'는 조건을 제외하고 선택하는 방법입니다. ❷ ':animated' 움직이는 요소만 선택합니다. 이 필터선택을 연결하여 사용하면 선택 된 움직이는 요소를 제외하고 선택을 하면 이펙트가 진행되는 이벤트가 계속 작동이 되어도 요소는 선택이 되지 않습니다.

실습을 통해 알아봅시다.(실습 파일: 05/05_02.html. 완성 파일: 05/all/05_02.html)

```
16    <button id="show">표시</button>
17    <button id="hide">비표시</button>
18    <div id="back_01">some forms of augmented reality can now be used in our daily lives
19    .......
20    </div>
21    <script>
22    $(function(){
23    $('#show').on('click',function(e){❶
24        $('#back_01:not(:animated)').slideDown(3000);❷
25    });
26    $('#hide').on('click',function(e){
27        $('#back_01:not(:animated)').slideUp(3000);
28    });
29    });
30    </script>
```

❶ #show를 클릭을 하면 ❷ #back_01이 움직이지 않을 때 선택을 하여 slideDown( )을 하도록 합니다. #show를 계속 클릭하여도 #back_01가 움직일 때는 선택이 되지 않아 광클릭이 하여도 이펙트 명령이 쌓이지 않게 됩니다.

5-2 animate( )

animate( )명령어는 이펙트와 같이 만들어져 있는 효과를 사용하는 것이 아니라 내가 원하는 스타일 속성을 사용하여 움직임을 만드는 명령어입니다.

형식:
$('선택자').animate({❶속성명: '값',속성명:'값'},❷소요시간,'❸가속도',❹콜백함수);

❶ 움직일 스타일은 설정합니다. 형식은

{ 속성명 : '값', 속성명:'값' }

{color:'red', Ⓐ backgroundColor:'yellow',fontSize:'3em'}
{color:'red', Ⓑ 'background-color':'yellow','font-size':'3em'}

자바스크립트 속성 설정 방법과 같이 중괄호{} 안에 Ⓐ속성명이 두 단어로 연결 되어 있으면 두 번째 단어를 대문자로 표시하는 방법과 Ⓑ 속성명을 따옴표"로 감싸는 방법이 있습니다. 이때 속성의 값은 무조건 따옴표"로 감싸 주어야 합니다.

❷ 소요시간

1. 키워드값: slow, normal, fast
2. 밀리언초: 1초 1,000으로 계산

❸ 가속도를 설정한다. 필요가 없는 경우는 생략할 수 있다.

키워드값: linear(일정 속도로 유지), swing(조금씩 빨라 졌다 느리게 끝냄)

❹콜백 함수는 이펙트가 끝난 후에 실행할 실행문을 설정한다. 필요가 없는 경우는 생략할 수 있다. 익명의 함수로 설정해야 한다.

애니메이션 제어 명령어

animate( )	스타일의 속성으로 움직이기해당
stop()	해당 선택자의 애니메이션 효과를 중간에 멈춘다.
delay()	설정된 값 만큼 지연했다가 애니메이션을 진행합니다.
queur()	큐에 사용자 정의 함수를 추가하거나 큐에 대기 중이 함수를 배열에 담아 반환합니다.
finish( )	해당되는 선택자의 애니메이션을 강제로 완료 시점으로 보낸 후 종료합니다.

6 크기와 위치

6-1 객체의 크기와 위치

객체의 크기

선택된 요소의 크기를 선정 또는 축출하는 명령어입니다.

형식:

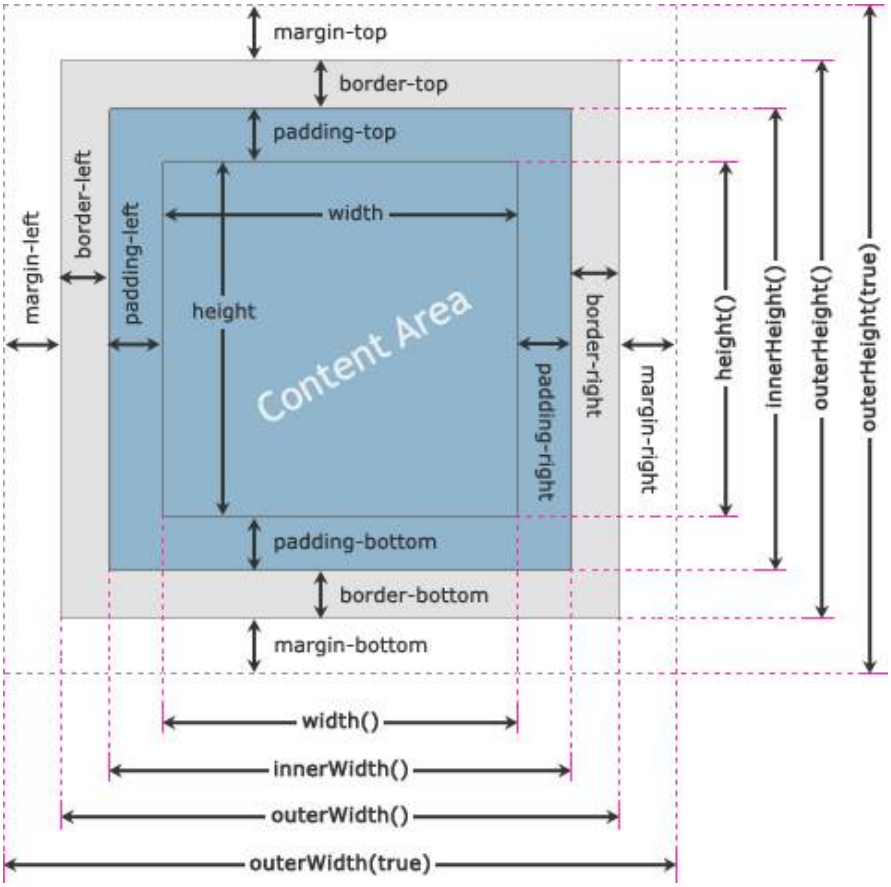
$('선택자').크기명령어(값) -> $('p').width(900); ❶ set

$('선택자').크기명령어() -> $('p').width( ); ❷ get

❶ p요소의 가로의 크기를 900px로 설정합니다.

❷ p요소의 가로의 크기를 가져옵니다.

매서드	내용
width()	선택자의 가로 크기를 설정 또는 축출합니다. 데이터의 자료형은 number데이터를 사용합니다.
height()	선택자의 세로 크기를 설정 또는 축출한다. $('선택자').height()
innerWidth()	padding값을 포함 한 가로 크기
innerHeight()	padding값을 포함 한 세로 크기
outerWidth() outerWidth(true)	border 값을 포함한 가로 크기 margin 값을 포함한 가로 크기
outerHeight() outerHeight(true)	border 값을 포함한 세로 크기 margin 값을 포함한 세로 크기



06-02 객체의 위치

offset()	선택자의 위치 값을 설정 또는 축출한다. $('선택자').offset({top:50,left:50}) $('선택자').offset().top $('선택자').offset().left
position()	선택자의 위치 값을 설정한다. $('선택자').position().top $('선택자').position().left

06-02 스크롤의 위치

scrollTop()	도큐멘트의 세로 스크롤 값
scrollLeft()	도큐멘트의 가로 스크롤 값

형식:

```
$(document).scrollTop(값) ->      ❶
$(document).scrollTop() ->        ❷
```

❶ DOM의 스크롤 top의 값을 설정한다.

❷ DOM의 스크롤 top의 값을 가져온다.

scroll animation 하기

형식:

```
$('html,body').animate({scrollTop:800,scrollLeft:1500},800,'swing'); ❶
```

❶ scroll animation을 할 경우에는 선택자를 document를 사용하지 말로 $('html,body')을 사용하애 IE에서는 문제가 해결됩니다.

07 폼

07-01 폼의 선택자

선택자	형식	설명
:input	$(":input")	모든 input 요소 선택
:text	$(":text")	모든 input 요소 중 type="text" 선택
:password	$(":password")	모든 input 요소 중 type="password" 선택
:radio	$(":radio")	모든 input 요소 중 type="radio" 선택 선택
:checkbox	$(":checkbox")	모든 input 요소 중 type="checkbox" 선택
:submit	$(":submit")	모든 input 요소 중 type="submit" 선택
:reset	$(":reset")	모든 input 요소 중 type="reset" 선택
:button	$(":button")	모든 input 요소 중 type="button" 선택
:image	$(":image")	모든 input 요소 중 type="image" 선택
:file	$(":file")	모든 input 요소 중 type="file" 선택
:enabled	$(":enabled")	모든 enabled input 요소 선택
:disabled	$(":disabled")	모든 disabled input 요소 선택
:selected	$(":selected")	모든 selected input 요소 선택
:checked	$(":checked")	모든 checked input 요소 선택

07-02 폼 이벤트와 매서드

폼요소의 이벤트 명령어

형식:

```
$(선택자) . focus(function(){원하는 행동});
$(선택자) . on('focus',function(){원하는 행동});
```

이벤트 명령어

명령어	내용
focus()	폼 컨트롤 요소에 마우스나 탭키가 선택된 상태를 감지 될 때
blur():	focus()와 반대개념으로 폼 컨트롤 요소에 focus가 벗어났을 때를 감지 될 때
change()	폼 컨트롤요소의 값이변경 된 것을 감지하는 명령 될 때
submit()	폼 태그의 전송 버튼을 눌렸을 때 발생하는 이벤트의 처리를 설정한다.

폼컨트롤 요소의 값 설정 및 사용하기

형식:

```
$(선택자).val( 값 ); ❶
$(선택자).val( );❷
```

❶ 선택자의 value값 넣기

❷ 선택자의 value값 가져오기