



하이브리드 : 제이쿼리 기초

제이쿼리 개요

jQuery란

HTML5 웹 문서 안의 스크립트 언어를 단순화하도록 설계된 자바스크립트 라이브러리이다.

자바 스크립트를 편하게 사용하기 위해 제이쿼리를 사용한다.

html 요소 선택이 쉽고, css 선택자를 알고 있으면 요소를 선택할 수 있다.

브라우저마다 생각하지 않아도 되고, 웹페이지를 조작하기 위해 만들어졌다.

즉, 제이쿼리는 웹페이지를 만들기 위해 필요하다.

jQuery 특성

1. 가장 인기 있는 자바 스크립트 라이브러리이다.
2. 크로스 브로우징(cross browsing)을 지원한다.
3. 생산성 향상을 지원한다.
4. DOM 트리 형식이다.

자바 스크립트 명세 방법

- 자바 스크립트 코드는 `<script>` 태그 안에 명세한다.
- `<head>` 태그 또는 `<body>` 태그 안에 포함시키거나 `<script src="sample.js">` 을 통해 별도의 자바스크립트 파일에 명세할 수도 있다.
- 자바 스크립트 코드는 명세된 순서대로 실행된다.

```

<html>
  <head>
    ... 생략 ...
    <script src="sample.js">
    </script>
    자바스크립트 코드2
  </head>
  <body>
    ... 생략 ...
    <script>
    자바스크립트 코드3
    </script>
    ... 생략 ...
  </body>
</html>

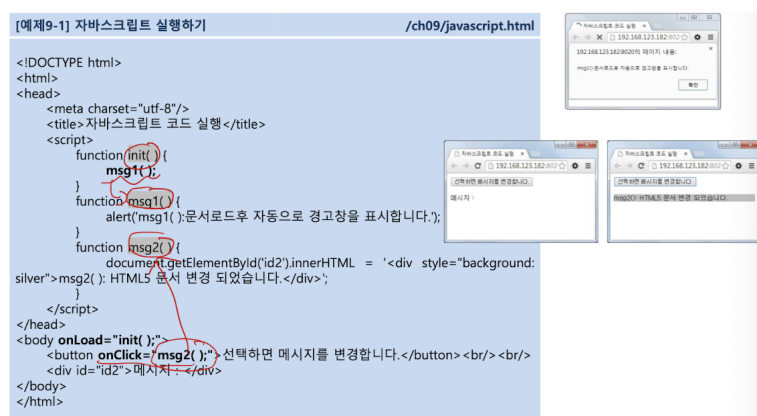
```

```

// sample.js
자바스크립트 코드1

```

• 자바 스크립트 실행 예제



제이쿼리 연동 방식

HTML5 문서 안에 사용할 제이쿼리 라이브러리 파일을 포함하도록 선언한다.
제이쿼리 라이브러리 파일의 URL 경로를 `<script>` 태그로 간단히 삽입한다.
제이쿼리 모바일 방식과 동일하다.

• CDN(Content Delivery Network) 연결 방식

제이쿼리 사이트에서 실시간으로 다운로드한다.

'jquery-1.11.1.min.js'파일은 앞으로 사용할 제이쿼리 라이브러리를 포함한다.

항상 인터넷과 연결되어 있어야한다는 제약, 최신 버전의 라이브러리를 사용할 수 있다.

• 다운로드 받는 방식

```

<!DOCTYPE html>
<html>
<head>
  <script src="http://code.jquery.com/jquery-1.11.1.min.js"></script>
  <script>
    /* 자바스크립트나 제이쿼리 코드를 명세하는 영역 */
  </script>
</head>
<body>
  ... 생략 ...
</body>
</html>

```

- 제이쿼리 실행 예제

[예제9-2] 제이쿼리 실행하기 /ch09/jquery.html

```

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
  <title>제이쿼리 코드 실행</title>
  <script src="http://code.jquery.com/jquery-1.11.1.min.js"></script>
  <script>
    $(document).ready( function() {
      msg1();
      $('#id1').click( function() {
        msg2();
      });
    });
    function msg1() {
      alert('msg1() :문서로드후 자동으로 경고 창을 표시합니다. ');
    }
    function msg2() {
      $('#id2').css('background','silver');
      $('#id2').text('msg2() :HTML문서 변경되었습니다. ');
    }
  </script>
</head>
<body>
  <button id="id1">선택하면 메시지를 변경합니다.</button><br/><br/>
  <div id="id2">메시지 : </div>
</body>
</html>

```

자바 스크립트와 제이쿼리 코드 비교

자바스크립트 코드	제이쿼리 코드
필요 없음	<script src="http://code.jquery.com/jquery-1.6.4.min.js"></script>
function init() { msg1(); } ... 생략 ... <body onLoad="init();"> ... 생략 ...	\$(document).ready(function() { msg1(); }); ... 생략 ... <body> ... 생략 ...
... 생략 ... <button onClick="msg2();"> ... 생략 ...	\$('#id1').click(function() { msg2(); }); ... 생략 ... <button id="id1"> ... 생략 ...
function msg2() { document.getElementById("id2").innerHTML = '<div style="background:silver">msg2() :HTML문서 변경되었습니다.</div>'; msg2(); } ... 생략 ... <div id="id2"> ... 생략 ...	function msg2() { \$('#id2').html('<div style="background:silver">msg2() : HTML문서 변경되었습니다.</div>'); } ... 생략 ... <div id="id2"> ... 생략 ... function msg2() { \$('#id2').css('background','silver'); \$('#id2').text('msg2() :HTML문서 변경되었습니다. '); } ... 생략 ... <div id="id2"> ... 생략 ...

DOM(Document Object Model; 문서 객체 모델)

HTML5와 같은 구조화 된 문서를 객체 개념으로 표현하고 접근하는 방식이다.

W3C가 정한 공식 표준으로 W3C가 표준화한 여러 API의 기반이다.

DOM API를 통해 HTML5와 XML 문서 안의 구성 요소이다. 즉, 엘리먼트, 속성, 텍스트 등을 접근하고 변경할 수 있어 문서의 내용이나 구조, 스타일 등을 동적으로 제어가 가능하다. 플랫폼과 언어에 종립적이며 동적으로 문서의 내용, 구조, 스타일을 접근하고 변경하는 중요한 수단이다.

- 자바 스크립트의 비호환성 문제가 발생

웹 브라우저 안에 자바스크립트 엔진이 제조사에 따라 일부 다르게 동작하는 문제를 DOM으로 해결할 수 있다.

DOM이라는 공통된 표준 모델(표준 API)을 따르면 스크립트 엔진이나 언어가 다르더라도 얼마든지 동일하게 HTML5이나 XML과 같은 구조화된 문서에 접근이 가능하다.

DOM 트리 구조

- HTML5이나 XML 문서의 구성 요소들은 계층적인 상호 관계를 갖고 구성된다.

DOM 모델에서 각 객체(HTML 문서의 구성 요소)들은 자연스럽게 계층적인 트리 구조로 표현한다.

- 노트 트리

1. 엘리먼트 노트

태그와 내용으로 표현되는 엘리먼트를 표현하는 노트이다.

하위에 또 다른 엘리먼트를 포함한다.

2. 속성 노트

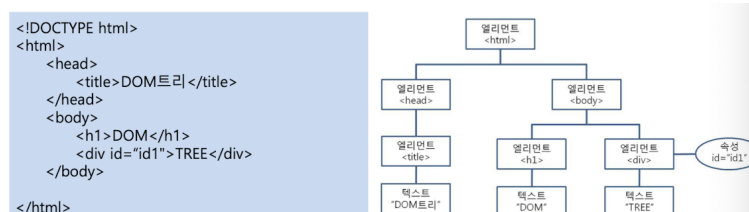
엘리먼트의 시작 태그 안에 포함되는 속성 이름과 속성 값을 쌍으로 갖는 노트이다.

3. 텍스트 노트

엘리먼트가 갖는 실제 내용을 의미하는 노트이다.

DOM트리에서 최하위 단말 노트가 해당된다.

- DOM 트리 구조 예



제이쿼리 함수

제이쿼리 기본 기능

DOM API들을 활용하는 제이쿼리 메소드들의 사용방법과 밀접하게 관련있다.

Core(핵심 개념) : jQuery() 선언 함수 정의 및 활용방법

Selectors(선택자) : DOM 트리의 노드 선택 표현식

CSS(스타일) : CSS 스타일 속성값 변경 메소드

분류	기능	지원 메소드	다루는 장
Core	핵심개념	jQuery() 선언 함수 정의 및 활용 방법	9장
Selectors	선택자	DOM 트리의 노드 선택 표현식	
CSS	스타일	CSS 스타일 속성값 변경 메소드	
Traversing	탐색	DOM 트리의 계층 구조를 이용한 노드 탐색 메소드	10장
Manipulation	조작	DOM 트리의 노드 변경 메소드	
Attributes	속성	엘리먼트 속성값의 조회 및 변경 메소드	
Events	이벤트	마우스, 키보드, 폼 및 문서 관련 이벤트 메소드	10장
Effects	효과	동적 스타일 변화를 위한 메소드	
Ajax	비동기교환방식	Ajax 관련 메소드	11장
UI	사용자 인터페이스	사용자 인터페이스용 라이브러리	-

jQuery() 함수

모든 시작은 jQuery() 함수를 통해 이루어진다.

괄호 앞의 'jQuery'문자열은 제이쿼리를 사용한다는 일종의 선언이다.

자바스크립트 문장과 제이쿼리 문장을 구별하는 역할이다.

- jQuery() 함수의 3가지 입력 인자 유형

1. 선택자 입력 인자

```
jQuery('body p');
```

1. HTML 입력 인자 (요소 중 하나로 코드를 넣을 수 있다.)

```
jQuery('<h1>HTML 동적 추가</h1>').appendTo('body');
```

1. 함수 입력 인자

```
jQuery(function() {alert("DOM 트리가 생성됨");});
```

문서 시작 이벤트 핸들러

- 문서 시작 이벤트 핸들러

```
jQuery(document).ready(function(){...});
```

'document' 선택자는 HTML5 문서 전체를 선택한다.

ready() 메소드는 HTML5 문서가 웹브라우저에 표시될 준비가 완료되면 실행되는 이벤트 핸들러이다.

function() 함수 안에 실행되기 원하는 제이쿼리 코드를 추가한다.

웹 브라우저가 HTML5 전체 문서의 DOM트리를 생성하고 준비가 되었음을 알려주면

function()은 콜백 함수로서 스스로 호출되어 실행한다.

jQuery(document).ready() 안에 제이쿼리 실행문들을 ';'으로 구분하여 나열하면 DOM 트리 구성이 완료된 후 까지 실행이 지연되었다가 자동으로 호출되어 순차적으로 실행된다.

-> C언어의 main() 함수와 같은 역할이다.

- 제이쿼리 코드를 `<head>` 태그 안에 많이 명세할 때 문제점

`<body>` 태그 안의 HTML5 엘리먼트 태그들이 DOM 트리를 구성하기 전에 먼저 실행된다.

-> 제이쿼리 코드가 처리할 DOM트리는 구성되지도 않았는데 코드가 먼저 실행될 수 있다.

제이쿼리를 포함한 문서를 처음 시작하는 2가지 방법

- jQuery(document).ready(function(){...});

DOM이 준비된다. (문서 100% 로드 되기 전)

페이지 DOM만 로드 완료된 이후 수행된다.

.ready() 메소드가 여러 개 있더라도 순서대로 실행한다.

비교적 첫 페이지 로드 시간이 빠르다.

- jQuery(window).load(function(){...});

DOM이 준비된다. (문서 100% 로드된 후)

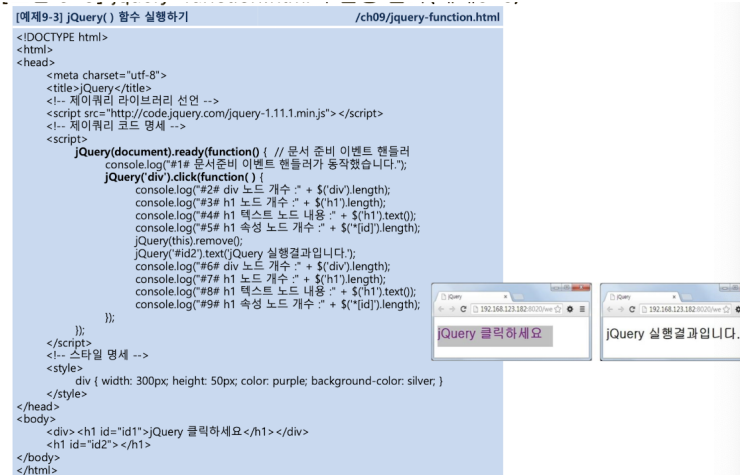
페이지 DOM + 리소스 전체(문서에 포함된 이미지 등의 모든 객체)가 모두 로드 완료된 이후 한 번만 수행한다.

.load() 메소드가 여러개 있으면 마지막만 실행한다.

기존 자바스크립트 window.onload = function(){...}를 대체한다.

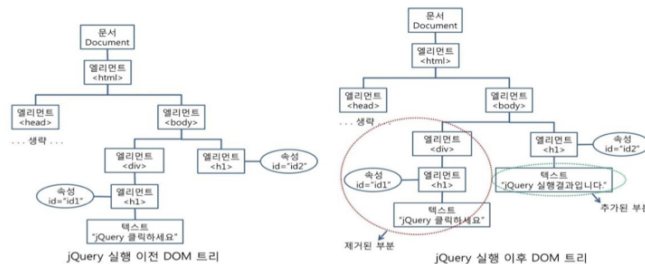
비교적 첫 페이지 로드 시간이 느리다.

제이쿼리 함수 실행하기 예제

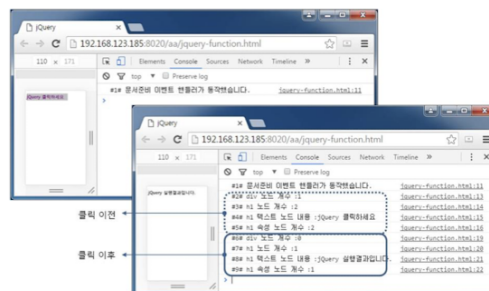


제이쿼리 실행 결과

- DOM 트리 변화



- 콘솔 창의 변화



제이쿼리 함수의 단축형

- \$ 기호

'jQuery'예약어의 별칭이다. 제이쿼리 변수, 함수를 자바 스크립트의 변수, 함수 등과 구별하는 역할이다.

- \$()

제이쿼리 함수 또는 제이쿼리 래퍼(wrapper)이다.

코드 안 'jQuery' 함수 식별이 어렵고 불편하기 때문에 줄여서 '\$()'로 표시한다.

'제이쿼리 객체'와 '제이쿼리 객체 집합'

-> \$(선택자) 형태로 제이쿼리 선택자를 입력 인자로 받아 들여 선택자 조건을 만족하는 엘리먼트 노드들을 DOM 트리에서 찾아 '제이쿼리 객체'형식으로 반환한다.

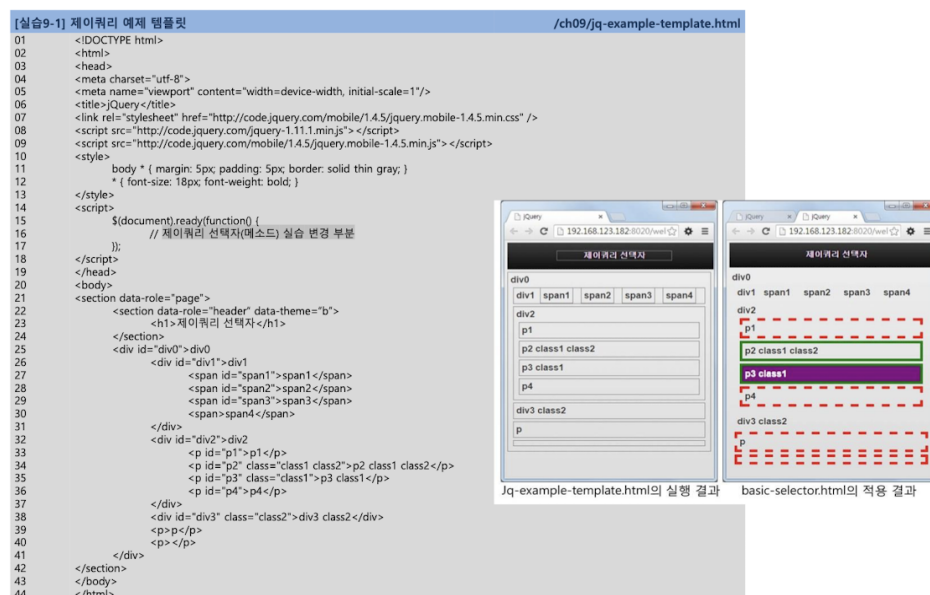
-> 일치하는 DOM 노드들이 여러 개이면 배열 형태의 '제이쿼리 객체 집합'을 반환한다.

-> 반환하는 DOM 엘리먼트들을 제이쿼리 객체 개념으로 감싸고 미리 준비된 메소드를 사용할 수 있도록 확장한다.

```
jQuery('div').click();
-> $('div').click();
```

• 제이쿼리 시작 이벤트 핸들러

```
jQuery(document).ready(function(){...});
-> $(document).ready(function(){...});
-> $(function(){...});
```



제이쿼리 선택자

기본 선택자

가장 많이 사용되는 선택자로 형식은 CSS3 선택자와 동일하다.

\$(선택자)'는 선택자 조건을 만족하는 DOM 노드 들을 제이쿼리 객체 집합으로 반환한다.

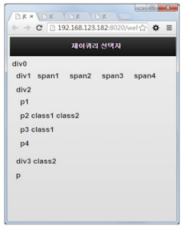
제이쿼리 메소드와 제이쿼리 이벤트 등에서 사용하는 기본요소이다.

- : 모든 엘리먼트를 선택한다. (ex) \$(' '))
 태그명 : 특정 엘리먼트를 모두 선택한다. (ex) \$('div'), \$('h1, h3, p'))
 #아이디 : 고유한 아이디 속성값을 가진 엘리먼트를 선택한다. (ex) \$('#id1'),
 \$('div#id1'))
 .클래스 : 특정 클래스 속성값을 가진 엘리먼트들을 선택한다. (ex) \$('.class1'),
 \$('div.class1'), \$('.class1.class2'))
- 예시

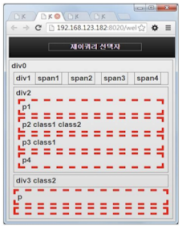
[예제9-4] 제이쿼리 기본 선택자 실행하기
/ch09/basic-selector.html

```


$('*').css('border-style', 'hidden');           // (1)
$('p').css('border', 'dashed thick red');       // (2)
$('.class1').css('border', 'solid thick green'); // (3)
$('#p3').css({'background-color': 'purple', 'color': 'white'}); // (4)
        
```




(1)



(2)



(3)



(4)

계층 선택자

DOM 트리에서 노드 사이의 상하 관계나 형제 관계를 이용해 노드를 선택한다. 형식은 CSS3선택자와 동일하다.

선택자 > 자식선택자 : 특정 엘리먼트 바로 밑에 위치한 하위 엘리먼트(자식 엘리먼트)를 선택한다. (ex) \$('ul > li'))

선택자 자손선택자 : 특정 엘리먼트 안에 포함된 모든 하위 엘리먼트(자식 엘리먼트)를 선택한다. (ex) \$('div p'))

선택자 + 형제선택자 : 특정 엘리먼트에 바로 다음에 나오는 특정한 형제 엘리먼트 하나를 선택한다. (ex) \$('li#id1 + li'))

선택자 ~ 형제선택자 : 특정 엘리먼트 다음에 나오는 특정한 모든 형제 엘리먼트들을 선택한다. (ex) \$('li#id1 ~ li'))

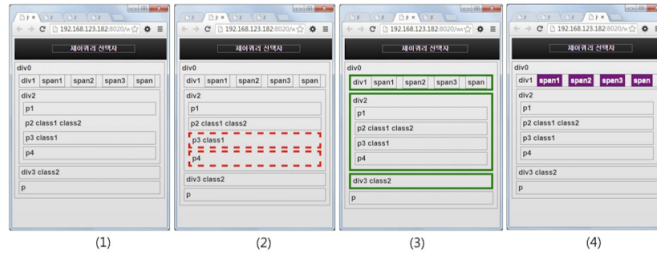
- 예시

[예제9-5] 제이쿼리 계층 선택자 실행하기
/ch09/hierarchy-selector.html

```

$('#p1 + p').css('border-style', 'hidden'); // (1)
$('#p2 ~ p').css('border', 'dashed thick red'); // (2)
$('div > div').css('border', 'solid thick green'); // (3)
$('div span').css({'background-color': 'purple', 'color': 'white'}); // (4)

```



속성 선택자

속성 관련 제이쿼리 선택자도 CSS# 선택자와 동일하다.

선택자 **[속성명]** : 해당 속성을 포함하는 엘리먼트들을 선택한다.

선택자 **[속성명 = "조건값"]** : 해당 속성값이 조건값과 일치하는 엘리먼트들을 선택한다.

선택자 **[속성명 *= "조건값"]** : 해당 속성값이 조건값을 포함하는 엘리먼트들을 선택한다.

선택자 **[속성명 ^= "조건값"]** : 해당 속성값이 조건값으로 시작하는 엘리먼트들을 선택한다.

선택자 **[속성명 \$= "조건값"]** : 해당 속성값이 조건값으로 끝나는 엘리먼트들을 선택한다.

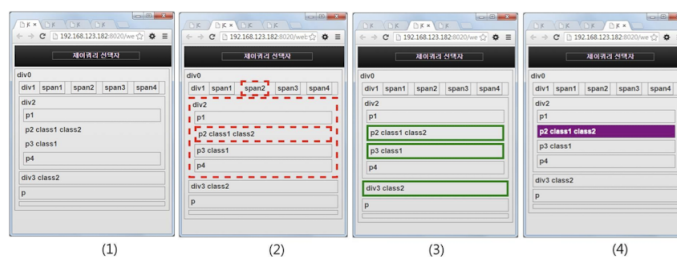
- 예시

[예제9-6] 제이쿼리 속성 선택자 실행하기
/ch09/attribute-selector.html

```

$('p[class]').css('border-style', 'hidden'); // (1)
$('*[id="2"]').css('border', 'dashed thick red'); // (2)
$('*[class^="cla"]').css('border', 'solid thick green'); // (3)
$('p[id$="2"]').css({'background-color': 'purple', 'color': 'white'}); // (4)

```



위치필터 기본 선택자

필터(filter)는 선택자에 의해 선택된 엘리먼트들 중에서 필터 조건에 맞는 엘리먼트들로 선택 대상을 좀 더 제한(한번 더 거르는)하는 선택자 유형이다.

선택자 뒤에 콜론(:)을 붙여 명세한다.

반환된 엘리먼트들은 자바스크립트의 배열 첨자를 적용하므로 첨자가 '0'부터 부여된다.

- 예) :first는 :eq(0)과 :lt(1)과 그 의미가 모두 같음

형식	기능(첨자가 0부터 시작됨)
선택자:first	선택된 엘리먼트들 중에서 첫 번째 엘리먼트를 선택 예) \$('li:first')
선택자:last	선택된 엘리먼트들 중에서 마지막 엘리먼트를 선택 예) \$('li:last')
선택자:odd	선택된 엘리먼트들 중에서 홀수 첨자 엘리먼트들을 선택 예) \$('li:odd')
선택자:even	선택된 엘리먼트들 중에서 짝수 첨자 엘리먼트들을 선택 예) \$('li:even')
선택자:eq(n)	선택된 엘리먼트들 중에서 첨자 n인 엘리먼트를 선택 예) \$('li:eq(4)') (다섯 번째 li 태그)
선택자:lt(n)	선택된 엘리먼트들 중에서 첨자 n인 엘리먼트보다 앞 엘리먼트들을 선택 예) \$('li:lt(3)') (첫 번째, 두 번째, 세 번째 li 태그들)
선택자:gt(n)	선택된 엘리먼트들 중에서 첨자 n인 엘리먼트보다 뒤 엘리먼트들을 선택 예) \$('li:gt(3)') (다섯 번째 li 태그를 포함한 이후의 li 태그들)
선택자:not()	선택된 엘리먼트들 중에서 필터링 조건을 만족하지 않는 엘리먼트들을 선택 예) \$('li:not(even)') (홀수 번째 노드들을 선택)