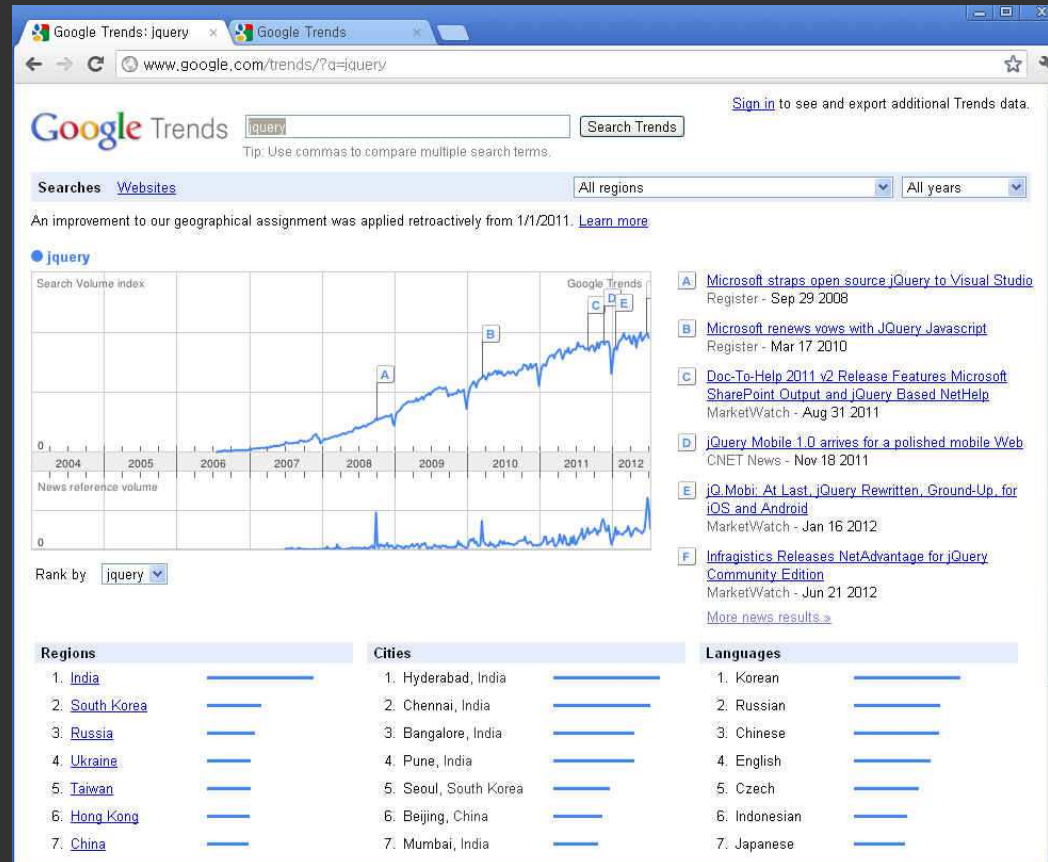

'Write Less, Do more.'

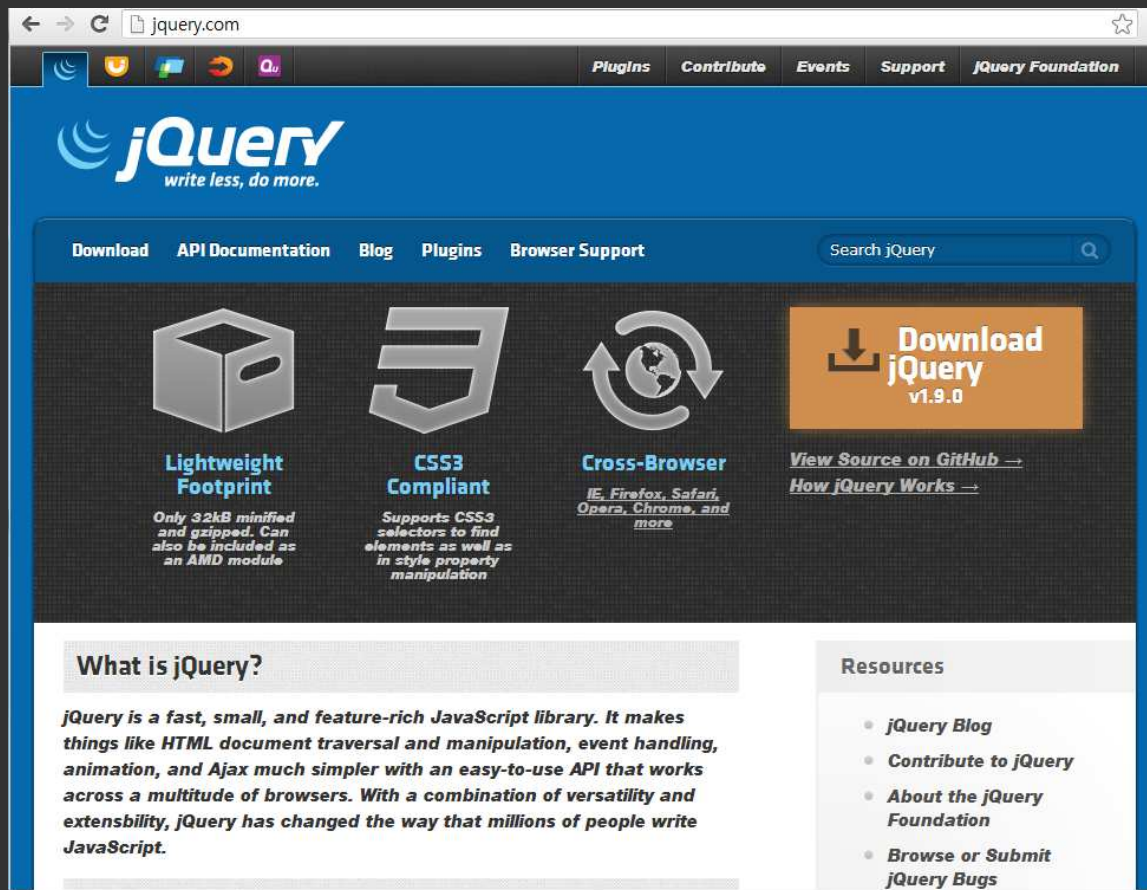
“jQuery는 HTML 속 클라이언트 사이드 스크립트 언어를 단순화 하도록 설계된 브라우저 호환성이 있는 자바스크립트 라이브러리이다. 존 레식에 의해, 2006년 뉴욕 시 바캠프(Barcamp NYC)에서 공식으로 소개되었다. jQuery는 오늘날 가장 인기있는 자바스크립트 라이브러리 중 하나다”

출처: 위키백과 사전
<http://ko.wikipedia.org/wiki/JQuery>



<http://www.google.com/trends/>

jQuery 준비



<http://jquery.com/>

jQuery 설치

```
<html>
<head>
  <script type="text/javascript" src="jquery.js"></script>
  <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js"></script>

  <script type="text/javascript">
    // 사용자가 코딩할 자바스크립트/jQuery 공간
  </script>
</head>

<body>
</body>
</html>
```

jQuery를 공부하기 위한 선수 학습

- JavaScript
 - DOM (Document Object Model)
 - HTML/CSS
-

JavaScript 탄생배경

넷스케이프

브라우저 전쟁

표준 DOM

JavaScript 프로그램기초

변수

함수

배열 / 객체

이벤트

연산자

제어문

반복문

타이머 메서드

변수

- 값을 담는 그릇인 저장소
- 변수에 포함된 값은 언제든지 변경가능

var 변수명 = 값;

```
var a = 10;
```

```
var name = "홍길동";
```

변수명 생성시 주의할 점

- JavaScript 내부적으로 미리 사용하는 키워드나 예약어는 사용할 수 없음

- 변수 이름 앞에 숫자 또는 특수 기호/문자는 제외

3tot = 45 (X) // 첫 글자에 숫자는 올 수 없음.

-count = 32 (X) // 특수 기호는 올 수 없음.

var = 3 (X) // 키워드 예약어는 변수명으로 올 수 없음.

변수의 범위

- 변수가 선언되는 위치에 따라 변수의 유지범위가 달라짐

total 변수는 함수 내부에 존재하지만 var 선언이 되어 있지 않기 때문에 자동으로 전역 변수가 되어 전체 영역에 영향을 미칩니다.

```
<script>  
var myAge = 10;  
function myFun( ) {  
    var temp = 10;  
    total = temp + 20;  
}  
function myFun2( ) {  
    var temp = 20;  
}  
</script>
```

myAge 변수는 변수 선언(var)이 <script> main 공간에 선언되었기 때문에 전역 변수로 전체 영역에 영향을 미칩니다.

temp 변수는 function() 내부에 변수가 선언되었기 때문에 지역 변수로 함수 내부에만 영향을 미칩니다.

함수

함수는 어떤 목적을 위해 만들어진 코드를 하나로 묶어 반복되는 작업을 효율적으로 관리하는 것이라고 보면 됩니다.

- 함수 정의
 - 함수 호출(invocation)
 - 함수 인자(arguments and parameters)를 활용한 함수 확장
 - 함수 종료 및 리턴 값
-

함수 정의

선언적 함수

```
function 함수이름 ( ) {  
    // 실행할 코드들  
}
```

```
function sum ( ){  
    var a = 10;  
    var b = 20;  
    var c = a + b;  
}
```

익명 함수(anonymous function)

```
var 변수 = function ( ) {  
    // 실행할 코드들  
}
```

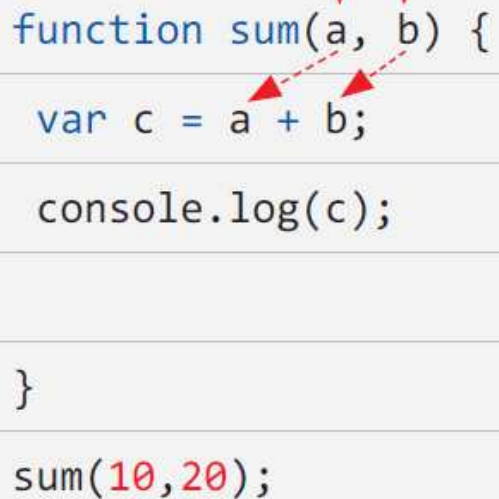
```
var sum = function ( ){  
    var a = 10;  
    var b = 20;  
    var c = a + b;  
}
```

함수 호출

함수는 정의만 해서는 절대로 수행되지 않습니다. 다시 말해서, 함수가 정의되어 메모리에 담겨 있을 뿐, 실제로 실행되는 것이 아닙니다. 정의된 함수를 수행하려면 반드시 함수를 호출해야 합니다.

```
sum( );
```

매개변수를 활용한 함수 확장



```
function sum(a, b) {  
  var c = a + b;  
  console.log(c);  
}  
sum(10, 20);
```

The diagram illustrates the execution of the `sum` function. A dashed red box encloses the function definition and its call. Red arrows show the flow of data: two arrows point from the arguments `10` and `20` in the function call `sum(10, 20);` to the parameters `a` and `b` in the function definition `function sum(a, b) {`. Another red arrow points from the expression `a + b` in the function body to the variable `c` in the assignment `var c = a + b;`.

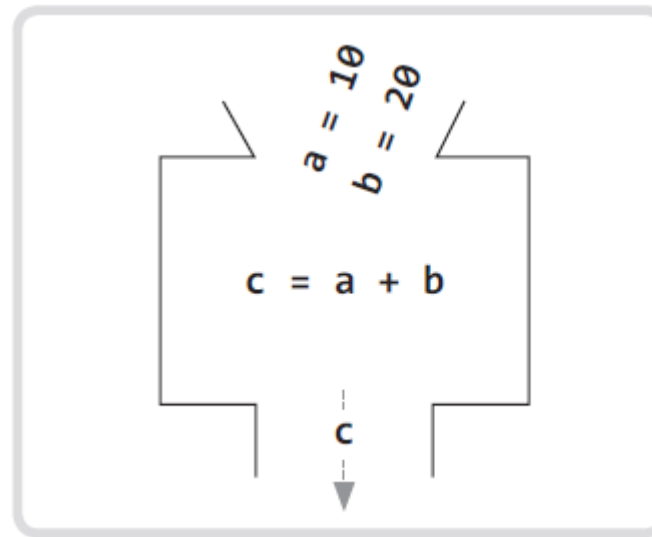
함수를 선언할 때 () 안에 변수를 지정할 수 있는데 이것을 매개변수라 합니다.

이렇게 매개변수를 사용하면 함수에서 사용할 DATA를 외부에서 입력해 줄 수 있기 때문에 함수의 확장이 가능합니다.

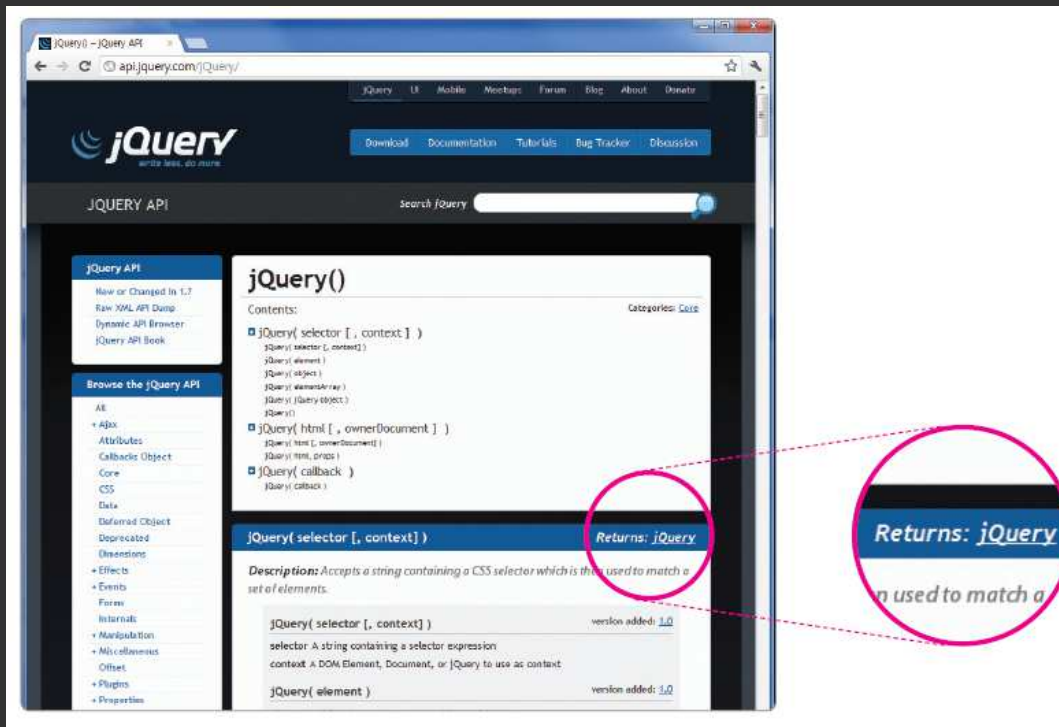
함수 종료 및 리턴 값

함수를 수행하고 난 후에 함수의 결과를 얻어 사용하려면 return 을 사용하여 결과 값을 전달합니다.

```
function sum(a, b) {  
  var c = a + b;  
  return c;  
}
```



jQuery 함수의 리턴값 확인



jQuery 함수들을 살펴보면
항상 확인해야 할것이
바로 returns 값의 데이터형
태 입니다. 함수마다 결과값
을 여러 데이터 형태로 돌려
주는 것을 확인하셔야 합니
다.

배열 (Array) – []

배열은 하나의 변수에 여러 개의 데이터를 담을 수 있는 그릇이라고 보면 됩니다. 배열은 여러 개의 변수를 인덱스 번호로 관리하며 [] (사각괄호)를 사용해 표현합니다. Index는 0부터 시작합니다.

arr = [

"a"	1	{ obj }	fn()
-----	---	---------	------

]

[0] [1] [2] [3]

생성 및 원소 추가

- new 생성자 사용
- 배열 객체를 생성하면서 요소 추가
- [] 연산자를 사용하여 직접 생성

jQuery () 의 결과도 배열로 관리

```
var arr = $('div.mark')
```

arr =

```
<div class="mark"> </div>  
<div class="star"> </div>  
<div class="mark"> </div>  
<div id="car"> </div>
```

<div class="mark"> </div>

<div class="mark"> </div>

jQuery wrapper 집합

객체(Object) – { }

객체는 한 개 이상의 데이터를 담을 수 있는 그릇이라고 보면 배열과 상당히 비슷하지만 배열보다 명확하게 값을 관리할 수 있습니다. 객체타입은 {} (중괄호)를 사용해 표현하고 인덱스 대신에 '이름:값' 형식을 사용합니다.

obj = {

str="a"	age=1	s={ obj }	sum=fn()
---------	-------	-----------	----------

 }

생성 및 원소 추가

- new 생성자 사용
- Object 객체를 생성하면서 요소 추가
- {} 연산자를 사용하여 직접 생성

jQuery에서 Object 객체 사용 예

```
$("#div").css({'backgroundColor': '#C2F5BF', 'borderColor': 'yellow',  
'marginBottom': '10px'});
```

```
$("#div").css({  
  'backgroundColor': '#C2F5BF',  
  'borderColor': 'yellow',  
  'marginBottom': '10px'  
});
```

```
var obj = {  
  'backgroundColor': '#C2F5BF',  
  'borderColor': 'yellow',  
  'marginBottom': '10px'  
};  
$("#div").css(obj);
```

jQuery에서 함수의 매개변수 형식으로 Object를 많이 사용합니다. 보이는 예제와 같이 여러값을 하나의 변수에 담아 보내는 모습을 많이 볼 수 있습니다.

이벤트

프로그래밍에는 반드시 이벤트가 있습니다. 이벤트는 말 그대로 '어떤 사건'입니다. JavaScript에서 사건(event)이라는 말의 의미는 아래와 같습니다.

- 사용자가 버튼이나 태그를 클릭하는 사건
 - 웹 브라우저의 창이 줄어드는 사건
 - 텍스트를 입력하기 위해 텍스트 필드를 클릭하는 사건
 - 마우스를 클릭하는 사건
 - 마우스를 움직이는 사건
 - 키보드를 입력하는 사건
-

JavaScript 이벤트 처리 방식

- 이벤트 등록

obj.attachEvent("이벤트 이름", 함수) // *IE(Internet Explorer)*

obj.addEventListener("이벤트 이름", 함수)

- 이벤트 제거

obj.detachEvent("이벤트 이름", 함수)

obj.removeEventListener("이벤트 이름", 함수)

JavaScript 이벤트 처리 방식

- 이벤트 등록

obj.attachEvent("이벤트 이름", 함수) // *IE(Internet Explorer)*

obj.addEventListener("이벤트 이름", 함수)

- 이벤트 제거

obj.detachEvent("이벤트 이름", 함수)

obj.removeEventListener("이벤트 이름", 함수)

JavaScript 이벤트 처리 방식

JavaScript	
1	<code>function fnClick(){</code>
2	<code> alert("hello");</code>
3	<code>}</code>
4	<code>var evtDiv = document.getElementById('box');</code>
5	<code>if(evtDiv.addEventListener) {</code>
6	<code> evtDiv.addEventListener("click", fnClick);</code>
7	<code>} else {</code>
8	<code> evtDiv.attachEvent("onclick", fnClick);</code>
9	<code>}</code>

jQuery 이벤트 처리 방식

- 이벤트 등록

```
$("#obj").click ( 함수 ) ;
```

```
$( "# obj" ).bind ( "click" , 함수 );
```

```
$( '# obj' ).bind( 'mouseenter mouseleave' , 함수 );
```

- 이벤트 제거

```
$( "# obj" ).unbind( "click" ); // id가 obj 객체에 click 이벤트를 제거
```

```
$( "# obj" ).unbind( ); // id가 obj 객체에 모든 이벤트를 제거
```

jQuery 이벤트 처리 방식

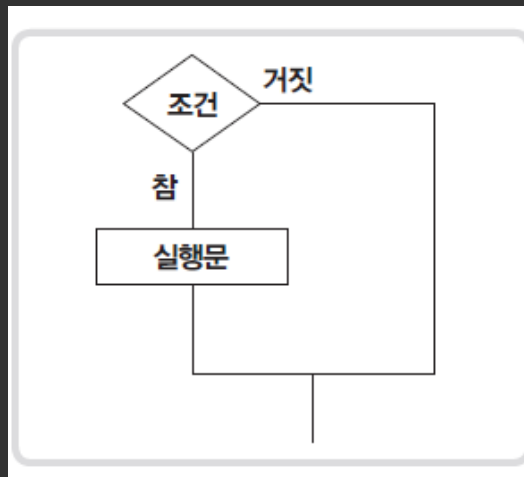
jQuery	
1	function sum(){
2	alert("hello");
3	}
4	\$('#div').click(sum);

jQuery	
1	function sum(){
2	alert("hello");
3	}
4	\$('#div').bind("click",sum);

조건문

조건문은 프로그래밍의 흐름을 제어하는 것으로, 대표적인 예로는 if문, switch문을 들 수 있습니다. 여기서는 if문에 대해 알아보겠습니다

if문



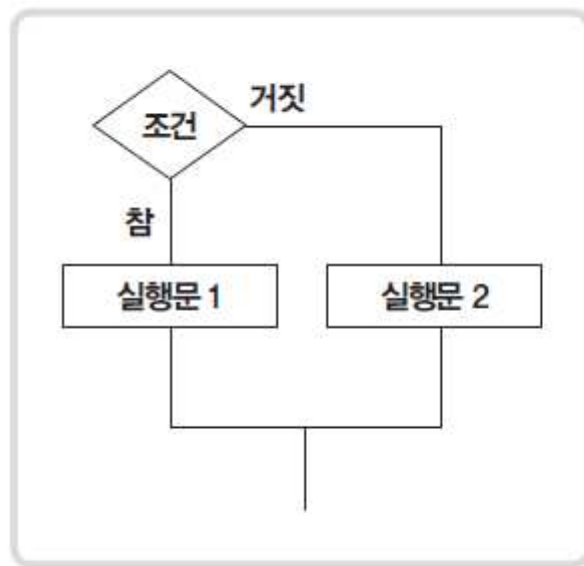
```
if (조건) {  
    // 실행문  
}
```

예

```
if( 4 < 5 ) {  
    alert ("참"); // if 조건이 참이므로 코드가 실행됩니다.  
}
```

if else문

if문의 조건이 참일 때와 거짓일 때 수행할 문장을 각각 따로 정할 수 있으며, 2개의 실행문은 절대로 함께 수행될 수 없습니다.



```
if (조건) {  
    // 실행문 1;  
} else {  
    // 실행문 2;  
}
```

예

```
if( 2 > 9 ) {  
    alert("참"); // if 조건이 거짓이므로 코드가 실행되지 않습니다.  
} else {  
    alert("거짓");// if 조건이 거짓이므로 else 코드가 실행됩니다.  
}
```

반복문

반복문은 아주 짧은 시간에 같은 명령을 반복하는 것으로, 대표적인 예로는 for문과 while문을 들 수 있습니다. 여기서는 for문에 대해 알아보겠습니다.

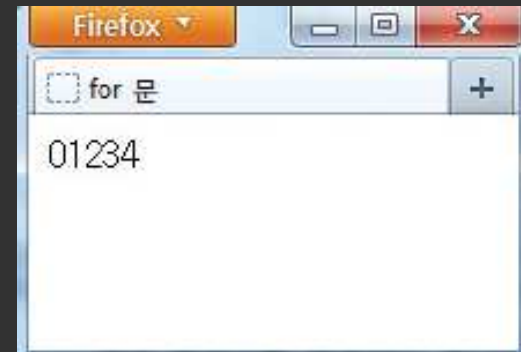
```
for (초기값; 조건식; 표현식) {  
  
    // 실행문;  
  
}
```

초기값 : for문을 시작할 때 선언하는 변수이며, 1개 이상 가능합니다.

조건식 : 조건이 거짓이면 반복을 멈춥니다.

표현식 : 초기값에 선언된 변수 값을 다양한 수식으로 변경할 수 있습니다.

```
for(var i=0; i<5; i++) {  
    document.write( i );  
}
```



for문의 반복 순서는 아래와 같습니다.

- ① 초기 값을 처리합니다.
- ② 조건식을 비교하여 참이면 문장을 실행하고, 거짓이면 for 반복문을 종료합니다.
- ③ 조건식이 참이 되어 문장을 실행하면 표현식을 실행합니다.
- ④ 종료될 때까지 ②와 ③을 반복합니다.

반복문 탈출(break, continue)

반복문을 실행하다가 더 이상 반복하지 않아도 될 상황이 발생하여
반복문을 탈출할 때에는 break문과 continue문을 사용합니다.

break

반복문 안에서 break문을 만나면 전체 반복문을 탈출합니다.

continue

continue문이 수행되는 반복문만 빠져 나갑니다. 전체
반복문이 빠져 나가는 것은 아닙니다.

연산자

연산자는 프로그래밍 언어에서 다양한 계산을 하기 위해 제공되는 기호를 말합니다

산술 연산

가장 대표적인 연산자로는 +(더하기), -(빼기), *(곱하기), /(나누기), 나머지 연산자(%)를 들 수 있습니다. 문자열 연산(+)

문자열 연산 (+)

+ 기호가 (문자와 문자, 문자와 숫자)를 연결하는데 사용되면 결과를 문자로 반환합니다.

문자와 문자 "good" + "time"의 결과는 "goodtime"입니다.
문자와 숫자 "good" + 3 의 결과는 "good3"입니다.

비교 연산

비교 연산은 표현식의 값을 비교하여 참과 거짓을 반환하며, 조건문에 많이 사용됩니다.

연산자	뜻
<	보다 작음
>	보다 큼
<=	보다 작거나 같음
>=	보다 크거나 같음
==	좌변과 우변이 같으면 참(true)
!=	좌변과 우변이 다르면 참(true)

대입 연산(=)

대입 연산은 (식별자) 변수 등에 값을 할당하는 데에 사용됩니다.

```
var x = 12
```

```
var arr = [1,2,4]
```

대입 연산자의 좌변에는 항상 1개의 변수가 있어야 합니다.

```
a + b = 3 (X)
```

한 번에 여러 개의 값을 대입하려면 아래와 같이 표현해야 합니다.

```
a = b = c = 10
```

기타 연산자에는 아래와 같은 것들이 있습니다.

- 논리 연산자 : &&, ||
- 증감 연산자 : ++, --
- 삼항 연산자 : (조건) ? 식 1 : 식 2(조건이 참일 때 식 1, 거짓일 때 식 2를 실행)

타이머 메서드

javaScript에는 일종의 타이머와 같이 지정된 시간마다 함수를 호출할 수 있는 메서드가 있습니다. 이는 윈도우 객체에서 제공해주는 전역 함수로, jQuery에서는 따로 제공하지 않습니다.

```
setTimeout("실행할 함수", 대기 시간)  
// 대기 시간이 지난 후 코드 실행(1번)  
setInterval("실행할 함수", 대기 시간)  
// 대기 시간이 지난 후 코드 반복
```

해제

```
clearTimeout( ) // setTimeout을 중지  
clearInterval( ) // setInterval을 중지
```

setTimeout

1	<code>function gallery(){</code>
2	<code> console.log("다음 사진");</code>
3	<code>}</code>
4	<code>setTimeout("gallery()",1000);</code>

setInterval

1	<code>var count = 1;</code>
2	<code>function gallery(){</code>
3	<code> console.log("다음 사진");</code>
4	<code> if(count==5){</code>
5	<code> clearInterval(clearId);</code>
6	<code> }</code>
7	<code> count = count + 1;</code>
8	<code>}</code>
9	<code>var clearId = setInterval ("gallery();",1000);</code>

DOM (Document Object Model)

“DOM은 HTML 문서의 요소를 제어하기 위해 웹 브라우저에서 처음 지원되었다. DOM은 동적으로 문서의 내용, 구조, 스타일에 접근하고 변경하는 수단이었다. 브라우저 사이에 DOM 구현이 호환되지 않음에 따라, W3C에서 DOM 표준 규격을 작성하게 되었다.”

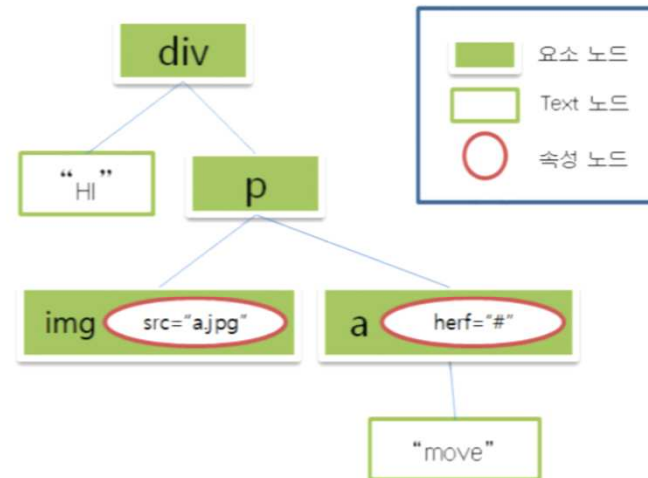
DOM 구성요소

요소 노드(element node)

텍스트 노드(text node)

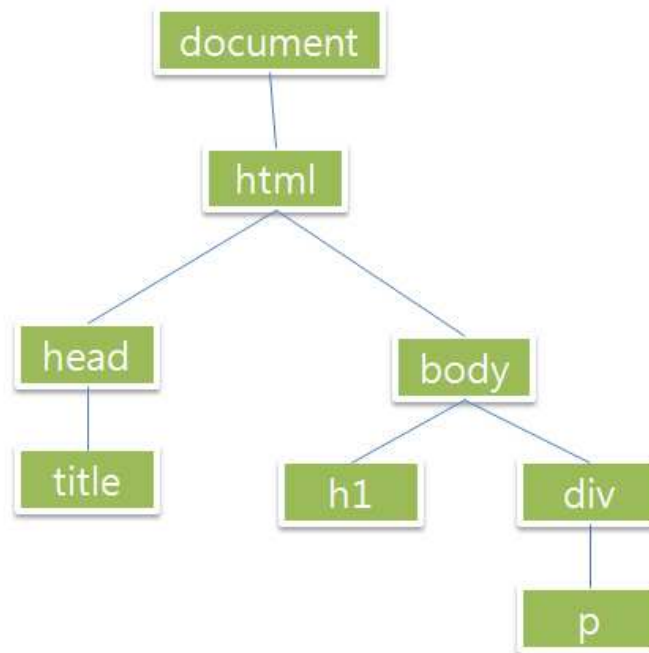
속성 노드(attribute node)

```
<div>  
  HI  
<p>  
    
  <a href="#"> move </a>  
</p>  
</div>
```

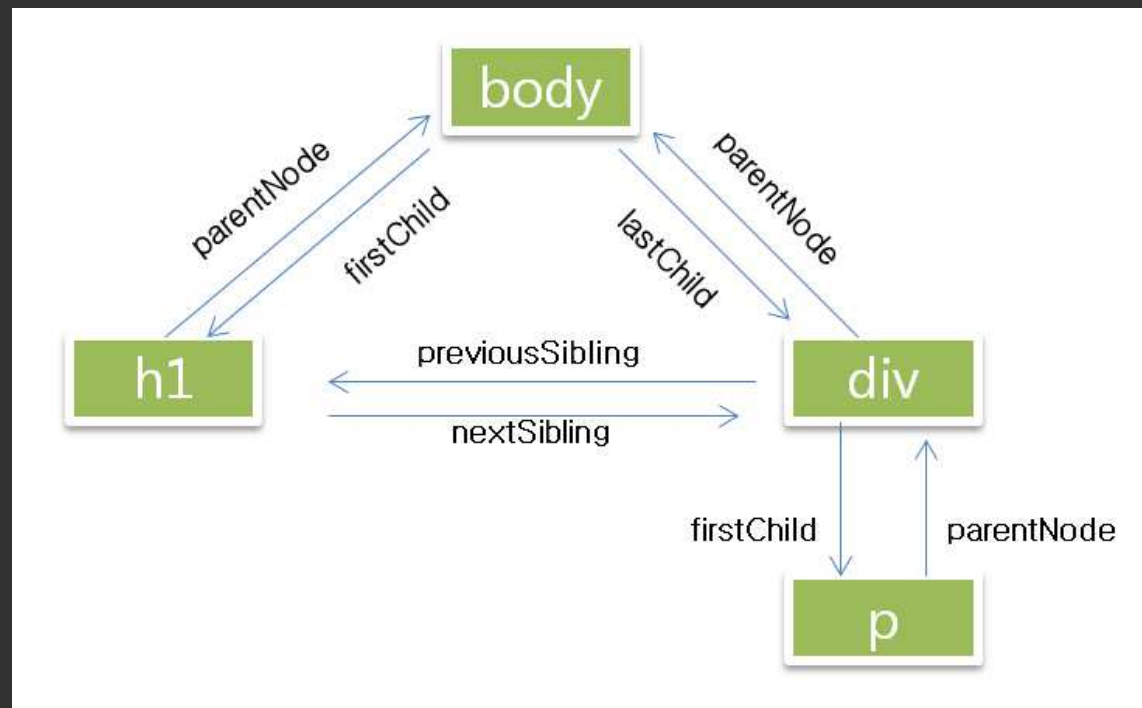


DOM Tree

```
<html>  
  <head>  
    <title> txt </title>  
  </head>  
  <body>  
    <h1> txt </h1>  
    <div>  
      <p> txt </p>  
    </div>  
  </body>  
</html>
```



계층적 관계



DOM을 제어하는 JavaScript 명령어

- `childNodes` - 지정된 노드의 자식노드들을 반환해 준다.
 - `firstChild` - 자식노드중에서 첫 번째 자식노드를 반환해 준다.
 - `nextSibling` - 같은 부모를 가진 노드들 중에 바로 다음에 나오는 노드를 반환해준다.
 - `createElement` - 동적으로 새로운 node를 생성할 수 있다
 - `getElementById` - 특정 아이디를 가진 요소에 바로 접근할 수 있다.
 - `getElementsByTagName` - 특정 태그를 가진 요소에 바로 접근할 수 있다.
 - `getAttribute` - 원하는 요소노드에 접근 후에 해당 요소의 속성을 얻을 수 있다.
 - `setAttribute` - 원하는 요소노드의 속성값을 바꿀 수 있다.
-

JavaScript 와 jQuery의 DOM접근 방식

```
<html>

  <body>

    <h2> Lorem Ipsum </h2>
    <p> This is paragraph </p>

    <div id = "main">

      <p> This is paragraph </p>

      <ul id = "navi">
        <li> item 1 </li>
        <li class="current"> item 2 </li>
        <li> item 3 </li>
      </ul>

    </div>

  </body>
</html>
```

미션1 : 문서의 모든 p 태그를 선택

미션2 : id 가 "navi" 인 태그를 선택

미션3 : div 자손 중 li 태그 class 속성값이 current 인 노드의 배경색을 빨강
색으로 변경

미션 : 문서의 모든 p 태그를 선택

JavaScript

document.getElementsByTagName("p");

jQuery

jQuery("p");

```
<html>
  <body>
    <h2> Lorem Ipsum </h2>
    <p> This is paragraph </p>

    <div id = "main">
      <p> This is paragraph </p>
      <ul id = "navi">
        <li> item 1 </li>
        <li class="current"> item 2 </li>
        <li> item 3 </li>
      </ul>
    </div>
  </body>
</html>
```

미션 : id 가 "navi" 인 태그를 선택

JavaScript

document.getElementById("navi")

jQuery

jQuery("#navi");

미션 : div 자손 중 li 태그 class 속성값이 current 인 노드의 배경색을 빨강색으로 변경

JavaScript

```
var main = document.getElementsByTagName("div");
var ulList = main[0].getElementsByTagName("ul");
var liList = ulList[0].getElementsByTagName("li");
for(var i=0;i<liList.length;i++) {
  if(liList[i].getAttribute("class") == "current") {
    liList[i].style.backgroundColor = "#F00";
    break;
  }
}
```

jQuery

\$("#div li.current").css("background-color", "#F00");

HTML/CSS



Unlimited Hosting
Free Domain included
Free instant script install
Now \$3.98

SpreadsheetGear
PERFORMANCE
SPREADSHEET COMPONENTS
DOWNLOAD FREE TRIAL

JS Basic

JS HOME

- JS Introduction
- JS How To
- JS Where To
- JS Statements
- JS Comments
- JS Variables
- JS Operators
- JS Comparisons
- JS If...Else
- JS Switch
- JS Popup Boxes
- JS Functions
- JS For Loop
- JS While Loop
- JS Break Loops
- JS For...In
- JS Events
- JS Try...Catch
- JS Throw
- JS Special Text
- JS Guidelines

JS Objects

- JS Objects Intro
- JS String
- JS Date
- JS Array
- JS Boolean
- JS Math
- JS RegExp

JS Advanced

- JS Browser
- JS Cookies
- JS Validation
- JS Timing
- JS Create Object
- JS Summary

JS Examples

- JS Examples
- JS Objects Examples
- JS Browser Examples
- JS DOM Examples

JavaScript Tutorial

« W3Schools Home

Next Chapter »



JavaScript is *THE* scripting language of the Web.

JavaScript is used in billions of Web pages to add functionality, validate forms, communicate with the server, and much more.

JavaScript is easy to learn. You will enjoy it.

Try it Yourself Examples in Each Chapter

This JavaScript tutorial contains hundreds of "Try it yourself" examples.

With our editor, you can edit JavaScript code online and click on a button to view the result.

Example

My First Web Page

This is a paragraph.

Display Date

Try it yourself »

Click on the "Try it yourself" button to see how it works

[Start learning JavaScript now!](#)

JavaScript Examples

Learn by 200 examples!

With our editor, you can edit the source code and click on a test button to view the result.

WEB HOSTING

- Best Web Hosting
- PHP MySQL Hosting
- Best Hosting Coupons
- UK Reseller Hosting
- Cloud Hosting
- Top Web Hosting
- \$3.98 Unlimited Hosting
- Premium Website Design

WEB BUILDING

- XML Editor - Free Trial!
- FREE Website BUILDER
- Build a FREE Website

W3SCHOOLS EXAMS

Get Certified in:
HTML, CSS, JavaScript,
XML, PHP, and ASP

W3SCHOOLS BOOKS

New Books:
HTML, CSS
JavaScript, and Ajax

STATISTICS

- Browser Statistics
- Browser OS
- Browser Display

SHARE THIS PAGE

Share with »



jQuery 기본 개념

Core

Selectors

Attributes/CSS

Manipulation

Traversing

Events

Effects

UI

Ajax

Core

jQuery의 핵심은 jQuery() 함수입니다. 모든 jQuery의 시작은 jQuery() 함수를 거쳐야 합니다.

이 함수를 통해 리턴되는 결과는 모두 jQuery 객체로서 jQuery의 수많은 기능들을 사용할 수 있게 됩니다.

jQuery()

더 자세한 사항은 이곳을 참고 <http://api.jquery.com/jquery/>

Selectors

jQuery 역시 문서 구조에서 원하는 HTML 노드를 선택하기 위해 Selector를 사용해야 하는데, 그 문법이 CSS와 거의 같습니다.

jQuery에서는 기본적으로 CSS, CSS2, CSS3 Selector는 물론 자체 jQuery 필터도 제공하기 때문에 원하는 HTML 노드를 쉽게 선택하여 작업할 수 있습니다.

더 자세한 사항은 이곳을 참고 <http://api.jquery.com/category/selectors/>

Attributes/CSS

Selector를 사용하여 조작을 원하는 HTML 노드를 선택했다면, 이제는 선택한 노드에 어떠한 작업을 할 순서입니다. 선택한 노드의 속성 값을 가져오거나 텍스트를 변경할 수 있고, CSS() 함수를 사용하면 CSS에 의해 적용되는 상태를 동적인 상황에서도 변경할 수 있습니다.

`jQuery('img').attr('src');` // 태그의 속성 src 값을 가져오는 것

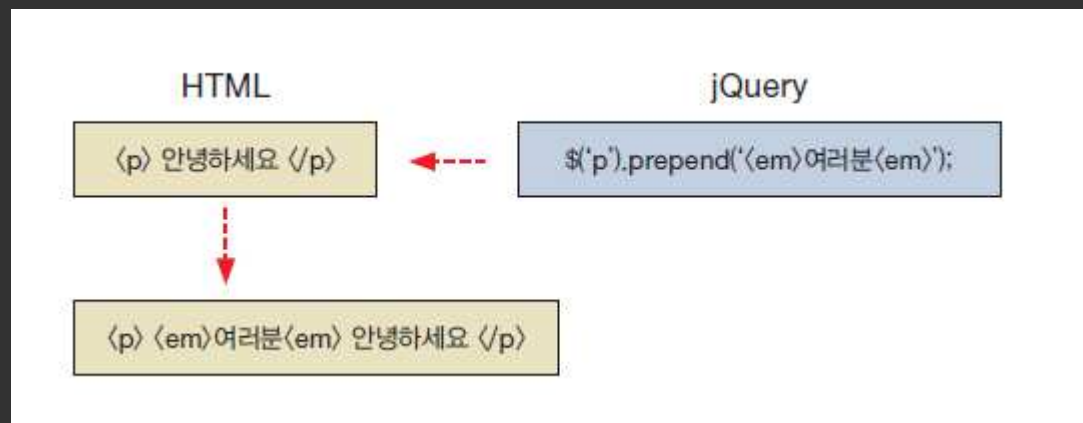
`jQuery('.main').css('border','1px #F00 solid')` // 클래스 이름이 main인 노드의 border에 1픽셀의 빨간색 외곽선 직선이 그려짐.

더 자세한 사항은 이곳을 참고 <http://api.jquery.com/category/attributes/>
더 자세한 사항은 이곳을 참고 <http://api.jquery.com/category/css/>

Manipulation

DOM에 대해 다양한 제어, 갱신 조작을 할 수 있는 명령어들입니다.
특정 노드의 앞뒤에 임의의 노드를 생성해 붙일 수 있고, 제거할 수도 있습니다.
또한 특정 노드를 교체할 수도 있습니다.

`jQuery('p').prepend('여러분');` // <p> 태그의 내용 맨 앞부분에
추가됩니다.



더 자세한 사항은 이곳을 참고 <http://api.jquery.com/category/manipulation/>

Traversing

DOM에 접근하여 원하는 노드를 찾는 방법을 제공합니다. 이 방법을 잘 활용하면 DOM에 id나 class를 최소한으로 사용할 수 있습니다. 선택한 노드를 기준으로 형제, 자식, 부모 등으로 접근하는 다양한 방법이 있습니다.

```
jQuery('li').parent( ).css('background-color', 'red');
```

parent() 메서드는 현재 지정된 li의 부모 노드를 의미합니다. 따라서 li 노드들의 부모를 선택하여 배경색을 빨간색으로 설정합니다.

더 자세한 사항은 이곳을 참고 <http://api.jquery.com/category/traversing/>

Events

프로그램은 기본적으로 위에서 아래까지 순차적으로 실행됩니다.

하지만 어떤 경우에는 사용자가 반응을 해주어야만(버튼에 클릭을 한다, 이미지 위에 마우스를 올려 놓는다 등) 작동되는 경우가 있습니다.

이때 사용자의 반응을 기다리며 그 사건이 일어날 때까지 기다리다가 그 사건이 일어나면 그때 바로 프로그램이 실행되도록 해주는 개념이 바로 '이벤트(Event)'입니다.

jQuery에는 JavaScript 에서 제공하는 이벤트에 다양한 이벤트를 더 추가하였습니다.

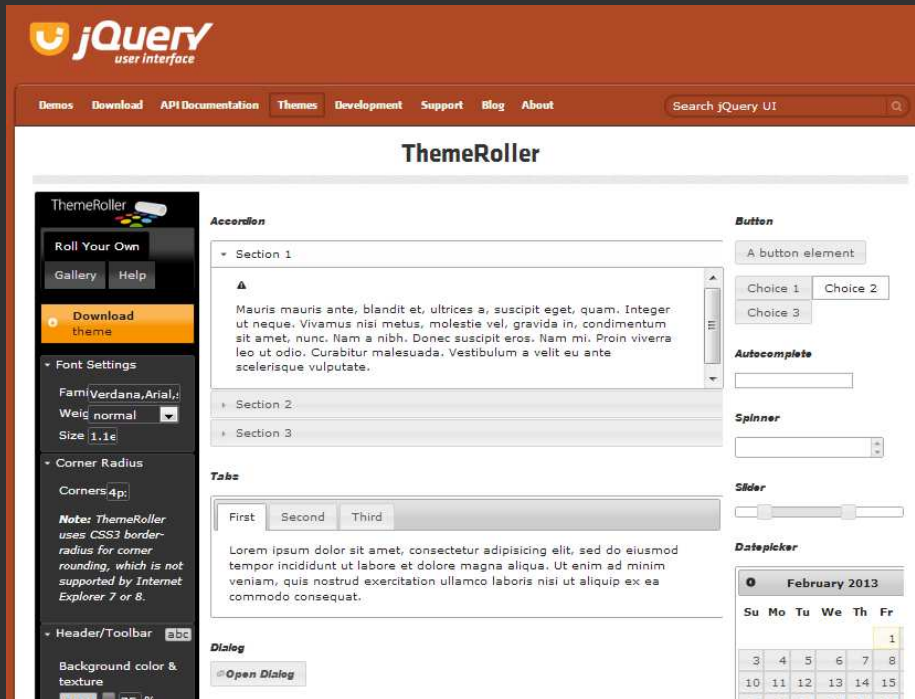
더 자세한 사항은 이곳을 참고 <http://api.jquery.com/category/events/>

Effects

웹 디자이너가 jQuery를 매력적이라고 생각하게 만든 일등공신이 바로 효과 부분이 아닐까 생각합니다. fadeIn, fadeOut, slideDown, slideUp, animate 그리고 Flash ActioScript에서 보았던 다양한 easing(가·감속 효과) 등을 사용하여 정적인 웹 페이지를 역동적인 웹 페이지로 변환시켜줄 강력한 무기입니다.

더 자세한 사항은 이곳을 참고 <http://api.jquery.com/category/effects/>

UI



jQuery는 UI(User Interface)에 대한 다양한 형태의 기능을 제공하고 있습니다. 위젯 형태의 Accordion, Datepicker, Dialog, Slider, Progressbar, Tabs와 애니메이션 확장 기능 그리고 상호작용이 가능한 요소들, 즉 Drag, Drop, Resizable 등과 같은 것을 모아놓은 라이브러리가 바로 'jQuery UI'입니다.

더 자세한 사항은 이곳을 참고 <http://jqueryui.com/>

Ajax

Ajax(Asynchronous javaScript and XML)는 서버와 비동기식으로 데이터를 교환하는 javaScript 프로그래밍 방식을 말합니다.

일반적으로 웹 페이지에서 새로운 내용이 갱신되려면 서버에 접속하여 페이지 전체를 다시 다운로드해야 합니다. 이때 Ajax를 사용하면 어느 한 부분만 서버에 접속하여 내용을 갱신할 수 있습니다.

이렇게 되면 사용자는 전체 페이지를 다시 갱신하지 않아도 되고, 필요한 내용만 변경할 수 있게 되므로 트래픽 용량도 커지지 않습니다.

이러한 이유 때문에 Ajax 기술은 주로 실시간 뉴스 전달이나 주식 시세 정보 등에 많이 사용되고 있습니다.

더 자세한 사항은 이곳을 참고 <http://api.jquery.com/category/ajax/>

한눈에 보는 jQuery 함수

<http://oscarotero.com/jquery/>

[illegible]