jQuery란?

jQuery 창시자인 존 레식(John Resig)이 2006년에 첫 번째 버전을 배포

-

-

Javascript 를 어떻게 하면 쉽게 이용할 수 있을 것인지에 고민으로부터 시작

-

-

여러 개의 자바스크립트 **라이브러리**(Javascript library) 중 하나.

(라이브러리란, 자주사용되는 기능 또는 구현하기 어려운 기능들을 하나로 묶어서 만들어 놓은 코드창고)

Ex) jQuery, Prototype, MooTools, qooxdoo, Dojo, YUI, Jindo

-

-

크로스브라우징을 가장 잘 구현하고,

Flash 처럼 화려한 기술을 용이하게 구현을 할수 있는 장점.

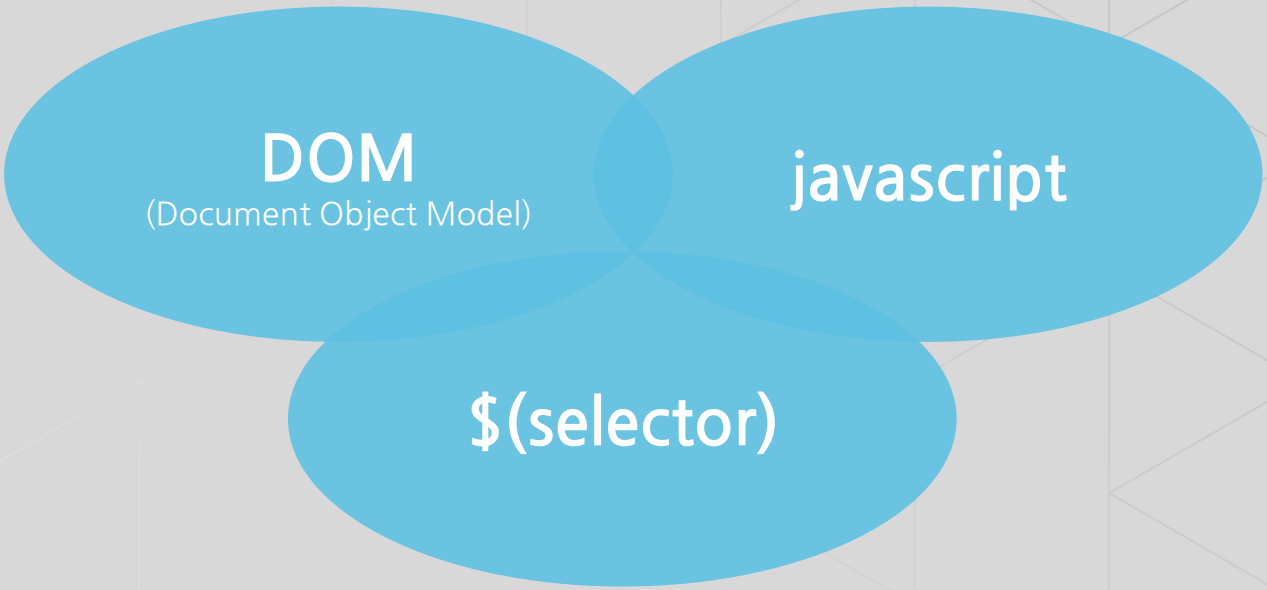
without jQuery

```
<script>
// javascript
</script>
```

with jQuery

```
<script src="jquery.js"></script>
<script>
// javascript/jQuery
</script>
```

jQuery is,



DOM (Document Object Model)

문서 객체 모델(The Document Object Model, 이하 DOM)은,
HTML 문서의 모든 요소에 접근하는 방법을 정의한 API.
즉, HTML, XML 문서의 프로그래밍 interface 이다.

DOM은 문서의 구조화된 표현(structured representation)을 제공하며 프로그래밍 언어가 DOM 구조에 접근할 수 있는 방법을 제공하여 그들이 문서 구조, 스타일, 내용 등을 변경할 수 있게 돕는다. DOM 은 구조화된 nodes와 property 와 method 를 갖고 있는 objects로 문서를 표현한다. 이들은 웹 페이지를 스크립트 또는 프로그래밍 언어들에서 사용될 수 있게 연결시켜주는 역할을 담당한다.

[참고]
https://developer.mozilla.org/ko/docs/Gecko_DOM_Reference/%EC%86%8C%EA%B0%9C
http://codingnuri.com/javascript-tutorial/html-dom-document-object-model-overview.html

DOM Tree 안의 노드들은 서로 계층적인 관계를 가지며,
parentNode, firstChild, lastChild, sibling 등의 속성으로 상호간에 접근을 한다.

(예)

childNodes - 지정된 노드의 자식 노드들을 반환
firstChild - 자식 노드 중에서 첫 번째 자식 노드를 반환
nextSibling - 같은 부모를 가진 노드들 중에 바로 다음에 나오는 노드를 반환
creatElement - 새로운 노드를 동적으로 생성
getElementById - 특정 아이디를 가진 요소에 바로 접근
getElementByTagName - 특정 태그를 가진 요소에 바로 접근
getAttribute - 원하는 요소 노드에 접근한 후 해당 요소의 속성을 얻을 수 있음
setAttribute - 원하는 요소 노드의 속성 값을 바꿀수 있음

[참고]
https://www.w3schools.com/jsref/dom_obj_all.asp

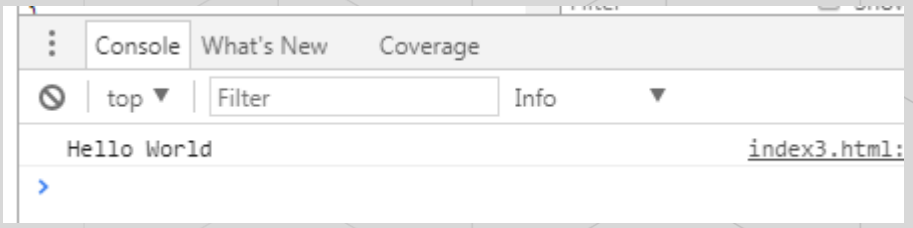
javascript

1. console.log
2. 주석(comment)
3. 변수
4. 예약어
5. 데이터 타입
6. 프로토타입
7. 연산자

console.log

- 변수 및 다양한 정보들을 확인할 수 있는 매우 중요한 함수 (개발자 도구[F12]를 통해 확인 가능)
- IE(Internet Explorer)는 8.x 이상에서만 지원되고, 이외의 웹브라우저는 대부분 지원
- 프로그램에 영향을 주지 않음
- 디버깅 용도로 사용 (*디버깅 : 버그를 해결하다라는 의미)

```
<script>
console.log('Hello World');
</script>
```



주석 (comment)

- 프로그램 작성시 코드를 설명하거나 기타 개발자의 참조사항을 기록하기 위해 사용
- 한 줄 코딩의 주석은 '//'
- 여러 줄을 한번에 주석처리 하기 위해서는 '/*주석*/'
- 프로그램에 영향을 주지 않음

```
<script>
// 한줄 주석
/*
        여러줄 주석
        여러줄 주석
*/
</script>
```

변수

- 어떤 값을 지속적으로 저장하여 필요할 때마다 사용할 수 있게 하는 저장소
- 변수명에 포함된 값들은 언제든지 변경할 수 있음
- Javascript 내부적으로 미리 사용하는 키워드나 **예약어**는 사용안됨
- 변수앞에 숫자 또는 특수기호/문자는 제외. 단, _(underscore)는 가능

```
<script>
// 변수 a에 10을 저장합니다.
var a = 10;


// 변수 name에 문자열 '홍길동'을 저장합니다.
var name = '홍길동';
</script>
```

변수

변수의 범위

- 변수의 범위를 명확히 하려면 var 를 선언하는 것이 좋다
- 특정 함수 안에서 var 를 통해 선언된 변수는 지역 변수로 인식되지만,
  var 를 사용하지 않는 변수는 전역 변수로 인식한다

```
<script>
var myAge = 10;
function myFunc () {
        var temp = 10;
        total = temp + 20;
};
function myFunc2 () {
        var temp = 20;
};
</script>
```

myAge 변수는 <script> 공간에 선언되었기 때문에 전역 변수로서 전체 영역에 영향을 미침

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

temp 변수는 function() 내부에 변수가 선언되었기 때문에 지역변수로써 함수 내부에만 영향을 미침

예약어

<script>
abstract, boolean, break, byte, case, catch, char, class, const, continue, default, do, double, else, extends, final, finally, float, for, function, goto, if, implements, import, in, instanceof, int, interface, long, native, new, null, package, private, protected, public, return, short, static, super, switch, synchronized, this, throw, transient, try, var, void, while, with, false, true
</script>

데이터 타입 (https://www.w3schools.com/js/js_datatypes.asp)


```
<script>
var intNum = 34;
var singleStr = 'Volvo XC60';
var boolVar = true;
var emptyVar;
var nullVar = null;
</script>
```

```
<script>
var objType = {};
var arrType = [];
var funcType = function () {};
var patt = /w3schools/i;
</script>
```

데이터 타입

기본 타입

- 자바스크립트에서 기본 타입은 **숫자, 문자열, 불린값**을 비롯해 **null**과 **undefined**라는 타입이 있다. 이들 타입의 특징은 그 자체가 하나의 값을 나타낸다는 것이다.

```
<script>
var intNum = 34;
var singleStr = 'Volvo XC60';
var boolVar = true;
var emptyVar;
var nullVar = null;

console.log(typeof intNum, typeof singleStr, typeof boolVar, typeof emptyVar, typeof
nullVar);
</script>
```

데이터 타입

참조 타입

- 자바스크립트에서 숫자, 문자열, 불린값, null, undefined 같은 기본 타입을 제외한 모든 값은 객체다. 따라서 배열, 함수, 정규표현식 등도 모두 자바스크립트 객체로 표현된다.
- 자바스크립트에서 객체는 단순히 '이름(key) : 값(value)' 형태의 프로퍼티들을 저장하는 컨테이너이다.
- 자바스크립트에서 기본타입은 하나의 값만을 가지는 데 비해, 참조 타입인 객체는 여러 개의 프로퍼티들을 포함할 수 있으며, 이러한 객체의 프로퍼티는 기본 타입의 값을 포함하거나, 다른 객체를 가리킬 수도 있다.
- 이러한 프로퍼티의 성질에 따라 객체의 프로퍼티는 함수로 포함할 수 있으며, 자바스크립트에서는 이러한 프로퍼티를 메서드라고 부른다.

데이터 타입

참조 타입의 특성

- 자바스크립트에서는 기본 타입인 숫자, 문자열, 불린값, null, undefined 5가지를 제외한 모든 값은 객체다. 배열이나 함수 또한 객체로 취급된다. 그리고 이러한 객체는 자바스크립트에서 참조타입이라고 부른다. 이유는 객체의 모든 연산이 실제 값이 아닌 참조값으로 처리되기 때문이다.

```
<script>
var objA = {
            val : 40
};
var objB = objA;

objB.val = 50;
console.log(objA.val); // (출력값) 50
console.log(objB.val); // (출력값) 50
</script>
```

프로토타입

- 자바스크립트의 **모든 객체는 자신의 부모 역할을 하는 객체와 연결되어 있다.** 그리고 이것은 마치 객체지향의 상속 개념과 같이 부모 객체의 프로퍼티를 자신의 것처럼 쓸 수 있는 특징이 있다.
- 자바스크립트에서는 이러한 부모 객체를 **프로토타입 객체**(짧게는 **프로토타입**)라고 부른다.
- 자신의 부모 역할을 하는 프로토타입 객체의 프로퍼티를 차례대로 검색하는 것을 **프로토타입 체이닝**이라고 하며, **체이닝의 종점**은 Object.prototype 객체이다.

```
<script>
var foo = {
            name : 'foo',
            age : 40
};
console.log(foo.toString());
console.dir(foo);
console.log(Object.prototype);
</script>
```

프로토타입

- 자바스크립트의 모든 객체가 프로토타입 체이닝으로 Object.ptorotype 에 정의한 메서드를 사용 가능하다. 즉, Object.prototype 에 정의된 메서드들은 자바스크립트의 모든 객체의 표준 메서드라고 볼 수 있다.
- 이와 같은 방식으로 자바스크립트의 숫자, 문자열, 배열 등에서 사용되는 표준 메서드들의 경우, 이들의 프로토타입인 **Number.prototype**, **String.prototype**, **Array.prototype** 등에 정의되어 있다. 이러한 기본 내장 프로토타입 객체 또한 **Object.prototype** 을 자신의 프로토타입으로 가지고 있어서 프로토타입 체이닝으로 연결된다.

연산자

다양한 계산을 하기 위해 제공되는 기호

- 산술 연산
- 문자열 연산(+)
- 비교 연산
- 대입 연산(=)
- 기타 연산자

연산자

산술 연산

- + , - , * , / , %

```
<script>
1 + 2;
1 - 2;
1 * 2;
1 / 2;
1 % 2;
</script>
```

연산자

문자열 연산(+)

- +연산자의 피연산자 2개(문자와 문자, 문자와 숫자)를 연결하고 결과를 문자로 반환

```
<script>
'good' + 'time';
'good' + 3;
</script>
```

연산자

비교 연산

- 값을 비교하여 참과 거짓을 반환하며, 조건문에 많이 사용됨

```
<script>
10 < 9  // 보다 작음
10 > 9  // 보다 큼
10 <= 9  // 보다 작거나 같음
10 >= 9  // 보다 크거나 같음
10 == 9  // 좌변과 우변이 같으면 참(true)
10 != 9  // 좌변과 우변이 다르면 참(true)
</script>
```

연산자

대입 연산(=)

- 변수 등에 값을 할당하는 데에 사용

```
<script>
var x = 12;
</script>
```

- 좌변에는 항상 1개의 변수가 있어야 함

```
<script>
var a + b = 3; // (X)
</script>
```

- 한번에 여러 개의 값을 대입하려면 아래와 같이 표현

```
<script>
var a = b = c = 12;
</script>
```

연산자

대입 연산(=)

- **복합대입연산** : 변수의 값을 상대적으로 처리할 때에 많이 사용

  예) a = a + 5 를 a += 5 로 줄여 쓸수 있음

```
<script>
= // 대입
*= // 곱하기 대입
/= // 나누기 대입
%= // 나머지 대입
+= // 더하기 대입
-= // 빼기 대입
</script>
```

연산자

기타 연산자

- 논리 연산자 : && , ||
- 증감 연산자 : ++, --
- 삼항 연산자 : (조건) ? 식1 : 식2 (조건이 참일 때 식1, 거짓일 때 식 2를 실행)

```
<script>
// 논리 연산자
10 < 9 && 9 < 10
// 증감 연산자
var time = 0;
time++;
// 삼항 연산자
var varia = (3 < 9) ? true : false;
</script>
```

$(selector)

주요 선택자 비교

javascript	jQuery
document.getElementById('id');	$('#id');
document.getElementsByClassName('class');	$('.class');
document.getElementsByName('names');	$('[name="names"]');
document.getElementsByTagName('tagname');	$('tagname');

[참고]

https://oscarotero.com/jquery/

- SELECTORS

주요 탐색자 비교

javascript	jQuery
.parentNode	.parent() .parents() .parentsUntil()
.childNodes / .children	.children() .find()
.nextSibling / .previousSibling .nextSibling .previousSibling	.siblings() .next() .nextAll() .nextUntil() .prev() .prevAll() .prevUntil()
.firstChild	.first()
.lastChild	.last()
	.eq() .filter() .not()

[참고]

https://oscarotero.com/jquery/

- TRAVERSING

Thank you.

HIVELAB

Question.