Chapter 1

자바스크립트 기본

개요

자바 스크립트

프로토타입 기반 객체지향 언어이다.
자바스크립트는 스크립트 언어의 특성 상 플랫폼에 독립적이며 모든 웹 브라우저에서 동일한 실행 결과를 얻을 수 있다.
무료이며, 쉽고 자유롭게 사용할 수 있다.
프로그램 코드가 직접 html문서에 삽입되어, 브라우저에서 html파일을 읽을 때 같이 해석되고 실행된다.
클라이언트에서만 실행되기에 정보를 서버에 보낼 필요 없이 처리할 수 있다.
타입 체크가 철저하지 않습니다. 즉, 변수들의 타입에 있어서 차이를 두지 않는다.
자바스크립트 기반 프론트엔트 기술의 발전으로 근래, 순수 스크립트의 사용을 선호하게 되며
JAVA보다 확장성이 뛰어난 언어이다.
자바스크립트 개발자가 있을 만큼 사용방법이 다양하며, 반드시 익혀두어야 합니다.
JAVA를 확실히 해두었다면, 자바스크립트는 3일 이면 충분하다

스크립트 언어란?

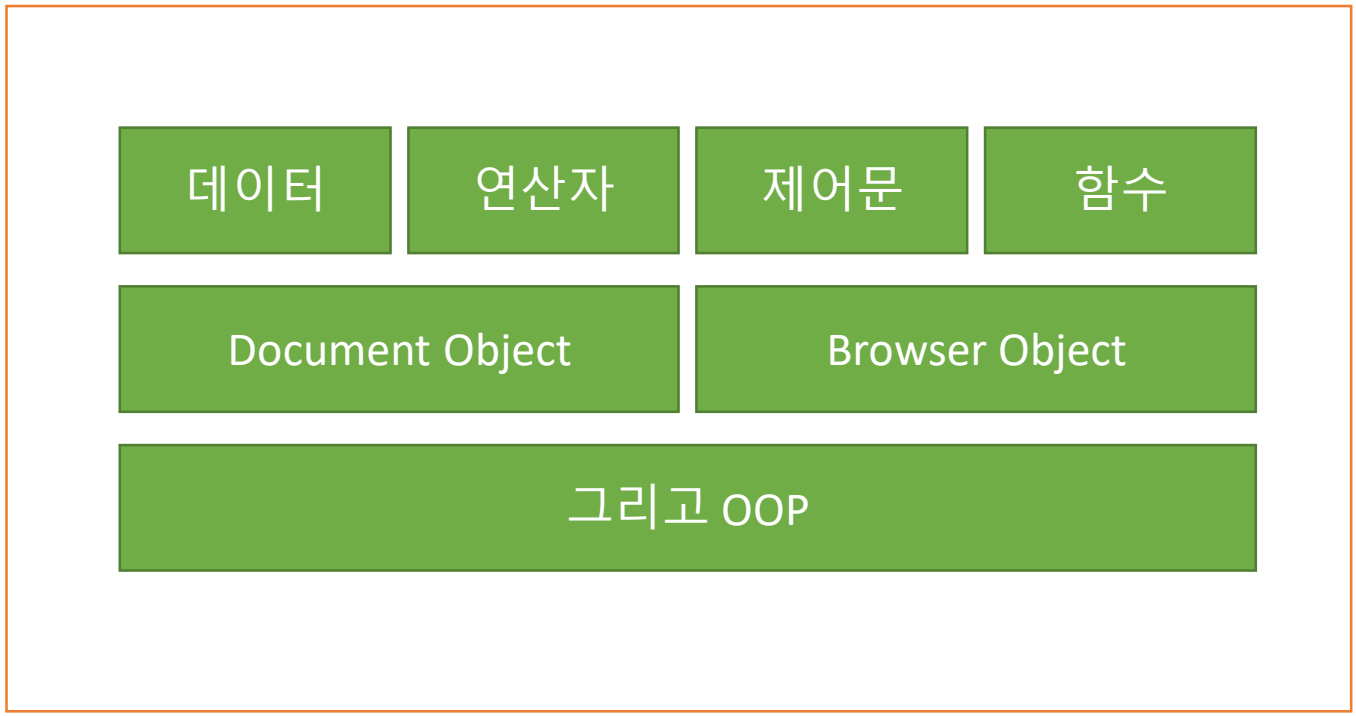
어플리케이션이 실행되는 동안 라인 단위로 해석(인터프리터)되어 실행되는 언어.
별도의 컴파일 과정이 없다.
HTML 문서 내에서 스크립트 언어는 <script>와 </script>사이에 작성한다.

자바스크립트는 웹 개발자에 아주 중요하다고 생각하며
반드시 복습 하시길 바랍니다

참고 사이트: w3schools.com

자바스크립트 범주

자바스크립트의 기본 구조



자바스크립트 기본 구조

자바스크립트의 기본 구조

<script> 태그를 사용하여 HTML 문서에 코드를 작성한다.

type 속성값으로"text/javascript"를 입력(HTML5 에서는 생략 가능)

```
1.  <!DOCTYPE html>
2.  <html>
3.  <head>
4.  <meta charset="UTF-8" />
5.  <title>javascript-base.html</title>
6.  </head>
7.  <body>
8.    <hr>
9.    <h1>자바스크립트 기본 구조</h1>
10.   <hr>
11.   <script type="text/javascript">
12.     document.write("<h2>안녕하세요!</h2>");
13.   </script>
14. </body>
15. </html>
```

자바스크립트 사용방법

자바스크립트를 외부 파일로 사용되는 경우 (CSS의 외부스타일 시트방법과 동일)

<body> 태그 부분이든 기능을 구현할 곳에 삽입
외부 파일 자바스크립트의 확장자는 "js"

script element의 src 속성값이 호출할 자바스크립트의 파일명

사용 방법

```
<script type="text/javascript" src="호출할 자바스크립트 파일명.js"></script>
```

자바스크립트 파일 안에는 <script> 태그를 포함할 수 없습니다.

변수

자바스크립트 구문

문장 끝에 세미콜론(;)을 입력한다.

구문 사이를 구분하기 위해 사용. 생략해도 무관하나 오류방지를 위해 권장한다.

자바스크립트 주석

// : 한 줄 주석 처리. 프로그램 코드 끝에 사용

/* ~ */ : 한 줄 이상의 주석 처리

변수 선언

var 키워드 이용

자바스크립트에서 변수의 스코프(scope)는 코드 블록({ })이 아닌 함수 단위

let키워드 이용(ES6문법에 추가된 변수)

var는 동일 변수의 중복을 허용하지만 let은 중복 변수의 선언을 허용하지 않습니다

```
var num = 10;
var num = 20;
```
가능

```
let num = 10;
let num = 20;
```
에러

출력

분류	명령어
출력	document.write("브라우 출력");
	alert("경고창 출력");
	console.log("콘솔창 출력");
확인	confirm("확인창출력");

자바 스크립트는 문법이 틀리면 화면상에 표시되지 않습니다.

개발자 도구 F12에서 확인해야 합니다

콘솔의 결과도 개발자 도구에서 확인합니다

데이터 타입

기본타입

숫자(number)

정수, 실수 구분 없음

문자열(string)

문자, 문자열 구분 없음
쌍따옴표("") 로 묶어도 되고, 홑따옴표('')로 묶어도 됨

불린(boolean) 타입

true/false
비교 결과가 0, null, ""(빈 문자열), false, undefined, NaN으로 판단되면 false, 나머진 true
동등 비교를 할 경우 타입도 같이 비교해 주는 === 사용을 권장

undefined

변수는 선언했으나 초기화를 하지 않은 경우 자동 저장됩니다.
불린형은 false로 반환, 숫자는 NaN, 문자열은 "undefined"로 반환됨

null

값은 저장했으나 존재하지 않을 때 나타납니다

참조타입

배열

[ ]

객체

{}로 생성

연산자

산술연산자

+ - * % ++ --

비교연산자

< > <= >= == != === !==

대입연산자

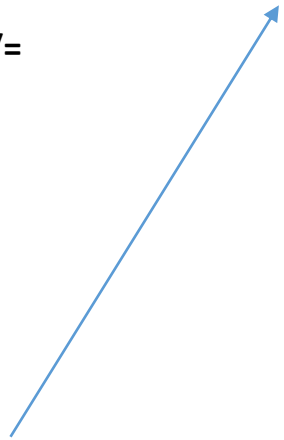
= += -= *= %= /=

논리연산자

! && || & |

삼항연산자

조건식 ? 실행문1 : 실행문2



기본적으로 자바와 동일하다
단 === !== 는 타입도 비교하는 연산자 이다

배열

자바 스크립트의 배열은 [ ] 로 묶어 주기만 하면 됩니다.
자바 스크립트의 배열은 저장은 자바와 다르게 타입과 상관없습니다

배열의 선언, 생성

```
var arr1 = [];
var arr2 = [1, 2, 3];
var arr3 = [1, 2, 3 , "가", "나"];

var arr4 = new Array();
```

전부 가능

배열의 사용

```
var arr1 = [];
arr1[0] = 1;
arr1[1] = 2;

document.write(arr1[0] ) ;
document.write(arr1[1] ) ;
```

결과
1
2

배열의 길이, 문자열로 출력

```
var arr2 = [1, 2, "가"];

document.write(arr2 );
document.write(arr2.length );
document.write(arr2.toString() );
```

결과
1, 2, 가
3
1, 2, 가

조건문

조건문	if else, switch
반복문	while, do while, for, for in
탈출문	break, continue

조건문

if() 문

```
if(조건식) {

}else {

}
```

```
switch(값) {
   case 값 :
      //실행문;
      break;
   case 값 :
      //실행문;
      break;
   default :
      //실행문;
}
```

자바와 동일

반복문 while

반복실행 할 수 있는 문장으로 while, do/while, for, for/in 문이 있다.

while문

```
var i=1;
while(i<=10) {

    i++;

}
```

자바와 동일, 단 i변수는 전역 변수가 된다

반복문

for, for/in 문

for 문
사용법
for(var i = 1; i <= 10; i++) {

}

for/in 문
사용법
for(변수 in 객체) {

}

자바의 향상된for문과 조금 다르다

```
var myObj = { p1:'a', p2:'b'}; //객체 정의
var result = '';
for(var prop in myObj) {
    result += '속성명:'  + prop + ', 값:' + myObj[prop] + '\n';
}
document.write(result + "<br>");
```

i 요소에 변수를 담는다(주의)

결과
속성명:p1, 값:a 속성명:p2, 값:b

```
var arr = ['가', '나', '다'];
var result = '';
for(var i in arr) {
    result += '속성명:' + i + ', 값:' + arr[i] + '\n';
}
document.write(result + "<br>");
```

i 요소에 인덱스 번호를 담는다(주의)

결과
속성명:0, 값:가 속성명:1, 값:나 속성명:2, 값:다

함수(선언적 함수)

자바스크립트에서 함수는 1급 최상위 함수입니다.
객체지향 프로그래밍 언어에서 메서드의 기능보다 훨씬 더 많은 기능을 한다.

자바스크립트 사용자 정의 함수

분류	종류	의미
사용자 정의 함수	-선언적함수 -익명함수	사용자가 정의하는 함수

```
function name(매개변수) {

    //return true;
}
```

선언적 함수
-반환 유형은 적지 않으며 매개변수도 적지 않을 수 있습니다.
-매개 변수는 데이터 타입을 적지 않습니다.
-리턴은 적어도 되며 적지 않아도 상관없습니다.

```
compute();
function compute() {
    var x = 100;
    var y = 10;
    var result = x / y;
    console.log(result);
}
//compute();
```

```
compute(100, 10);
function compute(a, b) {

    var result = a / b;
    console.log(result);
}
//compute(100, 10);
```

선언적 함수는 함수 선언
이전에 호출 하여도
에러를 발생시키지 않습니다

함수(선언적 함수)

```
function method3() {
    alert("method3실행");
}

var a = method3;
a();
```

선언적 함수
-함수를 변수에 저장할 수도 있다

함수를 a에 저장할 때 변수 선언 이후에 호출해야합니다

함수(익명함수)

익명 함수(중요)
-익명 함수는 변수에 함수 데이터를 저장하여 변수를 마치 함수처럼 사용 하도록 만들어 줍니다

```
var compute = function() {

}
```

```
//compute(); //에러

var compute = function() {
    console.log('익명함수');
}
compute();
```

익명함수는 변수 선언 이후에 호출해야 합니다

매개 변수와 리턴

매개 변수
- 함수가 필요한 값을 전달하는 매개체 입니다.
- 아무것도 적지 않을 수 있고, 여러 개 받을 수도 있으며 ,로 연결 하면 됩니다.
- 자바스크립트의 매개 변수는 함수를 호출하는데 영향을 끼치지 않습니다.

리턴
- 메서드가 실행 결과를 돌려주는 반환 값 입니다.
- 자바스크립트의 return뒤에 함수 실행 후 돌려줄 값을 적습니다.
- return false를 만나면 함수는 강제종료 됩니다.

```
function method(a, b) {
    return a + b;
}
```

함수의 가변 인자 argument

자바스크립트에서 매개변수는 큰 의미가 없다, 단순히 인자 값에 들어오는 것에 이름을 붙이는 형태이다

```
method(1,2,3,4,5,6,7);

function method(a, b) {
   return a + b;
}
```

다음 코드는 에러를 발생시키지 않는다.

함수(화살표 함수)

화살표 함수
-ES6에 추가된 함수 내용으로 => 를 이용하는 함수 입니다

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

```
var compute = (a, b) => a + b;
```

```
var compute = function(a, b) {

    var avg = (a + b) / 2;
    return avg;
}
```

```
var compute = (a, b) => {

    var avg = (a + b) / 2;
    return avg;
}
```

단일 명령문일 경우 { } 를 생략할 수 있습니다

우리 수업에서는 많이 사용하지 않겠지만
앞으로 스크립트 사용에 많은 변화를 줄 함수이기 때문에
알아 둡시다.

함수(즉시실행 함수)

즉시실행함수

```
(function() {

})();
```

즉시실행 함수
함수 표현(Function expression)은 함수를 정의하고, OR
변수에 함수를 저장하고 실행하는 과정을 거칩니다.

하지만 즉시 실행 함수는 함수를 정의하고 바로 실행하여 이러한 과정을 거치지 않는 특징이 있습니다.
함수를 정의하자마자 바로 호출하는 것을 즉시 실행함수라고 합니다.

사용이유라면?
페이지 시작 시 호출할 함수를 기술합니다

전역변수(global) 지역변수(local)

변수 범위	변수 선언	특징
함수 레벨 스코프	var	변수의 중복선언 가능 유일하게 함수 블록{ } 내부에서 지역 변수가 존재
블록 레벨 스코프	let	같은 블록{ } 에서는 이미 선언한 변수를 중복 선언 불가 블록 { } , 제어문 블록 { } 에서도 지역변수가 존재

```
var num1 = 50;
if(num1 === 50) {
   var num1 = 60;
}
alert("num1은?" + num1);

var num2 = 50;
function add() {
   var num2 = 60;
   var num2 = 70; //가능
}
add();
alert("num2는?" + num2);
```

60

50

```
let num3 = 100;
if(num1 === 100) {
   let num3 = 200;
}
alert("num3는?" + num3);

let num4 = 100;
function add2() {
   let num4 = 200;
   //let num4 = 300; //불가
}
add2();
alert("num4는?" + num4);
```

100

100

객체 (JSON)

자바스크립트의 객체는 { }로 표기합니다
자바스크립트의 객체는 함수로도 표기합니다

```
class person {
   String name = '홍길자';
   int age = 20;
   int[] arr = {1,2,3};

}  자바의 클래스
```

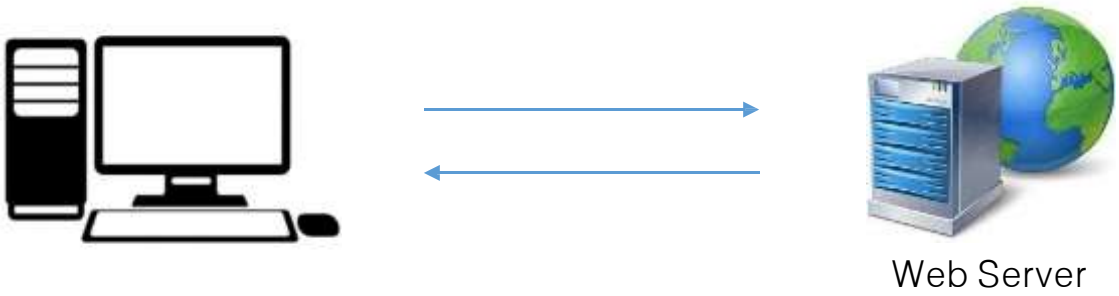
```
var person = {
   name : '홍길자' ,
   age : 20,
   arr : [1,2,3]
}
```

자바스크립트의 객체


```
function  People(name, age) {
        this.name = name;
        this.age =age;
        this.info = function() {
           return "이름:" + name + ", 나이:" + age;
        }
}
var people = new People('홍길순', 50);
console.log(people.name);
console.log(people.age);
console.log(people.info()  );
```

생성자 함수로 객체 선언 가능
-참고만 해두세요

객체 (JSON)



웹 브라우저(스크립트)

네트워크야. 너는 JSON이 뭔지 아니?
아니. 나는 몰라 오로지 문자열 이라는 데이터만 아는 걸??
그래 그럼 내가 **JSON형식으로 문자열로 바꾸어서 줄게!**

```
var  data = [
        {id:1, title: "hi", content: "뻴!"},
        {id:2, title: "bye", content: "헑!"},
        {id:3, title: "hello", content: "힣!"},
     ]
```

제이슨을 -> 문자열로
JSON.stringify(data);

문자열을 -> 제이슨으로
단, 이경우는 엄격한 형식을 지킨다(키, 값이 전부 " "로 묶여야 한다)
JSON.parse(문자열);

Chapter 1

수고하셨습니다