Javascript

1. 변수선언 : RAM 사용하는 문법
2. 데이터타입 : RAM 효율적으로 사용하는 문법
3. 연산자 : CPU 사용하는 문법
4. 조건문 : 코드를 효율적으로 사용하는 문법 : 조건에 따라 다른 코드 실행
5. 반복문 : 코드를 효율적으로 사용하는 문법 : 특정 코드를 반복적으로 실행
6. 함수 : 코드를 효율적으로 사용하는 문법 : 중복 코드를 묶어서 실행
7. 객체 : 식별자 1개에 여러개의 데이터를 저장하는 문법 : 클래스, 데이터 타입 문법

개요

- 웹브라우져에서 동작되는 동적으로 웹페이지를 바꾸고 이벤트를 처리하는 언어
- 인터프리터, 동적타이핑, 객체지향
- ECMAScript
  - ECMA5, ECMA6, ECMA8 버전이 있음
  - http://www.ecma-international.org/publications/standards/Ecma-262.htm (http://www.ecma-international.org/publications/standards/Ecma-262.htm)

1. variable : 변수선언

- 식별자 : 저장공간을 구분하는 문자열
- 식별자 규칙
  - 대소문자, 숫자, _ , $ 사용가능
  - 숫자 가장 앞에 사용 X
  - 예약어 사용 X
- 식별자 컨벤션
  - 상수 : UPPER_SNAKE_CASE
  - 변수 : camelCase
  - 함수 : camelCase
  - 모듈 : PascalCase

In [1]:

```
// 식별자 1개, 데이터 1개
var data1 = 10;
var data2 = 'js';

// 식별자 n개, 데이터 n개
var data3 = 20, data4 = 'node';
var data3 = 20,
    data4 = 'node';

// 식별자 n개, 데이터 1개
var data5 = data6 = 'web';
```

In [2]:

```
// console.log() : 식별자 데이터 출력
console.log(data1, data2, data3, data4, data5, data6);
```

```
10 'js' 20 'node' 'web' 'web'
```

2. datatype : 데이터 타입

- number : 숫자 데이터 타입(정수, 실수)
- string : 문자열 데이터 타입
- boolean : 논리값 데이터 타입
- function : 코드 저장하는 함수 데이터 타입
- object : 객체를 저장하는 데이터 타입

In [3]:

```
var data1 = 1;
var data2 = 'javascript';
var data3 = true;
var data4 = function(){ console.log('function'); };
var data5 = {key: 'value'};
```

In [4]:

```
// typeof : 함수가 아닌 명령어
// 동적타이핑 : 데이터에 따라서 자동으로 데이터 타입 설정
console.log(typeof data1, typeof data2, typeof data3)
console.log(typeof data4, typeof data5)
```

```
number string boolean
function object
```

없는 데이터의 표현

- undefined : 선언은 되었으나 값이 할당되지 않음
- null : 선언이 되어 값이 없음이 할당됨
- NaN : Number 데이터 타입에서의 undefined, 0/0

In [5]:

```
var data = undefined;
console.log(typeof data, data);
```

```
undefined undefined
```

In [6]:

```
var data = null;
console.log(typeof data, data);
```

```
object null
```

In [7]:

```
var data = NaN;
console.log(typeof data, data);
```

```
number NaN
```

In [8]:

```
// NaN 데이터 타입은 비교연산하면 항상 NaN 출력
console.log(NaN == NaN, NaN > NaN);
```

```
false false
```

데이터 타입의 형변환

- Number() : 숫자 데이터 타입으로 변환
- String() : 문자열 데이터 타입으로 변환
- Boolean() : 논리값 데이터 타입으로 변환

In [9]:

```
var data1 = '1';
var data2 = 2;
console.log(typeof data1, data1);
console.log(typeof Number(data1), typeof Boolean(data1));
```

```
string 1
number boolean
```

In [10]:

```
console.log(data1 + data2, Number(data1) + data2, data1 + String(data2));
```

```
12 3 12
```

묵시적 형변환

- 여러 데이터 타입을 혼합해서 연산하면 묵시적 형변환이 발생함(파이썬은 에러발생)

In [11]:

```
console.log(typeof data1, typeof data2);
console.log(typeof (data1 + data2));
```

```
string number
string
```

In [12]:

```
// 문자 > 숫자
console.log(typeof data1, typeof (data1 - 0));
```

```
string number
```

In [13]:

```
// 숫자 > 문자
console.log(typeof data2, typeof ('' + data2));
```

```
number string
```

3. operator : 연산자

- 산술 : 데이터 + 데이터 = 데이터 : +, -, *, /, %, **, ++, --
- 비교 : 데이터 + 데이터 = 논리값 : ==, ===, !=, !==, <, >, <=, >= : 조건 1개
- 논리 : 논리값 + 논리값 = 논리값 : !, &&, || : 조건 2개 이상
- 할당 : 변수 산술= 데이터 : 변수에 누적해서 연산
- 삼항 : (조건) ? (참) : (거짓) : 간단한 조건문 구현

In [14]:

```
// 산술연산자
var data1 = 11, data2 = 4;
console.log(data1 / data2, data1 % data2, data2 ** 3);
```

```
2.75 3 64
```

In [15]:

```
// ++data : +1하고 대입
data1 = ++data2;
console.log(data1, data2);
// data++ : 대입하고 +1
data1 = data2++;
console.log(data1, data2)
```

```
5 5
5 6
```

In [16]:

```
// 비교연산자
// ===, !==: 데이터 타입까지 비교
// ==, != : 데이터만 비교
var data1 = 1, data2 = '1';
console.log(data1, data2)
console.log(data1 === data2, data1 == data2, data1 != data2, data1 !== data2)
```

```
1 '1'
false true false true
```

In [17]:

```
// 논리연산자
console.log(true && false, true || false)
```

```
false true
```

In [18]:

```
// Quiz
// 조건 1개 : 예금 잔고에서 인출이 가능하면 true, 불가능 하면 false 출력
var balance = 10000;
var withdraw = 6000;
console.log(balance >= withdraw);
```

```
true
```

In [19]:

```
// 조건 2개 : 최대 출금 금액 5000원일때,
console.log(balance >= withdraw, withdraw <= 5000);
console.log((balance >= withdraw) && (withdraw <= 5000));
```

```
true false
false
```

In [20]:

```
// 할당연산자
var data1 = 10;
// data1 = data1 + 5;
data1 += + 5;
console.log(data1);
```

```
15
```

In [21]:

```
// 삼항연산자 : (조건)?(참):(거짓)
var balance = 10000, withdraw = 6000;
var msg = balance >= withdraw ? '인출가능' : '인출불가';
console.log(msg)
```

```
인출가능
```

4. condition : 조건문

- if, else if, else
- switch, case, default

In [22]:

```
// 학점출력 : if, else if, else
var point = 87;
if(point >= 90){
    console.log('A');
} else if(point >= 80){
    console.log('B');
} else if(point >= 70){
    console.log('C');
} else if(point >= 60){
    console.log('D');
} else{
    console.log('F');
}
```

B

In [23]:

```
// 승점출력 : switch, case, default
var point = '승';
switch(point){
    case '승':
        console.log('승점', 3);
        break;
    case '무':
        console.log('승점', 1);
        break;
    default:
        console.log('승점', 0);
}
```

승점 3

5. loop : 반복문

- while, for, break, continue

In [24]:

```
var count = 3;
while (count > 0){
    count -= 1;
    console.log('js');
}
```

js
js
js

In [25]:

```
for(var i = 0; i < 3; i++){
    console.log('js');
}
```

```
js
js
js
```

In [26]:

```
// break, continue
for(var i = 0; i < 5; i++){
    if(i === 2){
        continue;
    }
    console.log(i, 'js');
    if(i >= 3){
        break;
    }
}
```

```
0 'js'
1 'js'
3 'js'
```

In [27]:

```
// 실수하기 쉬운 코드 : 부동소수점 문제
var data1 = 0.1, data2 = 0.2;
data1 + data2 === 0.3
```

Out[27]:

```
false
```

In [28]:

```
// 소수 셋째 자리에서 반올림
Math.round(1.2345 * 1000) / 1000
```

Out[28]:

```
1.235
```

In [29]:

```
Math.round((data1 + data2) * 10) / 10
```

Out[29]:

```
0.3
```

In [30]:

```
Math.round((data1 + data2) * 10) / 10 === 0.3
```

Out[30]:

```
true
```

In [31]:

```
// 로또번호 출력 : 중복허용
// 1 ~ 45 숫자 6개 출력
var count = 6, lotto = '';
for(var i = 0; i < count; i++){
    var randomNumber = Math.ceil(Math.random() * 44) + 1;
    lotto = lotto + randomNumber;
    if(i !== count - 1) lotto += ' ';
}
console.log(lotto);
```

```
41 37 42 6 5 2
```

6. function : 함수

- 중복코드를 묶어서 코드 작성 및 실행 : 코드 유지보수 증대
- 사용법 : 함수선언(코드작성) > 함수호출(코드실행)
- function, argument, parameter, 표현식, 선언식, hoisting, scope, 익명함수, return

In [32]:

```
// 로또번호 출력
var count = 6, lotto = '';
for(var i = 0; i < count; i++){
    var randomNumber = Math.ceil(Math.random() * 44) + 1;
    lotto = lotto + randomNumber;
    if(i !== count - 1) lotto += ' ';
}
console.log(lotto);

// js 문자열 출력
console.log('js');

// 로또번호 출력
var count = 6, lotto = '';
for(var i = 0; i < count; i++){
    var randomNumber = Math.ceil(Math.random() * 44) + 1;
    lotto = lotto + randomNumber;
    if(i !== count - 1) lotto += ' ';
}
console.log(lotto);
```

```
20 26 42 9 27 35
js
37 15 20 39 2 26
```

In [33]:

```
// 함수선언 : 코드작성
function showLotto(){
    var count = 6, lotto = '';
    for(var i = 0; i < count; i++){
        var randomNumber = Math.ceil(Math.random() * 44) + 1;
        lotto = lotto + randomNumber;
        if(i !== count - 1) lotto += ' ';
    };
    console.log(lotto);
};
```

In [34]:

```
// 함수호출 : 코드실행
showLotto();
```

```
9 23 19 27 5 45
```

In [35]:

```
function showLotto(){
    var count = 6, lotto = '';
    for(var i = 0; i < count; i++){
        var randomNumber = Math.ceil(Math.random() * 44) + 1;
        lotto = lotto + randomNumber;
        if(i !== count - 1) lotto += ' ';
    }
    console.log(lotto);
};

// 로또번호 출력
showLotto();

// js 문자열 출력
console.log('js');

// 로또번호 출력
showLotto();
```

```
4 31 42 11 42 39
js
35 40 17 7 33 2
```

In [36]:

```
// argument, parameter : 함수 호출시 함수 선언하는 코드로 데이터 전달
function showLotto(count){
    var lotto = '';
    for(var i = 0; i < count; i++){
        var randomNumber = Math.ceil(Math.random() * 44) + 1;
        lotto = lotto + randomNumber;
        if(i !== count - 1) lotto += ' ';
    }
    console.log(lotto);
};

// 로또번호 출력 : 6개
showLotto(6);

// 로또번호 출력 : 7개
showLotto(7);
```

```
31 27 45 29 14 40
13 44 16 31 15 5 24
```

In [37]:

```
// 함수도 function 데이터 타입을 갖는 변수
typeof showLotto;
```

Out[37]:

```
'function'
```

In [38]:

```
// 함수선언 1 : 함수 선언식
function plus1(n1, n2){
    console.log(n1 + n2)
}
plus1(1, 2);
```

```
3
```

In [39]:

```
// 함수선언 2 : 함수 표현식
var plus2 = function(n1, n2){
    console.log(n1 + n2)
};
plus2(1, 2);
```

```
3
```

In [40]:

```
// 선언식과 표현식의 차이
// 표현식은 함수 호출 후에 함수가 선언되면 에러 발생
plus3(1, 2);

var plus3 = function(n1, n2){
    console.log(n1 + n2)
};
```

```
evalmachine.<anonymous>:3
plus3(1, 2);
^

TypeError: plus3 is not a function
    at evalmachine.<anonymous>:3:1
    at Script.runInThisContext (vm.js:122:20)
    at Object.runInThisContext (vm.js:329:38)
    at run ([eval]:1020:15)
    at onRunRequest ([eval]:864:18)
    at onMessage ([eval]:828:13)
    at process.emit (events.js:198:13)
    at emit (internal/child_process.js:832:12)
    at process._tickCallback (internal/process/next_tick.js:63:19)
```

hoisting : 호이스팅

- 선언식으로 변수(함수)가 선언되면 코드의 최상단으로 올라가서 선언됨

In [41]:

```
// 선언식은 함수 호출 후에 함수가 선언되어도 함수 호출 가능
plus4(1, 2);

function plus4(n1, n2){
    console.log(n1 + n2)
};
```

```
3
```

scope : 스코프

- 함수밖 : 전역변수 : global
- 함수안 : 지역변수 : local

In [42]:

```
var data = 10;
function change(){
    data = 20; // var 사용하지 않으면 전역변수로 사용
}
change();
console.log(data);
```

```
20
```

In [43]:

```
var data = 10;
function change(){
    var data = 20; // var 사용하면 지역변수로 선언
}
change();
console.log(data);
```

```
10
```

anonymous function : 익명함수

- 선언과 동시에 호출
- 전역함수로 사용할수 없도록 선언 > 외부에서 내부의 함수나 변수 사용 불가 > 보안

In [44]:

```
function minus1(n1, n2){
    console.log(n1 - n2);
}
minus1(1, 2);
```

```
-1
```

In [45]:

```
(function minus2(n1, n2){
    console.log(n1 - n2);
}(2, 4));
```

```
-2
```

In [46]:

```
// 전역영역에서 호출 불가능
minus2(4, 7);
```

```
evalmachine.<anonymous>:2
minus2(4, 7);
^

ReferenceError: minus2 is not defined
    at evalmachine.<anonymous>:2:1
    at Script.runInThisContext (vm.js:122:20)
    at Object.runInThisContext (vm.js:329:38)
    at run ([eval]:1020:15)
    at onRunRequest ([eval]:864:18)
    at onMessage ([eval]:828:13)
    at process.emit (events.js:198:13)
    at emit (internal/child_process.js:832:12)
    at process._tickCallback (internal/process/next_tick.js:63:19)
```

return

- 코드 실행 중단
- 함수 코드 실행 결과를 출력 및 변수에 저장

In [47]:

```
function plus(n1, n2){
    return n1 + n2;
}
result = plus(1, 2);
console.log(result);
```

```
3
```

In [48]:

```
// javascript의 default parameter 설정
var plus = function(n1, n2){
    console.log(n1, n2);
    n2 = n2 || 10;
    return n1 + n2;
}
console.log('argument 2 :', plus(20, 30));
console.log('argument 1 :', plus(20));
```

```
20 30
argument 2 : 50
20 undefined
argument 1 : 30
```

Module Pattern : 모듈패턴

- 모듈패턴을 이용한 객체생성

In [49]:

```
// Account 객체가 있으면 있는 객체 사용하고, 없으면 새로운 객체 생성
var Account = Account || {};
(function(_Account){

    var balance = 0;

    function getBalance(){
        return balance;
    }

    _Account.getBalance = function(){
        console.log('getter');
        return getBalance();
    }

    _Account.setBalance = function(money){
        console.log('setter', money);
        balance = money;
    }

    _Account.deposit = function(money){
        balance += money;
    }

    _Account.withdraw = function(money){
        balance -= money;
    }

})(Account);
```

In [50]:

```
// getter
console.log(Account.getBalance());
```

```
getter
0
```

In [51]:

```
// setter
Account.setBalance(10000);
```

```
setter 10000
```

In [52]:

```
// getter
console.log(Account.getBalance());
```

```
getter
10000
```

In [53]:

```
Account.withdraw(2000);
```

In [54]:

```
// getter
console.log(Account.getBalance());
```

```
getter
8000
```

7. 객체 : Object

- Array(배열), Class(클래스) 문법

In [55]:

```
// Arrary
var data = [1, 2, 3, 'A', 'B'];
console.log(typeof data, data);
```

```
object [ 1, 2, 3, 'A', 'B' ]
```

In [56]:

```
for(var i = 0; i < data.length; i++){
    console.log(i, data[i]);
}
```

```
0 1
1 2
2 3
3 'A'
4 'B'
```

In [57]:

```
// array.push() : 데이터 추가
var data = [1, 2, 3, 'A', 'B'];
data.push('C');
console.log(data);
```

```
[ 1, 2, 3, 'A', 'B', 'C' ]
```

In [58]:

```
function showLotto(count){
    var lotto = [];
    while(true){
        var randomNumber = Math.ceil(Math.random() * 44) + 1;
        var isNumber = false;
        for(var i = 0; i < lotto.length; i++){
            if(lotto[i] === randomNumber) isNumber = true;
        }
        if(!isNumber) lotto.push(randomNumber);
        if(lotto.length >= count){
            lotto.sort(function(a, b){ return a - b; });
            break;
        }
    }
    return lotto;
};
showLotto(6)
```

Out[58]:

```
[ 13, 19, 25, 26, 36, 45 ]
```

In [60]:

```
// Object : 객체
```

In [61]:

```
// 객체 생성 1
var obj = { name: 'andy' };
console.log(obj);
```

```
{ name: 'andy' }
```

In [65]:

```
// 객체 생성 2
function Person(name) {
    this.name = name;
}
var person = new Person('andy');
console.log(person);
```

```
Person { name: 'andy' }
```

In [66]:

```
console.log(typeof obj, typeof person);
```

```
object object
```

In [75]:

```
// 객체의 변수 추가 및 수정
var obj = {};
obj.name = 'andy';
obj['addr'] = 'seoul';
console.log(obj);
```

```
{ name: 'andy', addr: 'seoul' }
```

In [76]:

```
// 데이터 읽어오기
console.log(obj['name'], obj.addr);
```

```
andy seoul
```

In [77]:

```
// 객체에 함수 저장
var calc = {
    plus: function(n1, n2){
        return n1 + n2;
    }
}
console.log(calc.plus(1, 2));
```

```
3
```

In [78]:

```
// 객체의 변수(함수) 삭제
var obj = {
    name: 'andy',
    addr: 'seoul'
};
console.log(obj);
delete obj.addr;
console.log(obj);
```

```
{ name: 'andy', addr: 'seoul' }
{ name: 'andy' }
```

json object

- 웹서비스에서 데이터를 주고 받을때 주로 사용하는 데이터 포멧

In [79]:

```
// 객체 > 문자열
var obj = { name: 'andy', addr: 'seoul' };
var json = JSON.stringify(obj);
console.log(typeof obj, obj);
console.log(typeof json, json);
```

```
object { name: 'andy', addr: 'seoul' }
string {"name":"andy","addr":"seoul"}
```

In [80]:

```
// 문자열 > 객체
var jsonObj = JSON.parse(json);
console.log(typeof jsonObj, jsonObj);
```

```
object { name: 'andy', addr: 'seoul' }
```

웹브라우져 객체

- window : 전역객체 : 모든 전역변수를 저장하는 객체
- location : url 데이터를 포함하는 객체
- document : 페이지 문서에 대한 정보를 저장하는 객체

In [ ]:

```
// 브라우져의 console 에서 실행
```

In [ ]:

```
// window 객체
var data = 10;
var plus = function(n1, n2){
    return n1 + n2;
}
console.log(window.data);
console.log(window.plus(1, 2));
```

In [ ]:

```
// location 객체
location.href // 현재 페이지 URL
location.origin // 현재 페이지 domain
locaion.pathname // 현재 페이지 path
location.prot // 현재 페이지 port
```

In [ ]:

```
// documnet 객체
document.title // 현재 페이지 title
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

In [ ]:

In [ ]:

In [ ]: