INTERACTION PROGRAMMING 1

인터랙션 프로그래밍 1

3 Week.

2019. 3. 21.

Math

Math 수학

```
Math
Math.pow          제곱
Math.round        반올림
Math.ceil         올림
Math.floor        내림
Math.sqrt         제곱근
Math.random       랜덤
```

Math 수학

```
Math.pow(3, 2);
Math.round(1.4);
Math.ceil(1.2);
Math.floor(1.2);
Math.sqrt(9);
Math.random();
```

Math 수학

```
Math.pow(3, 2);      3 의 2 제곱
Math.round(1.4);     1.4 의 반올림
Math.ceil(1.2);      1.2 의 올림
Math.floor(1.2);     1.2 의 내림
Math.sqrt(9);        9 의 제곱근
Math.random();       0~1.0 사이의 랜덤한 숫자
```

Math 수학

```
Math.round(100 * Math.random());
```

Data Type

Data Type

```
//Data type.
Boolean
Number
String
undefined
null
1
1 이 아닌 수
NaN
```

```
//Data type.
Boolean      true | false
Number       -1 0 1 2 3 4 5...
String       "a" "b" "c"...
undefined    undefined
null         null
1            Boolean 의 true 로 간주
1 이 아닌 수   Boolean 의 false 로 간주
NaN          성립이 되지 않는 수, 계산할 수 없음을 의미함.
```

typeof

typeof

```
typeof 1
typeof "1"
typeof []
typeof {}
```

typeof

```
typeof 1      Number
typeof "1"    String
typeof []     Array
typeof {}     Object
```

Variable

Variable 변수

```
var variable;
```

Variable 변수

```
var variable = value;
```

변수명

Variable 변수

```
var variable = value;
```

변수명 변수값

Variable 변수

```
var a = 1;
var b = 2;
a + b
```

Variable 변수

```
var a = 1;
var b = "2";
a + b
```

Variable 변수

```
var a = "hello";
var b = "world";
a + b
```

Variable 변수

```
var a = "hello";
a = "HELLO~";
```

var 선언 이후에는 var 를 사용하지 않아도 된다.

Variable 변수의 재활용성

```
var a = "hello";
a = "HELLO~";
a = a + " WORLD!";
a += ":^)";
```

변수는 무수히 많은 재활용이 가능하다.

100 에 10 을 더한 후, 10 으로 나누고, 10 을 뺀 후, 10 을 곱한다.

100 에 10 을 더한 후, 10 으로 나누고, 10 을 뺀 후, 10 을 곱한다.

```
var sum = ( ( (100 + 10) / 10 ) - 10 ) * 10;
```

100 에 10 을 더한 후, 10 으로 나누고, 10 을 뺀 후, 10 을 곱한다.
200 에 20 을 더한 후, 20 으로 나누고, 20 을 뺀 후, 20 을 곱한다.
300 에 20 을 더한 후, 10 으로 나누고, 30 을 뺀 후, 40 을 곱한다.

100 에 10 을 더한 후, 10 으로 나누고, 10 을 뺀 후, 10 을 곱한다.
200 에 20 을 더한 후, 20 으로 나누고, 20 을 뺀 후, 20 을 곱한다.
300 에 20 을 더한 후, 10 으로 나누고, 30 을 뺀 후, 40 을 곱한다.

```
var a = 100;
var b = 10;
var sum = ( ( (a + b) / b ) - b ) * b;
```

100 에 10 을 더한 후, 10 으로 나누고, 10 을 뺀 후, 10 을 곱한다.
200 에 20 을 더한 후, 20 으로 나누고, 20 을 뺀 후, 20 을 곱한다.
300 에 20 을 더한 후, 10 으로 나누고, 30 을 뺀 후, 40 을 곱한다.

```
var a = 200;
var b = 20;
var sum = ( ( (a + b) / b ) - b ) * b;
```

100 에 10 을 더한 후, 10 으로 나누고, 10 을 뺀 후, 10 을 곱한다.
200 에 20 을 더한 후, 20 으로 나누고, 20 을 뺀 후, 20 을 곱한다.
300 에 20 을 더한 후, 10 으로 나누고, 30 을 뺀 후, 40 을 곱한다.

```
var a = 300, b = 20, c = 10, d = 30, e = 40;
var sum = ( ( (a + b) / c ) - d ) * e;
```

100 에 10 을 더한 후, 10 으로 나누고, 10 을 뺀 후, 10 을 곱한다.
200 에 20 을 더한 후, 20 으로 나누고, 20 을 뺀 후, 20 을 곱한다.
300 에 20 을 더한 후, 10 으로 나누고, 30 을 뺀 후, 40 을 곱한다.

```
var a = 1000, b = 300, c = 5, d = 10, e = 60;
var sum = ( ( (a + b) / c ) - d ) * e;
```

변할 수 있는 영역과 변하지 않는 영역으로 구분할 수 있다.
(유지보수)

비교 연산자

비교 연산자

```
//연산자
- + * / %
```

비교 연산자

```
//연산자
- + * / %

//대입 연산자(이항 연산자)
var variable = value;
```

비교 연산자

```
//연산자
- + * / %

//대입 연산자(이항 연산자)
var variable = value;

//비교 연산자
Boolean
true / false;
1 / 0
```

비교 연산자

==

> <

>= <=

비교 연산자

==

> <

>= <=

값이 같은지 큰지 작은지를 비교

비교 연산자

==

> <

>= <=

=>

주의합니다. 다른 명령어입니다.

비교 연산자

```
var a = 1;
var b = 1;
a == b
```

비교 연산자

```
var a = 2;
var b = 1;
a > b
```

비교 연산자

```
var a = 2;
var b = 1;
a < b
```

비교 연산자

```
var a = 2;
var b = 2;
a >= b
```

비교 연산자

```
var a = 1;
var b = 2;
a <= b
```

동등 연산자

동등 연산자

```
//대입 연산자(이항 연산자)
=

//동등 연산자
==
```

동등 연산자

```
var a = 1;
var b = 2;
a == b
```

동등 연산자

```
var a = 1;
var b = 1;
a == b
```

동등 연산자

```
var a = "one";
var b = "하나";
a == b
```

동등 연산자

```
var a = "one";
var b = "하나";
a == b
```

동등 연산자

```
var a = "one";
var b = "one";
a == b
```

동등 연산자

```
var a = 1;
var b = "1";
a == b
```

동등 연산자

```
var a = 1;
var b = "1";
a == b
```

?

동등 연산자

```
var a = 1;
var b = "1";
a === b
```

일치 연산자

일치 연산자

```
//대입 연산자(이항 연산자)
=

//동등 연산자
==

//일치연산자
===
```

일치 연산자

```
var a = 1;
var b = "1";
a == b
```

```
var a = 1;
var b = "1";
a === b
```

정확히 일치 하는지를 비교, **Strict** (엄격한)

동등 연산자는 버그를 발생시킬 위험이 있다.

일치 연산자

```
var a = null;
var a;
```

값이 없는 상태, 의도해서 값이 없는 상태로 만든 것

값이 정의되지 않은 상태

일치 연산자

```
var a = null;
var b;
a == b
```

일치 연산자

```
var a = null;
var b;
a == b
```

```
var a = null;
var b;
a === b
```

일치 연산자

```
0 === -0
```

일치 연산자

```
true == 1
```

일치 연산자

```
true == 1      true === 1
```

일치 연산자

```
NaN === NaN
```

```
NaN === NaN
```

둘 다 **NaN** 이라도 **false** 가 된다.

부정

부정

```
//부정
!=
!==
```

부정

```
var a = 1;
var b = 2;
a == b
```

부정

```
var a = 1;
var b = 2;
a == b

a != b
```

부정

```
var a = 1;
var b = 1;
a != b
```

부정

```
var a = "a";
var b = "b";
a != b
```

부정

```
var a = "a";
var b = "a";
a != b
```

Object

Object

```
var object = {};
```

Object

```
var object = { key : value };
```

Object

```
var person = {
    name : "김용원",
    job : "교수",
    phone : "010-9137-8688",
    email : "rh@102labs.com"
};
```

Object

```
var person = {
    "name" : "김용원",
    "job" : "교수",
    "phone" : "010-9137-8688",
    "email" : "rh@102labs.com"
};
```

Object

```
person.name;
person.job;
person.phone;
person.email;
```

Object

```
person["name"];
person["job"];
person["phone"];
person["email"];
```

조건문

if 안의 조건이 true 인 경우.
if 안의 조건이 false 인 경우.

true 일 경우 if 조건 안의 코드가 실행.
false 일 경우 else 안의 코드가 실행.

조건문

```
var a = 3;
var b = 10;
var c;
if (typeof c === 'undefined') {
    c = a % b;
};
```

조건문

```
var a = 3;
var b = 10;
var c = 0;
if (typeof c === 'undefined') {
    c = a % b;
};
```

조건문

```
var a = 3;
var b = 10;
var c = 0;
if (a && b) {
    c = a % b;
};
```

조건문

```
var a = 10;
var b = 0;
if (a && b) {
    console.log('---->');
};
```

조건문

```
var a = 10;
var b = 0;
if (b || a) {
    console.log('<----');
};
```

Array

Array 객체

한 번에 두가지 이상의 값을 포함할 수 있는 객체

사용빈도가 아주 높다.

Array 배열

```
var a = 10;
var b = 'apple';
var c = null;
var d = a;
var _array = [a, b, c, d];
console.log(_array[3]);
```

반복문

반복문

동일한 동작을 반복 수행할 수 있도록 도와주는 구문

for

while

반복문

```
var _array = ['a', 'b', 'c', 'd', 'e'];
for (var i = 0; i < _; i++) {
    console.log(_________);
};
```

반복문

```
var _array = ['a', 'b', 'c', 'd', 'e'];
for (var i = 0; i < 5; i++) {
    console.log(_array[i]);
};
```

```
var _say = ['Hello', 'My', 'Name', 'is', '_____'];
```

Hello My Name is _____.

위와 같이 출력되도록 반복문을 이용해서 작성해봅시다.

반복문

```
var _say = ['Hello', 'My', 'Name', 'is', '____'];
var say = '';
for (var i = 0; i < 5; i++) {
    say += _say[i];
};
console.log(say);
```

반복문

```
var _say = ['Hello', 'My', 'Name', 'is', '____'];
var say = '';
for (var i = 0; i < 5; i++) {
    if (i !== 0) {
        say += ' ';
    };
    say += _say[i];
    if (i === 4) {
        say += '.';
    };
};
console.log(say);
```