04. Operation & Loop

1. String concatenation

```
console.log('my' + ' cat');
console.log('1' + 2);
console.log(`string literals:
'''
1 + 2 = ${1 + 2}`);
console.log('smarte\'s book');
console.log(`smarte's book`);
```

2. Numeric operators

```
console.log(1 + 1);   // add
console.log(1 - 1);   // substract
console.log(1 / 1);   // divide
console.log(1 * 1);   // mutiply
console.log(5 % 2);   // remainder
console.log(2 ** 3);  // exponentiation
```

3. Increment and decrement operators

```
let counter = 2;
const preIncrement = ++counter;
// counter = counter + 1;
// preIncrement = counter;
console.log(`preIncrement: ${preIncrement}, counter: ${counter}`);

const postIncrement = counter++;
// postIncrement = counter;
// counter = counter + 1;
console.log(`postIncrement: ${postIncrement}, counter: ${counter}`);

const preDecrement = --counter;
console.log(`preDecrement: ${preDecrement}, counter: ${counter}`);

const postDecrement = counter--;
console.log(`postDecrement: ${postDecrement}, counter: ${counter}`);
```

4. Assignment operators

```
let x = 3;
let y = 6;
x += y;   // x = x + y;
```

```
x -= y;
x *= y;
x /= y;
```

5. Comparison operators

```
console.log(10 < 6);    // less than
console.log(10 <= 6);   // less than or equal
console.log(10 > 6);    // greater than
console.log(10 >= 6);   // greater than or equal
```

6. Logical operators: ¦¦ (or), && (and), ! (not)

```
//const value1 = false;
const value1 = true;
const value2 = 4 < 2;

// ¦¦ (or), finds the first truthy value
console.log(`or: ${value1 ¦¦ value2 ¦¦ check()}`);

// && (and), finds the first falsy value
console.log(`and: ${value1 && value2 && check()}`);

// often used to compress long if-statement
// nullableObject && nullableObject.somthing
// if (nullableObject != null) {
//   nullableObject.something;
// }

function check() {
  for (let i = 0; i < 10; i++) {
    //wasting time
    console.log('아이고!!!')
  }
  return true;
}

// ! (not)
console.log(!value1);
```

7. Eaulity

```
const stringFive = '5';
const numberFive = 5;

// == loose quality, with type converson
console.log(stringFive == numberFive);
```

```
console.log(stringFive != numberFive);

// === strict quality, no type converson
console.log(stringFive === numberFive);
console.log(stringFive !== numberFive);

// object quality by reference
const smart1 = { name: 'smart' };
const smart2 = { name: 'smart' };
const smart3 = smart1;
console.log(smart1 == smart2);
console.log(smart1 === smart2);
console.log(smart1 === smart3);

// equality - puzzler
// 0, null, '', undefined -> false로 취급
console.log(0 == false);    // true
console.log(0 === false);   // false
console.log('' == false);   // true
console.log('' === false);  // false
console.log(null == undefined);   // true
console.log(null === undefined);  // false
```

8. Consitional operators: if

```
// if, else if, else
const name = 'smart';
if (name === 'smart') {
  console.log('Welcome, smart!!');
} else if (name === 'coder') {
  console.log('You are amazing coder');
} else {
  console.log('unknown');
}
```

9. Ternary operator: ?

```
// condition ? value1 : value2;
console.log(name === 'smart' ? 'yes' : 'no');

console.clear();
```

10. Switch statement

```
// use for multiple if checks
// use enum-like value check
// use for multiple type checks in TS
```

```
const browser = 'IE';
switch (browser) {
  case 'IE':
    console.log('go away!');
    break;
  case 'Chrome':
    console.log('love you!');
    break;
  case 'Firefox':
    console.log('love you!');
    break;
  default:
    console.log('same all!');
    break;
}
```

11. Loops

```
// while loop, while the condition is truthy,
// body code is executed.
let i = 3;
while (i > 0) {
  console.log(`while: ${i}`);
  i--;
}

// do while loop, body code is executes first,
// then check the condition.
do {
  console.log(`do while: ${i}`);
  i--;
} while (i > 0);

// for loop, for(begin; condition; step)
for (i = 3; i > 0; i--) {
  console.log(`for: ${i}`)
}

for (let i = 3; i > 0; i = i - 2) {
  // inline variablr declaration
  console.log(`inline variable for: ${i}`);
}

// nested loops => Big O(n**2)  -> 성능이 좋지 않음
for (let i = 0; i < 10; i++) {
  for (let j = 0; j < 10; j++) {
    console.log(`i: ${i}, j:${j}`);
  }
}
```

```
// break, continue
// Q1. iterate from 0 to 10 and print only even numbers (use continue)
for (let i = 0; i < 11; i++) {
  if (i % 2 !== 0) {
    continue;
  }
  console.log(`q1: ${i}`)
}

// Q2. iterate from 0 to 10 and print numbers until reaching 8 (use break)
for (let i = 0; i < 11; i++) {
  if (i > 8) {
    break;
  }
  console.log(`q2. ${i}`);
}
```