CONTACT: 010-4114-5410
Discord : newbiejump

2026.06.10

# JAVA SCRIPT

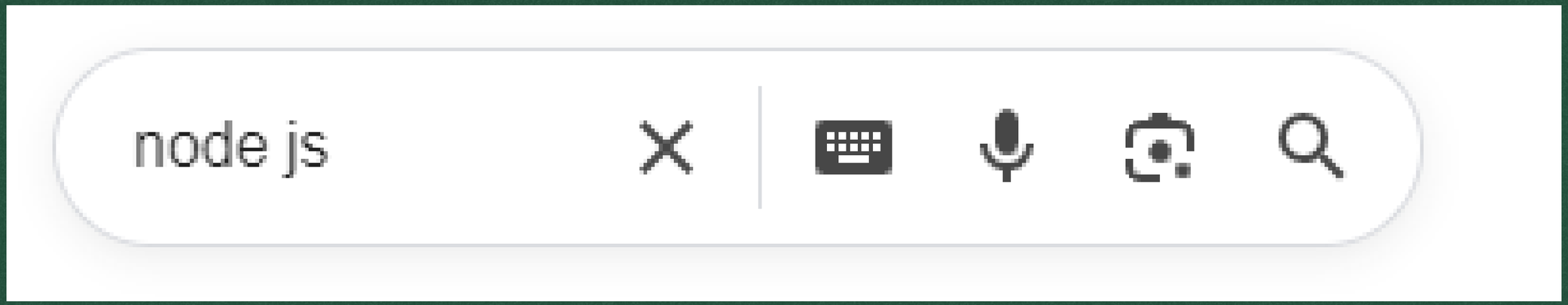
## PYTHON만큼 쉽다

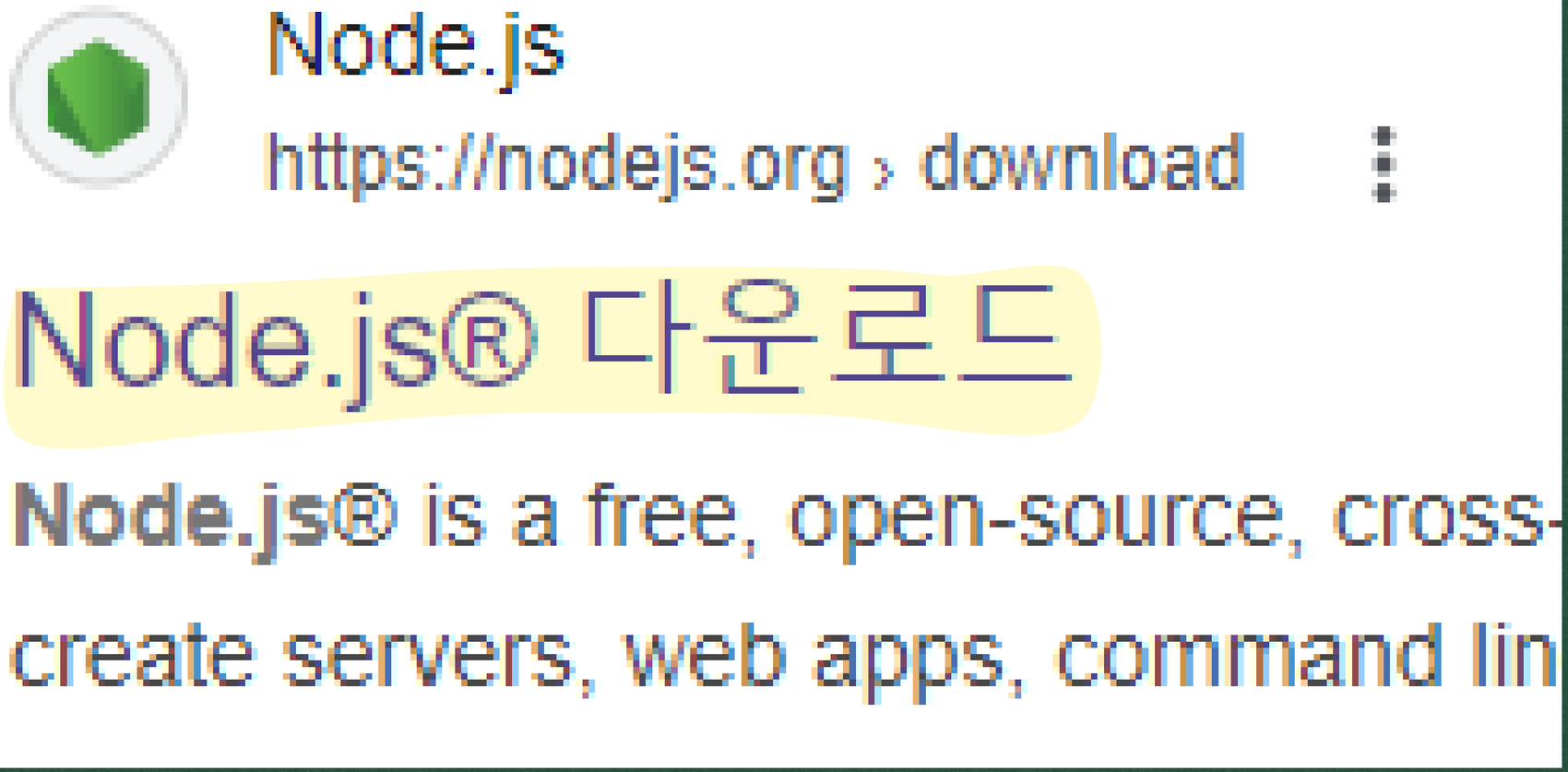
박순민 신유민 황다인

I

# INDEX

0




Node.js
https://nodejs.org › download
Node.js® 다운로드
Node.js® is a free, open-source, cross-
create servers, web apps, command lin

0

또는 x64 아키텍처가 실행 중인 Windows 환경에

Windows 설치 프로그램 (.msi) Standalone Binary (.zip)

또는 x64 아키텍처가 실행 중인 macOS 환경에서 미리

macOS 설치 프로그램 (.pkg) Standalone Binary (.gz)

0

```
console.log("Hello JS");
```

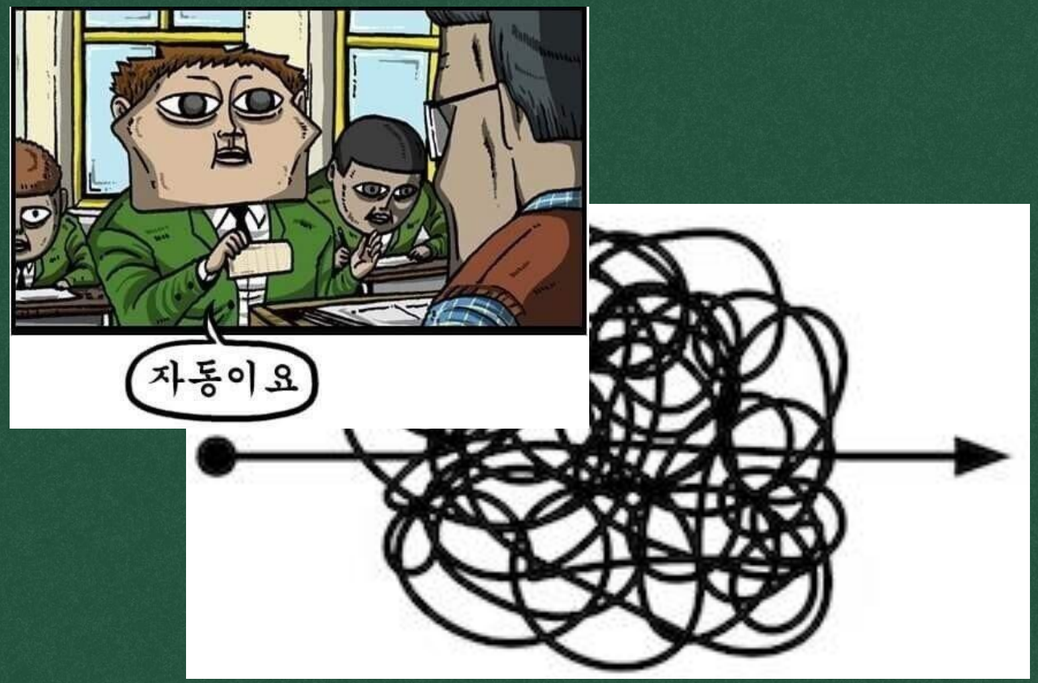
1

# JS란?

JavaScript는 웹 페이지에서 동적이고 복잡한 기능을 구현할 수 있는 스크립팅 및 프로그래밍 언어로, 한 번에 하나의 작업만 처리할 수 있는 싱글 스레드 언어다

1

# 스크립팅?

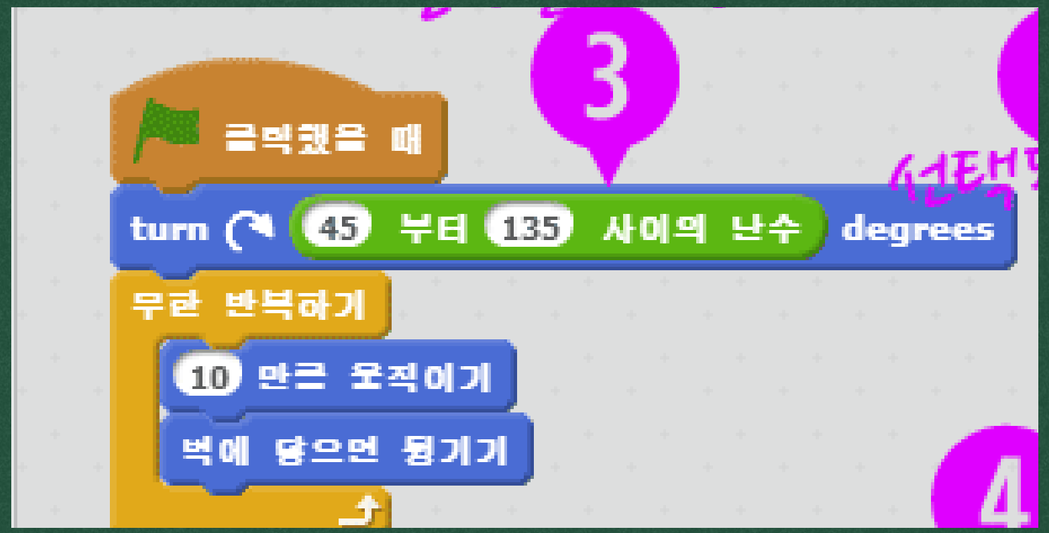



프로세스

1

# 스크립팅?


제어 ←




짧고 간단한 코드

1

# JS는 왜 사용하는가?

높은 호환성

빠른 실행

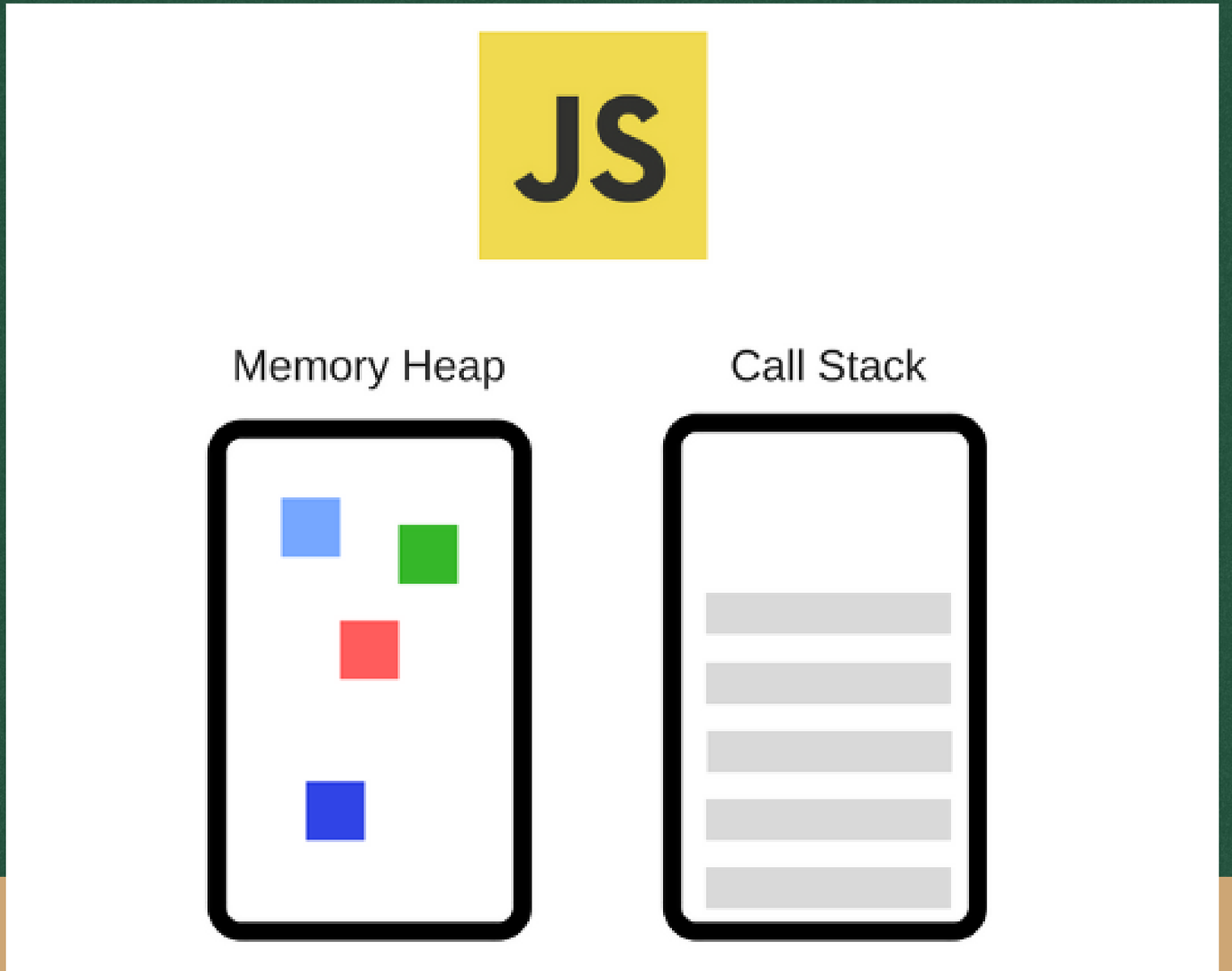
다양한 라이브러리

배우기 쉬움

백엔드 개발 가능

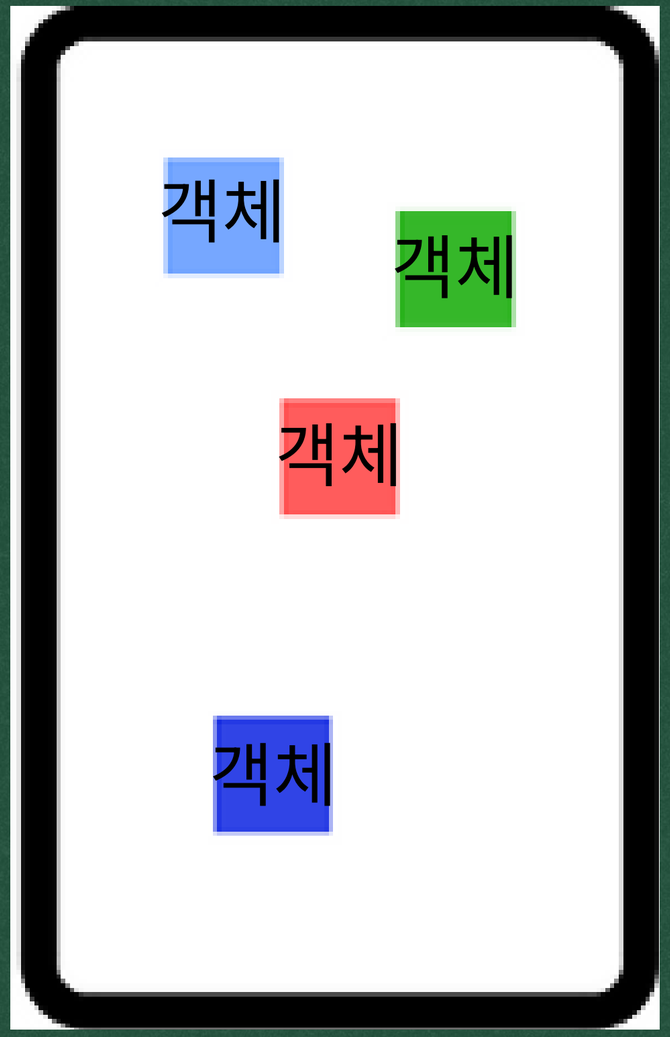
1

# JS는 어떻게 작동하는가



1

# Memory Heap




객체(Object)나 배열 같은 참조 타입 데이터가 저장되는 자유 공간이다.

1

# Call Stack


들어온 코드를 순서(LIFO)대로 실행하는 역할이다

1

# 작동 원리

```
console.log('A');

setTimeout(() => {
    console.log("C");
}, 0);

console.log('B');
```

1

# 작동 원리

Call Stack

Task Queue


```
[Running]
```

1

# 작동 원리

Call Stack

Task Queue


```
[Running]
a
```

1

# 작동 원리

Call Stack


Task Queue


```
[Running]
a
```

1

# 작동 원리

Call Stack

Task Queue


```
[Running]
a
b
```

1

# 작동 원리

Call Stack

Task Queue

```
[Running]
a
b
c
```

2

# 변수

**변수 선언법**

VAR, LET

CONST

2

# 변수 선언법

```
var v = 1; //변수를 선언. 추가로 동시에 값을 초기화.

let l = 2; //블록 스코프* 지역 변수를 선언. 추가로 동시에 값을 초기화.

const c = 3; //블록 스코프* 읽기 전용 상수를 선언.

let noInitialValue; /*
                        var과 let은 초기값을 넣지 않고 선언할 수 있다.
                        undefined값을 가진다.
                    */
```

블록 스코프* :블록('{}'), 제어문(if, switch/case), 반복문(for, while) 내부에서 선언 된 변수

*주의*: abc와 Abc는 전혀 다른 변수임

2

# 변수 선언법

```
let 1abc = 1; //변수명의 첫 번째는 문자, $, _ 만을 사용해야한다.

let asdf* = 2; //$, _ 외의 특수 문자는 사용 불가하다.

let for = 3; //예약어*는 사용이 불가하다.

const ab; //const를 선언할때는 초기값을 무조건 넣어야한다.
```

이런식으로는 사용이 불가함

2

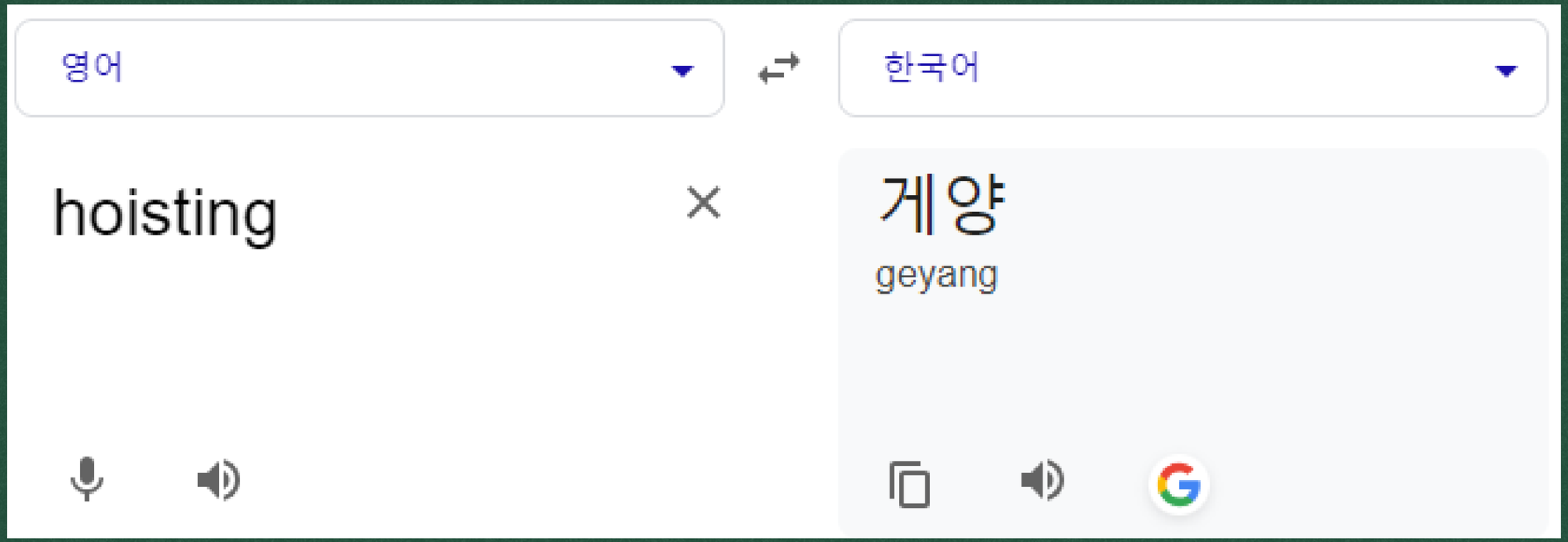
# var ? let ?

Q : var, let 똑같은 거 아님?

A : TDZ 차이
(TDZ 설명은 나중에 나올 예정)

2 - 1

# 호이스팅이란?



2-1

# 변수 호이스팅이란?

변수가 어떤 의미에서 함수나 문의 최상단으로 올려지는것

2-1

# 변수 호이스팅이란?

```
console.log(x);
var x = 'a';
```

```
[Running]
undefined
```

2-1

# 변수 호이스팅이란?

```
console.log(x);
var x = 'a';
```

=

```
var x;
console.log(x);
x = 'a';
```

2-1

# 변수 호이스팅이란?

```
console.log(x);
let x = 'a';
```

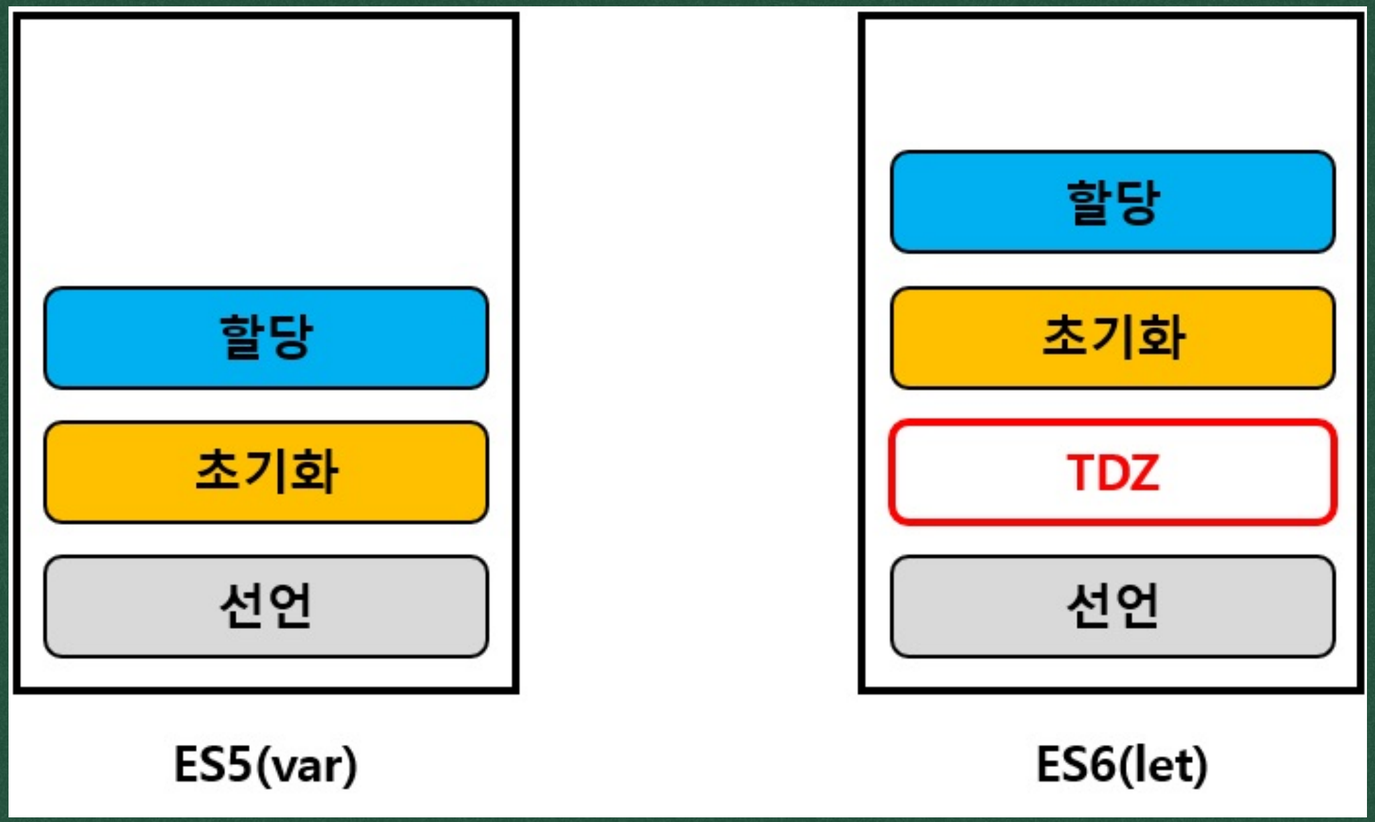
```
[Running] node "c:\Users\user\Desktop\asdf\i.js"
c:\Users\user\Desktop\asdf\i.js:28
console.log(x);
            ^

ReferenceError: Cannot access 'x' before initialization
```

2 - 1

TDZ(Temporal Dead Zone) : 일시적 사각 지대

선언 전에 변수에 접근하는 것을 금지하는 역할

# 왜 var를 사용하지않는가




+ var은 같은 변수명을 재사용할 수 있는 문제가 있다

Q

# Quiz

다음 문항중 변수 이름으로써 적절한 것은?

1.const 2. 1str 3. star* 4. num1 5. if

3

# 자료형

숫자형, 문자열

참과 거짓

배열

3

# 숫자형

숫자형?

다양한 형태의 숫자를 표현해서 나타내는 값을 말하는 변수

JS에서는 숫자의 형태를 구체적으로 나누어 정의하지 않는다.

3

# 숫자형

숫자형?

**C++ : int, float, long, short, byte로 정의해서 숫자를 표현하여 사용**

3

# 숫자형

0이 많은 숫자를 쓸 때는 e를 사용한다

```
let billion = 1e6;
```

```
let ms = 1e-6;
```

```
console.log(1e9);
```

3

# 숫자형

16진수, 8진수, 2진수로 숫자를 표현할 수 있다

3

# 숫자형

16진수는 0x를 사용하여 표현한다

```
console.log(0xff); // 255
```

```
console.log(0xFF); // 255
```

대-소문자 구분 상관 없이 같은 값이 나온다

3

# 숫자형

2진수와 8진수는 드물게 사용하나, 0b와 0o를 사용한다

```
let a = 0b11111111 // 255의 2진수
let b = 0o377 // 255의 8진수
```

3

# 숫자형

QUIZ. a와 b는 같을까? 다를까?

→ 같습니다

```
alert(a == b) // true, 진법은 다르나 a와 b는 같은 수 이기 때문임.
```

3

# 숫자형

정상적인 것도 있으면 이상한 것도 있죠

3

# 숫자형

0.1 + 0.2의 답이 뭐라고 생각하나요?

??? : 0.3 아닌가요?

3

# 숫자형

??? : 0.3 아닌가요?

→ 하지만 이것은 틀렸습니다

```
alert( 0.1 + 0.2 == 0.3 ); // false
```

3

# 숫자형

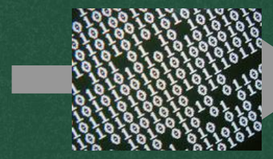
아니 그러면 뭔데요?

```
alert( 0.1 + 0.2 ); // 0.30000000000000004
```

3

# 숫자형

0.1 , 0.2



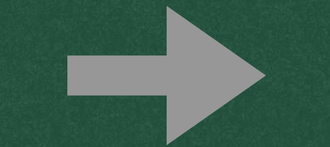
2진
수

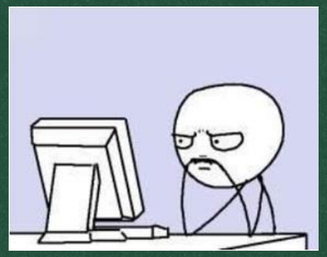
무한소수

3

# 숫자형

```
0.0001100110011001100110011...
0.0011001100110011001100110...
```





```
0.30000000000000004
```

배정밀도 부동소수점

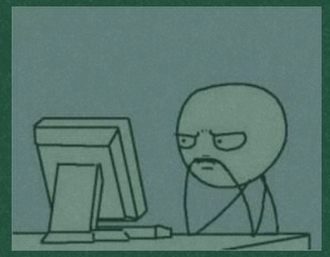
3

# 숫자형

0.0001100110011001100110011...

0.0011001100110011001100110...

## 부동소수점 오차



3

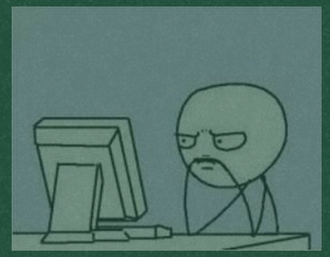
# 숫자형

0.0001100110011001100110011...

0.0011001100110011001100...

0.30000000000000004

## 자세한건 찾아보는걸로



3

# 연산자

연산자는 말 그대로 연산을 해주는 글자로
할당 연산자,비교 연산자,산술 연산자,비트 연산자,논리 연산자
등의 연산자 유형이 있다

3

# 산술 연산자

| | |
|---|---|
| 나머지 (`%`) | 이항 연산자입니다. 두 피연산자를 나눴을 때의 나머지를 반환합니다. |
| 증가 (`++`) | 단항 연산자입니다. 피연산자에 1을 더합니다. 전위 연산자(`++x`)로 사용하면 피연산자에 1을 더한 값을 반환합니다. 후위 연산자(`x++`)로 사용한 경우 피연산자에 1을 더하기 전의 값을 반환합니다. |
| 감소 (`--`) | 단항 연산자입니다. 피연산자에서 1을 뺍니다. 반환 값은 증가 연산자처럼 동작합니다. |
| 단항 부정 (`-`) | 단항 연산자입니다. 피연산자의 부호를 반대로 바꾼 값을 반환합니다. |
| 단항 플러스 (`+`) | 단항 연산자입니다. 피연산자가 숫자 타입이 아니면 숫자로 변환을 시도합니다. |
| 거듭제곱 (`**`) | `base^exponent`, 즉 `base`를 `exponent`로 거듭제곱한 결과를 반환합니다. |

3

# 논리 연산자

| | |
|---|---|
| 논리 AND (`&&`) | `expr1 && expr2` |
| 논리 OR (`||`) | `expr1 || expr2` |
| 논리 NOT (`!`) | `!expr` |

3

# 문자열

String(문자열)은 문자들이 연속적으로 나열된 것으로,
작은따옴표, 큰따옴표, 백틱 등으로 표현할 수 있다.
0개 이상의 문자를 포함한다.

3

# 문자열

```
let dQ = "쌍 따옴표";

let sQ = '따옴표';

let bT = `백틱`;

console.log(typeof dQ);
console.log(typeof sQ);
console.log(typeof bT);
```

→

```
[Running]
string
string
string
```

3

# 문자열

Q: ‘, “, ` 를 그대로 사용하고 싶어요.

A: 다른 따옴표로 감싸거나 \를 활용하면 가능

3

# 문자열

```
let case1 = " ' "; // or ` ' `

let case2 = ' " '; // or ` " `

let case3 = " ` "; // or ' ` '

let case4 = ' \' ';

let case5 = " \" ";

let case6 = ` \` `;
```

```
[Running]
 '

 ''

 `

 '

 ''

 `
```

3

# + 특수 문자

| | |
|---|---|
| `\0` | Null Byte |
| `\b` | Backspace |
| `\f` | Form feed |
| `\n` | New line |
| `\r` | Carriage return |
| `\t` | Tab |
| `\v` | Vertical tab |
| `\'` | Apostrophe 혹은 작은 따옴표 |
| `\"` | 큰 따옴표 |
| `\\` | 백슬래시 |

3

# 문자열 인덱싱 / 슬라이싱

```
let str1 = 'Hello World';
console.log(str1.slice(0, 6));
```

```
[Running]
Hello
```

*주의점* : 슬라이싱 할 때는 [시작 위치 : 끝 위치 +1]로 잘라야 한다

3

# 문자열 인덱싱 / 슬라이싱

```
let str1 = 'Hello World';
console.log(str1.slice(0, 6));
```

파이썬은 문자열[a:b]로 잘랐는데 이건 뭐지?

3

# 문자열 매서드

```
let str1 = ' Hello World ';

console.log(str1.length);

console.log(str1.toUpperCase());

console.log(str1.toLowerCase());

console.log(str1.includes('World'));

console.log(str1.indexOf('o'));
```

2

# 문자열 매서드

```
13
HELLO WORLD
hello world
true
5
```

2

# 문자열 매서드

```
console.log(str1.slice(0, 6));

console.log(str1.replace('l', '1'));

console.log(str1.replaceAll('l', '1'));

console.log(str1.split(' '));

console.log(str1.trim());
```

2

# 문자열 매서드

```
Hello
Hello World
Hello World
[ '', 'Hello', 'World', '' ]
Hello World
```

2

# 참과 거짓

값을 비교, 조건 구분할 때 등등으로 참(true)과 거짓(false)이 많이 사용함.

2

# 참과 거짓

```
console.log(5 > 1); // true
console.log(3 >= 4); // false
console.log(2 < 5); // true
console.log(1 <= 0); // false
```

```
let value1 = 50;
let value2 = 30;

console.log(value1 > value2); // true
console.log(value1 < value2); // false
```

비교 연산자를 사용하여 참과 거짓 출력

2

# 참과 거짓

```
let value1 = true;
let value2 = false;

console.log( !value1 ); // true -> false
console.log( !value2 ); // false -> true
```

논리연산자를 사용하여 출력

2

# 참과 거짓

```
console.log(Boolean('Hi')); // true

console.log(Boolean(1)); // true

console.log(Boolean(0)); // false
```

Boolean 함수를 사용하여 참과 거짓 출력

2

대표적인 True가 되는 값

‘0' (문자열)　　‘false’ (문자열)

[] (빈 배열)　　{} (빈 객체)

function () {} (빈 함수)

2

대표적인 False가 되는 값

undefined

null

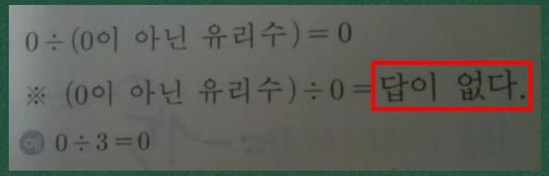
false

0, -0 (숫자)

NaN (오류값)

2

# 참과 거짓




거짓이거나 비어있으면 false, 그 외에는 true

라고 생각하면 쉽다.

2

# 참과 거짓

여기서 QUIZ

“” 와 “ ” 중에서 True가 뭐고, False가 뭘까?

2

# 참과 거짓

```
alert(Boolean("")); // False

alert(Boolean(" ")); // True
```

2

# 참과 거짓

## Why?

공백은 문자열로 인식하기 때문이다

```
alert(Boolean("")); // False
alert(Boolean(" ")); // True
```

2

# 참과 거짓

```
Hello World
```

몇글자일까?

2

# 참과 거짓

1 2 3 4 5  6 7 8 9 10

```
Hello World
```

일상

2

# 참과 거짓

1 2 3 4 5 6 7 8 9 10 11

```
Hello World
```

JS

2

# 참과 거짓

```
let str = "";

alert(str2.length); // 0
```

```
let str2 = " ";

alert(str2.length); // 1
```

Q

# Quiz

지수 표기법(e)을 활용하여 10억이라는 숫자를 변수에 저장하여 출력하세요

```
let billion = 1e9;
cosole.log(billion)
;
```

Q

# Quiz

['Hello World'] 대괄호 안의 문자를 출력하시오.

```
console.log("'Hello World'");
console.log('\'Hello World\'');
```

.

.

Q

# Quiz

문자열에서 name0과 age를 변수의 값으로 바꿔 출력하시오.

let string = "My name is name0, I'm age years old";

let name = "홍길동";

let age = "20";

(replace 메서드 사용)

Q

# Quiz

string.replace("name0", name).replace("age", age)

2

# 배열

```
let name1 = "박순민";

let name2 = "신유민";

let name3 = "황다인";
```

.

.

.

2

# 배열

```
let name = [ //const, var도 가능
    'element0', 'element1', 'element2', '...' , 1, 2, 3, 4
]; //다른 자료형 함께 넣기 가능
```

2

# 배열

```
let names = ["박순민", "신유민", "황다인"];
```

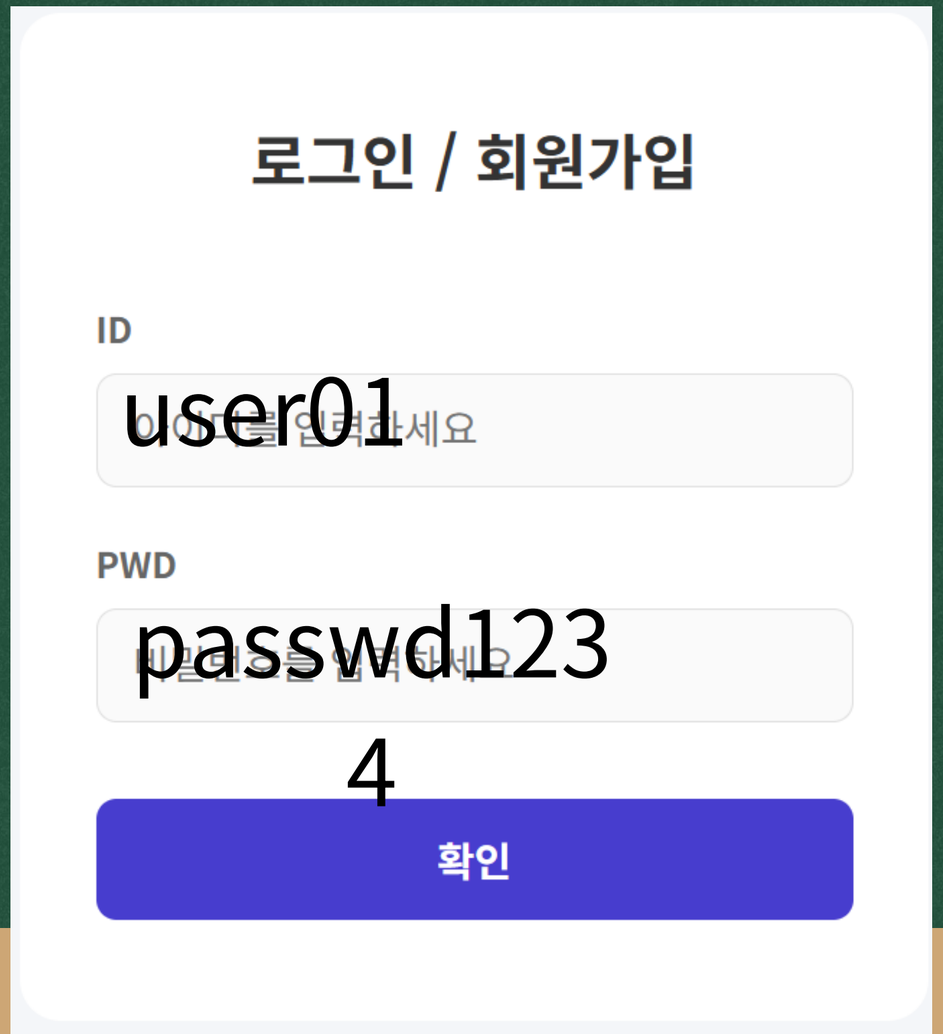
2

# 배열

배열을 수정한다면...

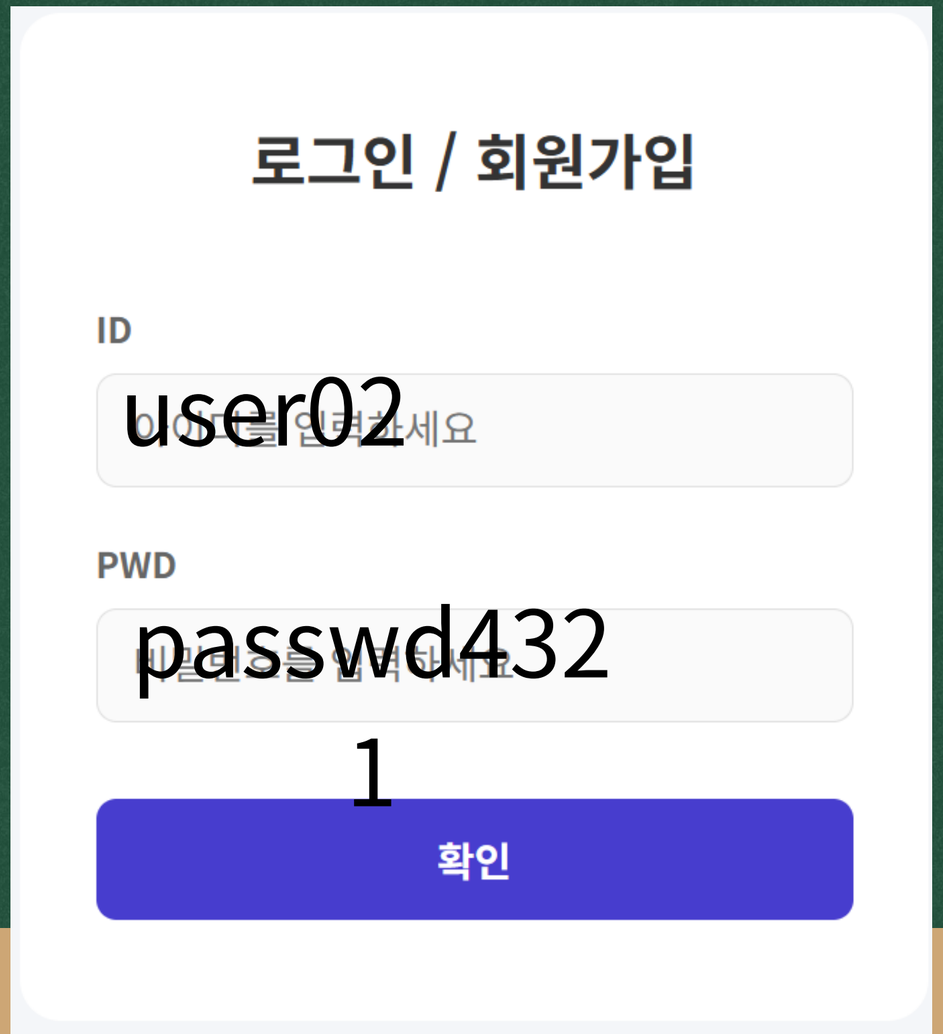
2

# 배열



2

# 배열



2

# 배열

```
let login_info = [['user01', 'passwd1234']];
```

```
let login_info = [['user01', 'passwd1234'], ['user02', 'passwd4321']];
```

2

# 배열 매서드

push, pop, shift, unshift, sort, reverse,
copywithin, foreach, filter, map

2

# 배열을 수정하는 매서드

```
let arr1 = ['element1', 'element2'];

arr1.push('element3');

arr1.pop();

arr1.unshift('element0')

arr1.shift();
```

```
['element1', 'element2','element3'  ];
```

2

# 배열을 수정하는 매서드

```
let arr1 = ['element1', 'element2'];

arr1.push('element3');

arr1.pop();

arr1.unshift('element0')

arr1.shift();
```

```
['element1', 'element2',                    ];
```

2

# 배열을 수정하는 매서드

```
let arr1 = ['element1', 'element2'];

arr1.push('element3');

arr1.pop();

arr1.unshift('element0')

arr1.shift();
```

```
['element0', 'element1', 'element2'  ];
```

2

# 배열을 수정하는 매서드

```
let arr1 = ['element1', 'element2'];

arr1.push('element3');

arr1.pop();

arr1.unshift('element0')

arr1.shift();
```

```
[              , 'element1', 'element2'   ];
```

2

# 배열을 수정하는 매서드

```
let arr1 = [2, 1, 4, 6, 3];

arr1.sort()

arr1.splice(4, 0, 5)

arr1.copyWithin(1, 0, 1)
```

```
[1, 2, 3, 4, 6    ]
```

2

# 배열을 수정하는 매서드

```
let arr1 = [2, 1, 4, 6, 3];

arr1.sort()

arr1.splice(4, 0, 5)

arr1.copyWithin(1, 0, 1)
```

```
[1, 2, 3, 4, 5, 6]
```

2

# 배열을 수정하는 매서드

```
let arr1 = [2, 1, 4, 6, 3];

arr1.sort()

arr1.splice(4, 0, 5)

arr1.copyWithin(1, 0, 1)
```

```
[1, 1, 3, 4, 5, 6]
```

2

# sort

```
arr1.sort()
```

2

# sort

```
let arr1 = [3, 100, 21];

arr [Running] node
    [ 100, 21, 3 ]
console.log(arr1)
```

2

# sort

```
arr
[Running] node
[ 3, 21, 100 ]
-b )
```

↓

양수면 오른쪽으로 음수면 왼쪽으로

2

# 배열을 수정하지 않는 매서드

```
let arr1 = [1, 2, 3, 4, 5];

arr1.forEach((e, i) => {
    console.log(`${i}: ${e}`);
});

let arr2 = arr1.map(num => num ** 2);

let arr3 = arr1.filter(i => i>=3);

console.log(arr1.slice(0, 2));

let arr4 = arr1.concat(arr2);

let arr5 = [[1, 2], [3, 4], [5, [6]]];
console.log(arr5.flat())
console.log(arr5.flat(2))

let total = arr1.reduce((a, b) => a+b);
```

2

# 배열을 수정하지 않는 매서드

```
0: 1
1: 2
2: 3
3: 4
4: 5
[ 1, 4, 9, 16, 25 ]
[ 3, 4, 5 ]
[ 1, 2 ]
```

```
[
  1, 2, 3,  4,  5,
  1, 4, 9, 16, 25
]
[ 1, 2, 3, 4, 5, [ 6 ] ]
[ 1, 2, 3, 4, 5, 6 ]
15
```

Q

# Quiz

배열의 값들을 역순으로 (5 -> 1) 배열하세요

let arr1 = [21, 12, 42, 53, 3];

(sort 함수 사용)

arr1.sort((a, b) => b-a)

Q

# Quiz

배열의 값들에 +1을 한 새로운 배열을 만들어 출력하세요

let arr2 = [2, 4, 6, 8, 10];

(map 함수 사용)

```
console.log(arr2.map(i => i+1));
```

2

# 제어문

**조건문**

**반복문**

3

# 조건문

IF문, switch/case
조건에 따라 다른 코드가 실행되는 구문

3

# 조건문

대중적으로 사용되는 IF문을 중점으로 배워보자

3

# IF

if ★★ 저장272

접속사

1. (만약) ... 면

3

# IF

```
let 조건 = true;


if (조건) {
    console.log('진실');
};
```

조건이 진실 → 코드 실행

3

# IF

Q: 조건을 여러개 검사하고 싶으면 어떻게 해야하나요?
A: else if, else를 사용하면 가능하다

3

# IF

```
if (조건 == 'a') {
    console.log('조건은 a이다');
} else if (조건 == 'b'){
    console.log('조건은 b이다');
} else {
    console.log('조건은 a도 b도 아니다');
}
```

else if: 위의 조건이 아니라면~

else: 위의 모든 조건을 충족하지 않으면~

3

# == vs ===

= 은 할당할 때, == 은 같은지 판별할 때 사용된다

== 을 '동등연산자' 라고 부른다

```
alert(4 == 4); // True

alert(7 == 2); // False
```

3

# == vs ===

```
let str = "Hello";
let str2 = 'world';

alert(str == str2); // Fals
alert("password" == "passwo
```

문자열

```
let boo = true;
let boo2 = false;

alert(boo == boo2); // false
alert(false == false); // true
```

불린형(boolean)

3

# == vs ===

그런데, 0과 false을 구분하고 싶을 때

동등연산자를 사용하면?

```
alert( 0 == false ); // true
```

0과 false가 구분이 안되고, 같다고 한다.

3

# == VS ===

QUIZ. ‘’과 false는 구분이 될까?

```
alert('' == false); // true
```

구분이 안된다

3

# == VS ===

Why?

JS는 비교하려는 값의 데이터타입이 다르면

숫자형으로 바뀌기 때문이다.

3

# == vs ===

ex)

```
alert('01' == 1); // true
```

문자열 ‘01’이 숫자 1로 변환되어 비교 진행된다

3

# == vs ===

```
alert( true == 1 ); // true

alert( false == 0 ); // true
```

boolean의 경우에는
True는 1로, false는 0으로 변환한 뒤에 비교한다

3

# == vs ===

Quiz. null과 undefined을 동등연산자로 비교하면?

```
alert( null == undefined ); // true
```

3

# == vs ===

```
alert( null == undefined ); // true
```

둘다 비어있고, 값이 없다는 비슷한 의미로 사용

3

# == vs ===

```
alert( 0 == false ); // true
```

```
alert('' == false); // true
```

3

# == vs ===

```
alert( 0 == false ); // true
```

어떻게 해야 구분이 될까?

```
alert('' == false); // true
```

3

# == VS ===

동등 연산자 → 일치 연산자

3

== VS ===

일치연산자

3

# == vs ===

데이터 타입의 변환 없이 비교할 때

일치연산자를 사용한다.

```
alert( 0 === false ); // false
```

3

# == vs ===

Quiz. null과 undefined를 일치연산자로 비교한다면?

```
alert( null === undefined ); // false
```

값과 타입을 비교하기 때문

3

# == vs ===

Quiz. null과 undefined를 일치연산자로 비교한다면?

## 직접 타입을 확인해보자

값과 타입을 비교하기 때문

3

# == vs ===

```
object
undefined
```

3

# switch / case

```
let 조건 = 1;

switch (조건) {
    case 1:
        console.log('조건은 1');
        break;
    case 2:
        console.log('조건은 2');
        break;
    default:
        console.log('default은 기본으로 출력됨');
}
```

조건이 1일 때~

조건이 2일 때~

break?

?

```
[Running] node "d:\asdf\i.js"
조건은 1
```

break란?
반복문, switch등의
코드에서 빠져나오기
위해 사용된다

3

# 만약 break가 없다면?

```
let 조건 = 1;

switch (조건) {
    case 1:
        console.log('조건은 1');
        break;
    case 2:
        console.log('조건은 2');
        break;
    default:
        console.log('default은 기본으로 출력됨');
}
```

3

# 만약 break가 없다면?

```
[Running] node "d:\asdf\i.js"
조건은 1
조건은 2
default은 기본으로 출력됨
```

Q

# Quiz

변수 score가 90점 이상이면서 동시에 day가 80점 이상일 때만 우수 학생을, 아닐 경우 미해당을 출력하는 if문을 작성하세요.

let score = 92;

let day = 85;

Q

# Quiz

```
if ( score >= 90 && day >= 80) {
    console.log("우수 학생");
} else {
    console.log("미해당");
}
```

Q

# Quiz

switch case를 사용하여 month변수에 따라
계절을 출력하는 switch case문을 작성하세요.
(1 ~ 3이면 "봄", 4 ~ 6이면 "여름", 7 ~ 9이면 "가을",
10 ~ 12면 "겨울", 그 외의 숫자는 "알 수 없는 달"을 출력)

```
let month = 9;
```

Q

# Quiz

```
let month = 9;

switch (month) {
  case 1:
  case 2:
  case 3:
    console.log("봄");
    break; . . .

default:
    console.log("알 수 없는 달");
```

3

# 반복문

특정 조건이 만족될 때까지 코드 블록을 반복 실행하는 데 사용한다

대표적으로 for, while 등이 있다.

3

# FOR

3

# FOR

특정 조건이 거짓(false)이 될 때까지 코드 블록을 반복 실행하는 데 사용됩니다.

ex) for문,for in문, for of이 있습니다

3

# FOR문

```
for (초기화; 조건식; 증감식;) {
    //같은 코드를 여러 번 실행하는 것
}
```

초기값 설정 → 조건이 진실(true) → 코드실행
→ 값 증감 → 다시 조건확인

3

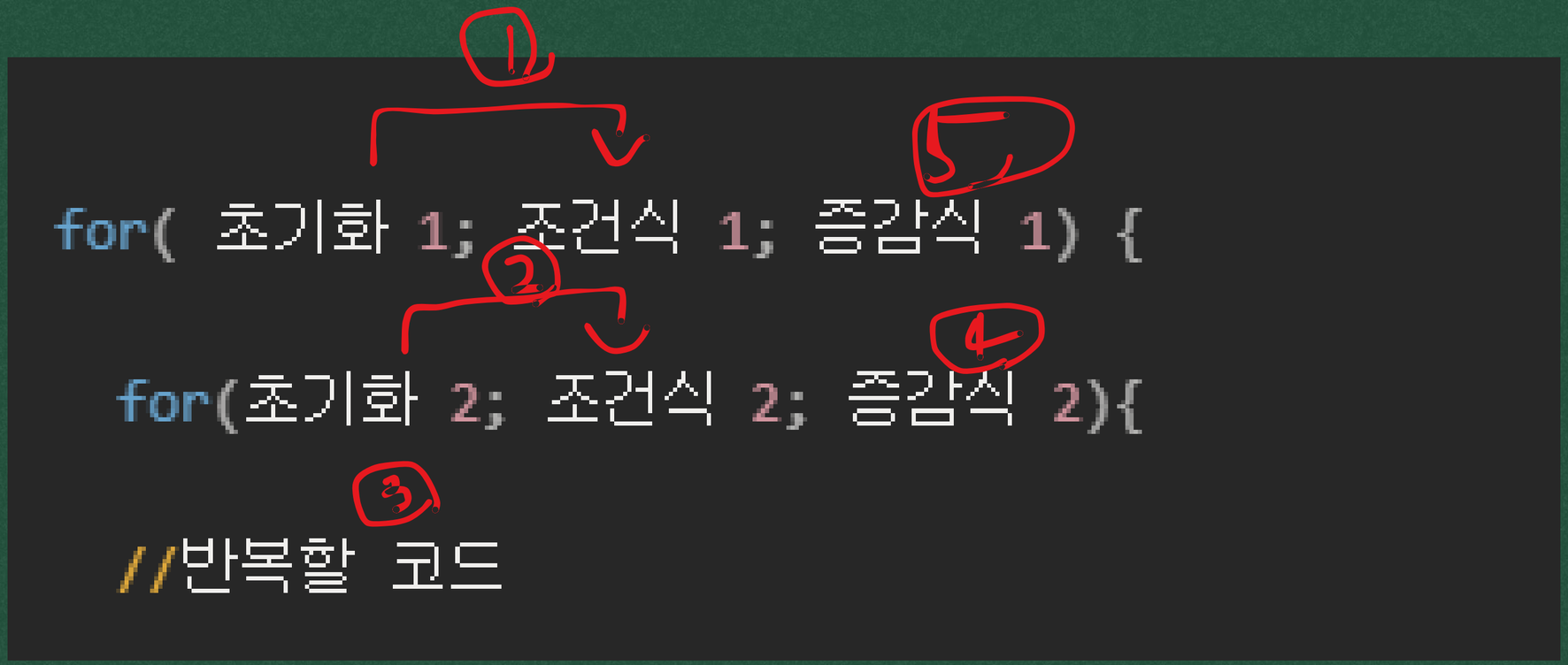
# 이중(중첩) FOR문

```
for(초기화 1; 조건식 1; 증감식 1) {
  for(초기화 2; 조건식 2; 증감식 2){
  //반복할 코드
  }
}
```

반복문 안에 또 다른 반복문

3

# 이중(중첩) FOR문



3

# 이중(중첩) FOR문

```
for (let ho
    for (le
        cor
    }
}
```

```
분입니다. );
```

```
현재 시각은 1시 10분입니다.
현재 시각은 1시 20분입니다.
현재 시각은 1시 30분입니다.
현재 시각은 2시 10분입니다.
현재 시각은 2시 20분입니다.
현재 시각은 2시 30분입니다.
```

3

# FOR IN

```
for (variable in object) {
    // 실행할 코드
}
//variable: 반복문 내에서 현재 순회 중인 속성(key)을 저장할 변수의 이름
//object: 반복하고자 하는 이름(key)으로 순회할 수 있는 객체
```

속성(key)으로 순회할 수 있는 객체를 위한 반복문

3

# FOR IN

```
const person = {
    name: "John",
    age: 30,
    occupation: "Developer"
} // 객체 리터럴로 정의된 일반 객체(Plain object)
    //-> 중괄호 {}를 사용해 만든, 오직 데이터만 담고 있는 가장 단순한 기본 상자

for (let key in person) {
    console.log(key); // 속성(key) 이름 출력
}
```

3

# FOR IN

```
const person = {
    name: "John"
    age: 30,
    occupation:
} // 객체 리터럴
    //-> 중괄호 {

for (let key in
    console.log(
}
```

상자

```
name
age
occupation
```

3

# FOR OF

```
for (variable of iterable) {
    // 반복할 코드
}

//variable: 반복문 내에서 현재 순회 중인 값(value)을 저장할 변수의 이름
//iterable: 반복하고자 하는 값(value)으로 순회할 수 있는 객체
```

값(value)으로 순회할 수 있는 객체를 위한 반복문

3

# FOR OF

```
const iterable = [1, 2, 3, 4, 5]; // 배열을 예시로 사용

for (const item of iterable) {
    console.log(item);
}
```

3

# FOR OF

```
const iterable = [1, 2,                    배열을 예시로 사용

for (const item of itera
    console.log(item);
}
```

```
1
2
3
4
5
```

3

# while

```
let count = 1; // 1. 초기값 설정

while (count <= 3) { // 2. 조건식
  console.log(count);

  count++; // 3. 조건 변경
}
```

Q

# Quiz

for문을 사용하여 1부터 10까지의 숫자 중
홀수만 골라서 한 줄씩 출력하세요.

```
for (let i = 1; i <= 10; i++) {
    if (i % 2 !== 0) {
        console.log(i);
    }
}
```

Q

# Quiz

while문을 사용하여 5부터 1까지 숫자를 거꾸로(5, 4, 3, 2, 1)
카운트다운하며 출력하는 코드를 작성하세요.

```
let num = 5;

while (num >= 1) {
    console.log(num);
    num--;
}
```