

SeSAC 도봉 1기 웹 풀스택 과정

JS

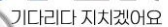
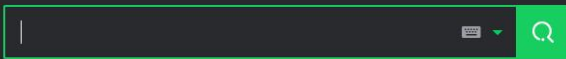




WITHOUT

JS





뉴스홀 · 연애 스포츠 경제

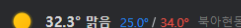
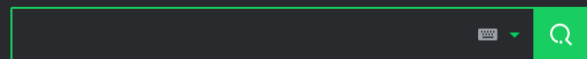
674 개의 글 | ⚙️ 관심주제 설정

NAVER 로그인

 [회원가입](#)



상품 쇼핑물 MEN



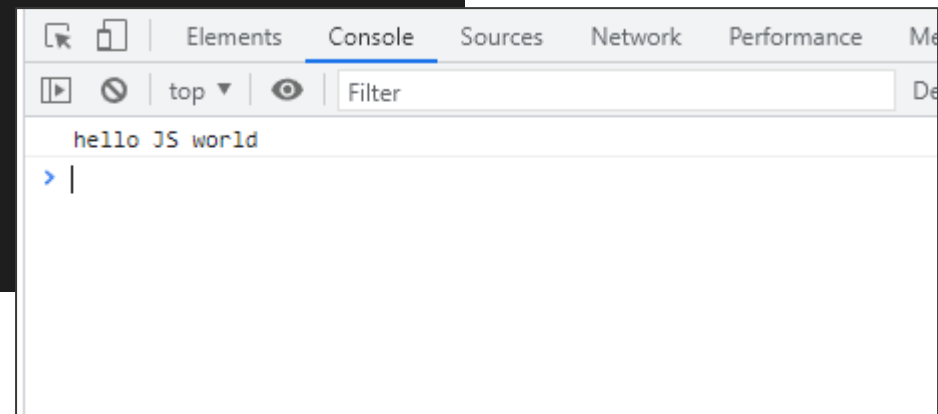
뉴스홀 · 연애 스포츠 경제

674개의 글 | 관심주제 설정

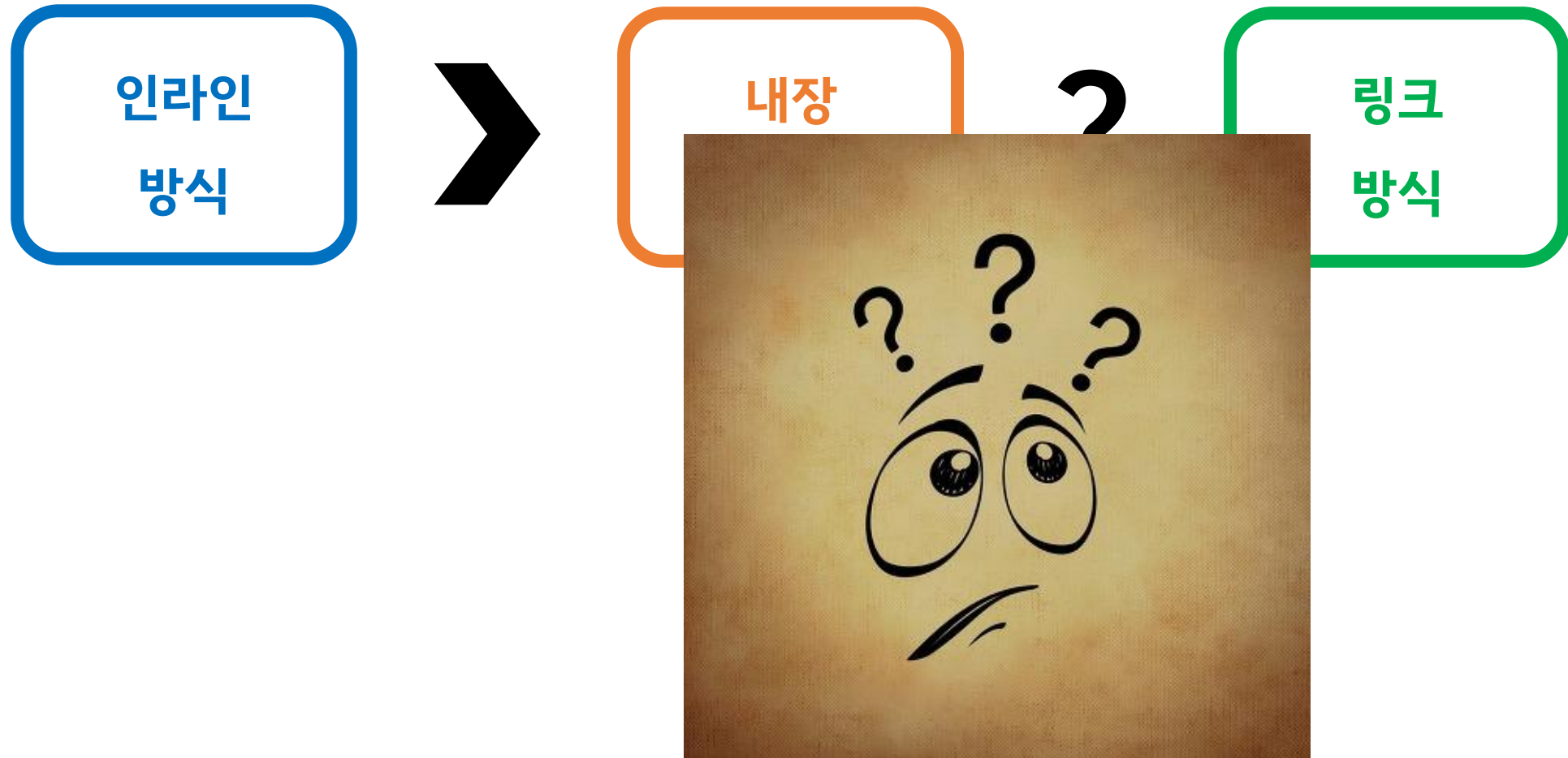
1

```
<!DOCTYPE html>
<html lang="ko">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>js start</title>
    <script>
      console.log('hello JS world');
    </script>
  </head>
  <body>

</body>
</html>
```



3가지 CSS 참조 방식





?



내장 방식!

```
<script>  
    alert("헤드 그런데 js 파일 링크 위");  
</script>
```

- 위치는 어디서나 사용이 가능합니다!
 - Head 태그 내부
 - Body 태그 내부
 - Head 와 body 사이
 - Body 아래 등

링크 방식!

- Java Script 파일을 따로 만들어서 링크하는 방식 like CSS

```
<script src="./index.js"></script>
```

- 위치는 어디서나 사용이 가능합니다!

- Head 태그 내부
- Body 태그 내부
- Head 와 body 사이
- Body 아래 등

각각의 장단점이 존재

- 내장 방식

- 간단하게 만들 수 있음
- 특정 페이지에서만 작동하는 기능일 경우 내장 방식으로만 따로 구현 가능

- 링크 방식

- JS 코드의 양이 많아지면 파일로 관리하는 편이 편함
- 같은 기능을 다른 페이지에서 사용하고 싶을 때 JS 파일 링크만 걸어서 사용 가능
- 유지 보수 용이성이 편리

일단 해봅시다!

- **Console.log("TEXT");**

- 우리의 디버깅 친구!
- 물론 이걸 쓰는건 안 좋은 방식입니다!

- **Alert("TEXT");**

- 화면에 뭐든 떠야 재미 있으니까!
- 자매품 confirm("TEXT");
 - 애는 취소 버튼이 있어요! → 값을 리턴한다 & IF문에 사용 가능!

읽기 순서?

- CSS의 방식 또는 선언 위치에 따라서 읽기 순서가 달라졌지요?
- JS는 어떤지 확인해 봅시다!

Index.js

```
alert("링크방식");
```

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <script>
    alert("헤드 그런데 js 파일 링크 위")
  </script>
  <script src="./index.js"></script>
  <script>
    alert("헤드 그런데 js 파일 링크 아래")
  </script>
</head>
<script>
  alert("헤드랑 바디 사이");
</script>
<body>
  <h1>hello, Js World!</h1>
  <script>
    alert("바디 안쪽");
  </script>
</body>
<script>
  alert("바디 아래");
</script>
</html>
```

Index.html

읽기 순서!

- 말 그대로 읽히는 순서에 따라서 작동 합니다!!

JS 기초!

표기법

dash-case(kebab-case)

snake_case

camelCase

ParcelCase

**the quick brown fox jumps over
the lazy dog**

HTML

CSS

dash-case(kebab-case)

the-quick-brown-fox-jumps-over-the-lazy-dog

HTML

CSS

snake_case

the_quick_brown_fox_jumps_over_the_lazy_dog



camelCase



theQuickBrownFoxJumpsOverTheLazyDog

JS

PascalCase

TheQuickBrownFoxJumpsOverTheLazyDog

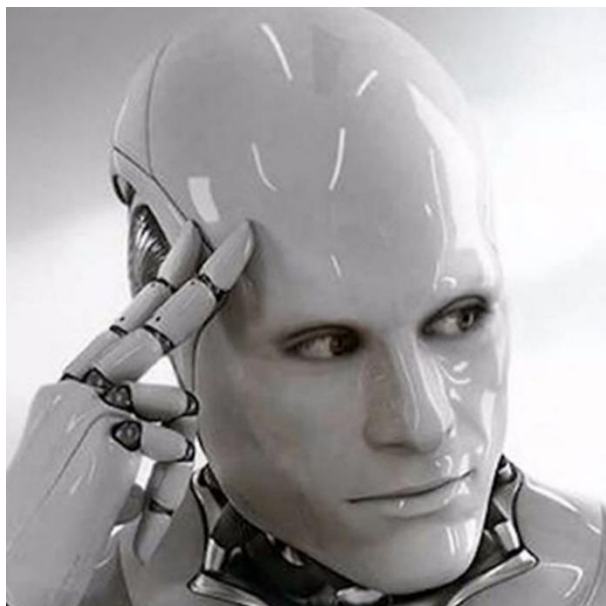
Zero-based Numbering

0 기반 번호 매기기!

특수한 경우를 제외하고 0부터 숫자를 시작합니다.



1 2 3 4 5 6 7 8 9 10 ~



0 1 2 3 4 5 6 7 8 9 ~


```
let fruits = ['Apple', 'Banana', 'Cherry'];
```

```
console.log(fruits[0]); // 'Apple'
```

```
console.log(fruits[1]); // 'Banana'
```

```
console.log(fruits[2]); // 'Cherry'
```

주석

Comments

```
// 한 줄 메모
```

```
/* 한 줄 메모 */
```

```
/**
```

```
 * 여러 줄
```

```
 * 메모1
```

```
 * 메모2
```

```
 */
```

JS 의

데이터 종류(자료형)

데이터 종류(자료형)

String

Number

Boolean

Undefined

Null

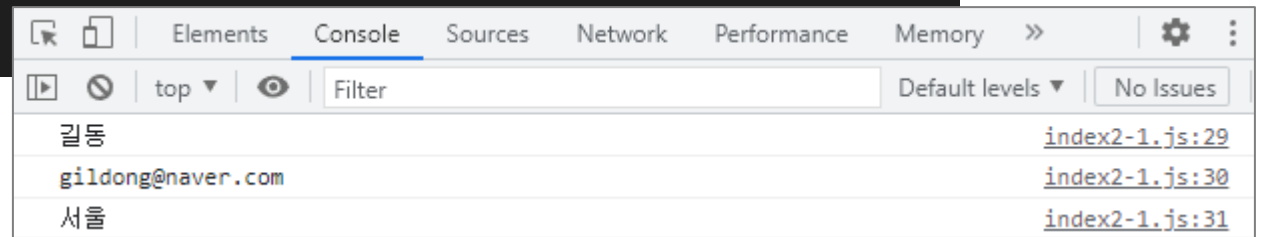
Object

Array

String 문자열

```
let myName = '길동';  
var email = 'gildong@naver.com';  
let city = '서울';  
console.log(myName);  
console.log(email);  
console.log(city);
```

String 문자형 데이터
따옴표를 사용



Number 숫자

```
// Number 숫자형 데이터  
// 정수 및 소수점 숫자를 나타냄  
let number = 123;  
let opacity = 0.7;  
  
console.log(number);  
console.log(opacity);
```

123

[index2-1.js:57](#)

0.7

[index2-1.js:58](#)

Boolean 참, 거짓 데이터

```
// Boolean 참, 거짓 데이터  
// true, false 두 가지 값만 가지는 데이터  
let checked = true;  
let isShow = false;  
  
console.log(checked);  
console.log(isShow);
```

true

[index2-1.js:69](#)

false

[index2-1.js:70](#)

undefined 미할당 데이터

```
// Undefined  
// 값이 할당되지 않은 상태를 표기  
let undef;  
let obj = {  
  abc: 123,  
};  
  
console.log(undef);  
console.log(obj.abc);  
console.log(obj.efg);
```

undefined

[index2-1.js:97](#)

123

[index2-1.js:98](#)

undefined

[index2-1.js:99](#)

Null 빈 데이터

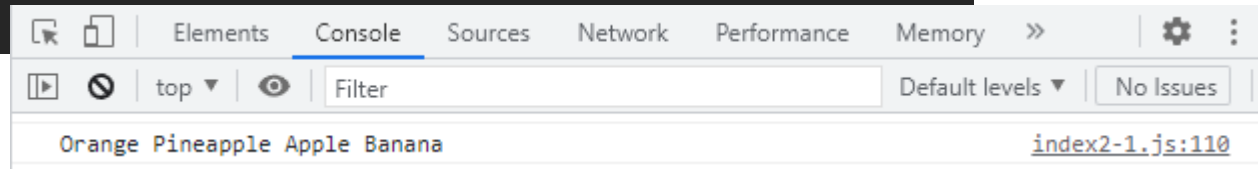
```
// Null  
// 어떤 값이 "의도적"으로 비어 있음을 의미할 때 사용  
let empty = null;
```

```
console.log(empty);
```

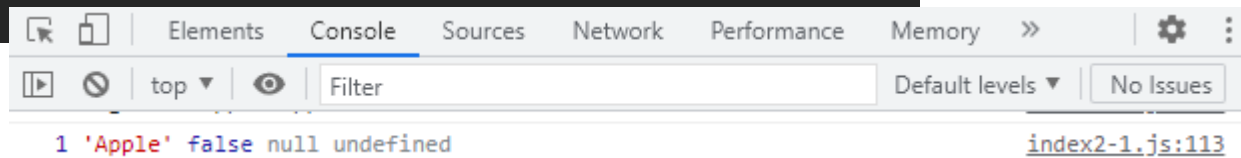
null

Array, 배열

```
// Array 배열 데이터  
// 여러 데이터를 순차적으로 저장  
let fruits = ["Orange", "Pineapple", "Apple", "Banana"];  
  
console.log(fruits[0], fruits[1], fruits[2], fruits[3]);
```



```
let data = [1, "Apple", false, null, undefined];  
  
console.log(data[0], data[1], data[2], data[3], data[4]);
```



Object, 데이터 꾸러미

```
let cat = {  
  name: '나비',  
  age: 1,  
  isCute : true,  
  mew: function () {  
    return '냐옹';  
  },  
};
```

```
console.log(cat);  
console.log(cat.name);  
console.log(cat.age);  
console.log(cat.mew());
```

```
▼ {name: '나비', age: 1, isCute: true, mew: f} ⓘ  
  age: 1  
  isCute: true  
  ▼ mew: f ()  
    arguments: null  
    caller: null  
    length: 0  
    name: "mew"  
    ▶ prototype: {constructor: f}  
      [[FunctionLocation]]: index2-1.js:172  
      ▶ [[Prototype]]: f ()  
      ▶ [[Scopes]]: Scopes[2]  
    name: "나비"  
    ▶ [[Prototype]]: Object
```

나비

1

냐옹

실습

- 각자를 소개할 수 있는 Object 형태의 변수 선언
- 지금 까지 배운 데이터를 사용하여 자신을 소개하는 console.log 만들기!

typeof

자료형을 알려주는 **typeof**

- 해당 자료형이 어떤 것인지 알려주는 착한 친구입니다!

```
console.log(typeof "안녕");  
console.log(typeof 3);  
console.log(typeof null);
```

string

number

object

실습

- **Typeof** 를 사용해서 아래의 문구를 만드세요
- “” 안의 문구는 **typeof** 의 결과값으로 출력해야 합니다.
- 출력 문구
 - “**number**” isn’t “**string**” data type.
 - **typeof** 를 **boolean** 이나 **null** 에 사용하면, “**object**” 결과를 얻을 수 있습니다.

형 변환

성적을 구하는 프로그램 만들기

```
let mathScore = prompt("수학 점수를 입력 하세요");  
let engScore = prompt("영어 점수를 입력 하세요");  
  
let avg = (mathScore + engScore) / 2;  
  
console.log(avg);
```

- 그런데 결과값이 좀 이상하죠?
- **Prompt** 로 입력 받은 값은 “문자”로 저장이 됩니다!
- “80” + “50” = “8050” → “8050” / 2 → 4025

JS의 자동 형변환!

- 처음에는 편할 수도 있지만 큰 문제를 일으키게 됩니다.
- 방금도 Error 가 났으면 바로 문제를 수정 했겠지만, Error 가 뜨지 않고 프로그램이 구동이 되었죠!
- 지금은 작은 프로그램이라 문제가 없었지만, 프로그램이 더 커진다면 의도하지 않았지만 정말 중대한 문제를 일으킬 수도 있습니다.
 - 만약, 비트코인 거래 사이트라면?

명시적 형변환

- 자동 형변환에 의존 하지 않고 개발자가 직접 형 변환을 시키는 것!
- 문자로 변환 → **String()**;
- 숫자로 변환 → **Number()**;

```
// 문자 데이터로 변환
let num = 123;
num = String(num);
console.log(typeof num);

// 숫자 데이터로 변환
num = Number(num);
console.log(typeof num);

console.log(Number("abcdefg"));
```

string

number

NaN

실습!

- 변수 `mathScore`, `engScore` 를 만들어 주세요.
- `mathScore = "77"; engScore = "88";`
- 시험 점수 평균을 계산하여 `avgScore` 에 저장하고,
이를 `console.log(avgScore)`를 이용해서 콘솔창에서 확인하는
프로그램을 작성해보세요.
- 형 변환을 사용하여 평균 점수가 정확하게 나와야 합니다!

변수!

(variable)

변수

데이터를 저장하고 참조(사용)하는 데이터의 이름
var, let, const

~~var~~

외알되?



```
// var  
var name = "홍길동";  
var name = "나비";  
  
console.log(name);
```

나비


```
// let  
let name = "홍길동";  
let name = "나비";  
  
console.log(name);
```

❌ Uncaught SyntaxError: Identifier 'name' has index.js:78
already been declared (at index.js:78:5)

```
// var  
var name = "홍길동";  
var name = "나비";  
  
console.log(name);
```

나비

```
// const  
const name = "홍길동";  
const name = "나비";  
  
console.log(name);
```

❌ Uncaught SyntaxError: Identifier 'name' has index.js:78
already been declared (at index.js:78:5)

var 의 문제점

- 중간의 같은 이름의 변수를 다시 선언해도 기존의 변수에 덮어 씌움
- 변수를 선언 했다는 건 분명히 다른 데이터를 넣으려는 것인데, 그것을 기존의 데이터에 덮어 씌우면!? → 문제 발생!
- 그리고 변수가 {블록 단위} 에서 끝나는 것이 아니라, 자기 맘대로 전역으로 돌아다니고 영향력을 행사함 → 의도치 않은 문제 발생!
- 따라서 ES6 문법 부터는 var 대신 let 사용을 권장

변수, let

```
// 재사용이 가능!
```

```
// 변수 선언!
```

```
let a = 2;
```

```
let b = 5;
```

```
console.log(a + b); // 7
```

```
console.log(a - b); // -3
```

```
console.log(a * b); // 10
```

```
console.log(a / b); // 0.4
```

변수, let

```
// 값(데이터)의 재할당 가능!
```

```
let a = 12;
```

```
console.log(a); // 12
```

```
a = 999;
```

```
console.log(a); // 999
```

변수, const

```
// 값(데이터)의 재할당 불가!
```

```
const a = 12;
```

```
console.log(a); // 12
```

```
a = 999;
```

```
console.log(a); // TypeError: Assignment to constant variable.
```

변수 기본 규칙 1

- 변수 이름으로는 문자 / 숫자 / \$ / _ 만 사용 가능

```
// 변수 이름으로는 문자 / 숫자 / $ / _ 만 사용 가능
```

```
let myName = "유진형";      // O
let my$ = "null";           // O
let my_dream= "rich";       // O
let my-house = undefined;   // X
let my* = "KTH"             // X
```

변수 기본 규칙 2

- 첫 글자는 숫자가 될 수 없어요

```
// 첫 글자는 숫자가 될 수 없어요
```

```
let 1stName = "진형" // X  
let firstName = "진형" // O
```


변수 기본 규칙 3

- 예약어도 사용할 수 없어요

```
// 예약어도 사용할 수 없어요
```

```
let let = "let me use this!";  
let if = "if i can use this...";
```

```
let this = 'Hello!'; // SyntaxError  
let if = 123; // SyntaxError  
let break = true; // SyntaxError
```

```
✖ Uncaught SyntaxError: let is disallowed as a lexically bound name (at dom.js: dom.js:15  
15:5)
```

예약어

특별한 의미를 가지고 있어, 변수나 함수 이름 등으로 사용할 수 없는 단어
Reserved Word

break, case, catch, continue, default, delete, do, else, false, finally, for, function, if, in, instanceof, new, null, return, switch, this, throw, true, try, typeof, var, void, while, with, abstract, boolean, byte, char, class, const, debugger, double, enum, export, extends, final, float, goto, implements, import, int, interface, long, native, package, private, protected, public, short, static, super, synchronized, throws, transient, volatile, as, is, namespace, use, arguments, Array, Boolean, Date, decodeURI, decodeURIComponent, encodeURI, Error, escape, eval, EvalError, Function, Infinity, isFinite, isNaN, Math, NaN, Number, Object, parseFloat, parseInt, RangeError, ReferenceError, RegExp, String, SyntaxError, TypeError, undefined, unescape, URIError ...

변수 기본 규칙 4

- 변수 이름은 읽기 쉽도록 센스 있게!

```
// 변수 이름은 읽기 쉽도록 센스 있게!  
let a = 1;  
let b = "allie";  
  
let userNumber = 1;  
let userName = "allie";
```

변수 기본 규칙 5

- 상수는 대문자로 선언해서 다른 개발자도 알 수 있게 해주세요

```
// 상수는 대문자로 선언해서 다른 개발자도 알 수 있게 해주세요
```

```
const WIDTH = 1100;  
const LOL_TIER = "silver";
```

문자 + 변수를 동시에 쓰고 싶을 때!

- 메소드의 매개 변수로 넣어서 사용
 - `Console.log("문자", 변수, "문자");`
- + 연산자를 사용해서 변수를 문자로 변환 후 더하여 사용
 - `Console.log("문자" + 변수 + "문자");`
- 백틱 문자 사용
 - '문자를 쓰다가 변수를 쓰고 싶으면 `${variable}` 처럼 쓰면 됩니다'

기본 연산자

기본 연산자

- % 나머지 연산자

- 홀수 판단 : $\text{num} \% 2 == 1$ 이면 홀수
- 짝수 판단 : $\text{num} \% 2 == 0$ 이면 짝수

- ** 거듭 제곱

- ** 를 사용
- $2^{**} 3 = 8$
- $3^{**} 3 = 27$

연산자 줄여서 쓰기

- $\text{num} = \text{num} + 5 \rightarrow \text{num} += 5$
- $\text{num} = \text{num} - 5 \rightarrow \text{num} -= 5$
- $\text{num} = \text{num} * 5 \rightarrow \text{num} *= 5$
- $\text{num} = \text{num} / 5 \rightarrow \text{num} /= 5$

증가, 감소 연산자

- **Num++;**
- **Num--;**

```
let result1, result2;  
let num = 10, num2 = 10;  
  
result = num++;  
console.log(result);  
  
result2 = ++num2;  
console.log(result2);
```

10

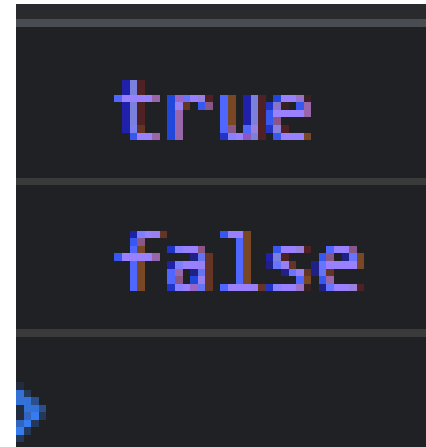
11

비교 연산자

일치 연산자(=== vs ==)

- 변수의 값 뿐만 아니라 자료형 까지도 비교!

```
let a = 1;  
let b = "1";  
  
// 비교 연산자  
console.log(a == b);  
  
// 일치 연산자  
console.log(a === b);
```



true

false

구문	의미
A==B	A와 B의 값이 같으면 true, 아니면 false
A===B	A와 B의 값과 data type이 같으면 true, 아니면 false
A!=B	A와 B의 값이 다르면 true, 아니면 false
A!==B	A와 B의 값과 data type이 다르면 true, 아니면 false

대소 비교

- 값의 크기를 비교하는 연산자 $<$, $>$, $<=$, $>=$

구문	의미
$A < B$	A보다 B가 크면 true, 아니면 false
$A > B$	A보다 B가 작으면 true, 아니면 false
$A <= B$	B가 A보다 크거나 같으면 true, 아니면 false
$A >= B$	B가 A보다 작거나 같으면 true, 아니면 false

논리 연산자

논리 연산자

|| (OR)

&& (AND)

! (NOT)

논리 연산자

|| (OR)

여러개 중 **하나라도 true** 면 **true**

즉, 모든값이 **false** 일때만 **false** 를 반환

논리 연산자

&& (AND)

모든값이 true 면 true

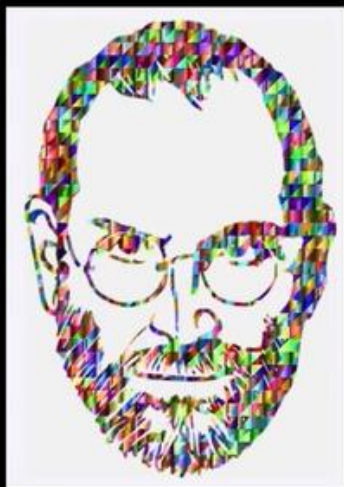
즉, 하나라도 false 면 false 를 반환

논리 연산자

! (NOT)

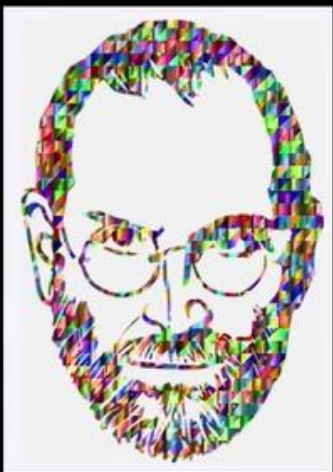
true 면 false

false 면 true



true

스티브 잡스는 **한국인** 이거나 **OR**, **남자**이다.



스티브 잡스는 **한국인** 이고 **AND**, **남자**이다.

평가

OR는 첫번째 **true**를 발견하는 즉시 평가를 멈춤

스티브 잡스는 **남자** 이거나 **OR**, **한국인** 이거나, **군인** 이거나..

평가

AND는 첫번째 **false**를 발견하는 즉시 평가를 멈춤

스티브 잡스는 **남자** 이고 **AND**, **한국인** 이며, **군인**인 동시에..

함수

Function!

함수

특정 동작(기능)을 수행하는 일부 코드의 집합(부분)
function

함수 선언문 vs 함수 표현식

```
function sayHello(){  
    console.log('Hello');  
}
```

함수 선언문

함수 선언문 vs 함수 표현식

```
let sayHello = function(){  
  console.log('Hello');  
}
```

함수 표현식

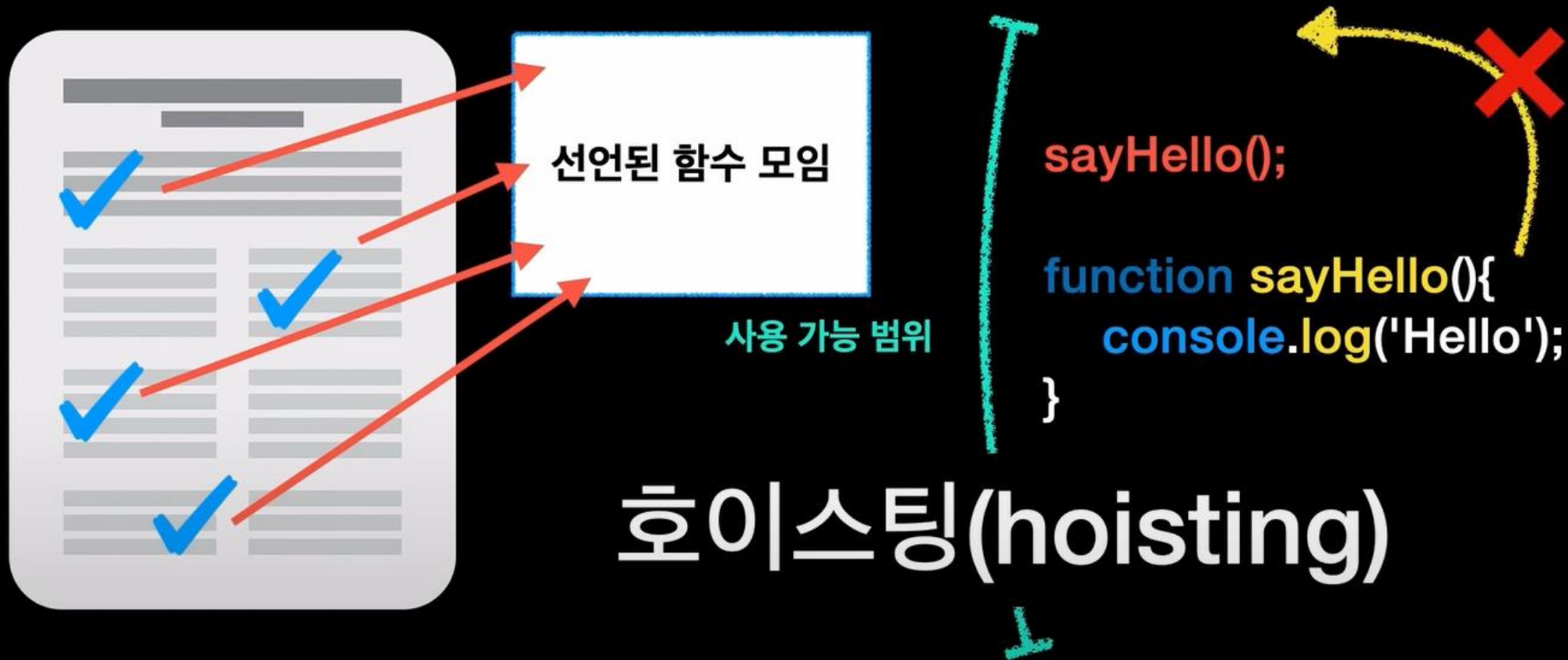
함수 선언문 : 어디서든 호출 가능

sayHello();



```
function sayHello(){  
    console.log('Hello');  
}
```

함수 선언문 : 어디서든 호출 가능



함수 표현식 : 코드에 도달하면 생성

1 ...

2 ...

3 `let sayHello = function(){` 생성
 `console.log('Hello');` 사용가능
 `}`

4 `sayHello();`

화살표 함수(arrow function)

```
let add = function(num1, num2){  
    return num1 + num2;  
}
```

화살표 함수(arrow function)

```
let add = (num1, num2) => {  
  return num1 + num2;  
}
```

화살표 함수(arrow function)

```
let showError = () => {  
    alert('error!');  
}
```


함수의 세가지 선언 방법

```
// 함수 선언문
function sayHello(name) {
    console.log(`Hello, ${name}`);
}

// 함수 표현식
let sayHello = function (name) {
    console.log(`Hello, ${name}`);
}

// 화살표 함수
let sayHello = (name) => {
    console.log(`Hello, ${name}`);
}
```