

JavaScript 선행 학습

VS Code에서 main.js 연결 테스트!

표기법

dash-case(kebab-case)

snake_case

camelCase

ParcelCase

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.


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

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

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

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

```
console.log(new Date('2021-01-30').getDay()); // 6, 토요일
```

```
console.log(new Date('2021-01-31').getDay()); // 0, 일요일
```

```
console.log(new Date('2021-02-01').getDay()); // 1, 월요일
```

주석

Comments

데이터 종류(자료형)

String

Number

Boolean

Undefined

Null

Object

Array

```
let a = 123;  
const n = obj.name;  
    document.querySelector('.data-abc')  
    { name: 'Heropy', age: 85 }  
function mount(params) {  
    return this;  
}
```



```
// String(문자 데이터)
```

```
// 따옴표를 사용합니다.
```

```
let myName = "HEROPY";
```

```
let email = 'thesecon@gmail.com';
```

```
let hello = `Hello ${myName}?!` 🗨️
```

```
console.log(myName); // HEROPY
```

```
console.log(email); // thesecon@gmail.com
```

```
console.log(hello); // Hello HEROPY?!
```

```
// Number(숫자 데이터)  
// 정수 및 부동소수점 숫자를 나타냅니다.  
let number = 123;  
let opacity = 1.57;  
  
console.log(number); // 123  
console.log(opacity); // 1.57
```

```
// Boolean(불린 데이터)  
// true, false 두 가지 값밖에 없는 논리 데이터입니다.  
let checked = true;  
let isShow = false;  
  
console.log(checked); // true  
console.log(isShow); // false
```



```
// Undefined
```

```
// 값이 할당되지 않은 상태를 나타냅니다.
```

```
let undef; 
```

```
let obj =  { abc: 123 };  
           
```

```
console.log(undef); // undefined
```

```
console.log(obj.abc); // 123
```

```
console.log(obj.xyz); // undefined
```

```
// Null
```

```
// 어떤 값이 의도적으로 비어있음을 의미합니다.
```

```
let empty = null;
```



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

```
// Object(객체 데이터)
// 여러 데이터를 Key:Value 형태로 저장합니다. { }
let user = {
  // Key: Value,
  name: 'HEROPY',
  age: 85,
  isValid: true
};
```

```
console.log(user.name); // HEROPY
console.log(user.age); // 85
console.log(user.isValid); // true
```

```
// Array(배열 데이터)  
// 여러 데이터를 순차적으로 저장합니다. [ ]  
let fruits = ['Apple', 'Banana', 'Cherry'];  
  
console.log(fruits[0]); // 'Apple'  
console.log(fruits[1]); // 'Banana'  
console.log(fruits[2]); // 'Cherry'
```

변수

데이터를 저장하고 참조(사용)하는 데이터의 이름

`var, let, const`

```
// 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
```

// 값(데이터)의 재할당 가능!

```
let a = 12;
```

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

```
a = 999; 
```

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

// 값(데이터)의 재할당 불가!

```
const a = 12;
```

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

```
a = 999;
```

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


예약어

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

Reserved Word

```
let this = 'Hello!'; // SyntaxError
```

```
let if = 123; // SyntaxError
```

```
let break = true; // SyntaxError
```

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 ...

함수

특정 동작(기능)을 수행하는 일부 코드의 집합(부분)

function

```
// 함수 선언
```

```
function helloFunc() {
```

```
    // 실행 코드
```

```
    console.log(1234);
```

```
}
```

```
// 함수 호출
```

```
helloFunc(); // 1234
```

```
function returnFunc() {  
    return 123;  
}
```

```
let a = returnFunc();
```

```
console.log(a); // 123
```

```
// 함수 선언!
```

```
function sum(a, b) { // a와 b는 매개변수(Parameters)  
    return a + b;  
}
```

```
// 재사용!
```

```
let a = sum(1, 2); // 1과 2는 인수(Arguments)
```

```
let b = sum(7, 12);
```

```
let c = sum(2, 4);
```

```
console.log(a, b, c); // 3, 19, 6
```

```
// 기명(이름이 있는) 함수  
// 함수 선언!  
function hello() {  
    console.log('Hello~');  
}
```

```
// 익명(이름이 없는) 함수  
// 함수 표현! 📄  
let world = function () {  
    console.log('World~');  
}
```

```
// 함수 호출!  
hello(); // Hello~  
world(); // World~
```



```
// 객체 데이터
```

```
const heropy = {  
  name: 'HEROPY',
```

```
  age: 85,
```

```
  // 메소드(Method)
```

```
  getName: function () {  
    return this.name;
```

```
  }
```

```
};
```

```
const hisName = heropy.getName();
```

```
console.log(hisName); // HEROPY
```

```
// 혹은
```

```
console.log(heropy.getName()); // HEROPY
```

조건문

조건에 따라 실행되는 코드(Truthy, Falsy)에 따라 다른 코드를 실행하는 구문
if, else

```
let isShow = true;
let checked = false;

if (isShow) {
    console.log( 'Show! ' ); // Show!
}

if (checked) {
    console.log( 'Checked! ' );
}
```

```
let isShow = true;
```

```
if (isShow) {  
    console.log( 'Show! ' );  
} else {  
    console.log( 'Hide? ' );  
}
```

DOM API

Document Object Model, Application Programming Interface



```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Document</title>
  <script src="./main.js"></script>
</head>
<body>
  <div class="box">Box!!</div>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Document</title>
</head>
<body>
  <div class="box">Box!!</div>
  <script src="./main.js"></script>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Document</title>
  <script defer src="./main.js"></script>
</head>
<body>
  <div class="box">Box!!</div>
</body>
</html>
```


// HTML 요소(Element) 1개 검색/찾기

```
const boxEl = document.querySelector( '.box' );
```



// HTML 요소에 적용할 수 있는 메소드!

```
boxEl.addEventListener();
```

// 인수(Arguments)를 추가 가능!

```
boxEl.addEventListener(1, 2);
```

// 1 - 이벤트(Event, 상황)

```
boxEl.addEventListener('click', 2);
```



// 2 - 핸들러(Handler, 실행할 함수)

```
boxEl.addEventListener('click', function () {  
    console.log('Click~!');  
});
```

// HTML 요소(Element) 검색/찾기

```
const boxEl = document.querySelector( '.box' );
```

// 요소의 클래스 정보 객체 활용!

```
boxEl.classList.add( 'active' );
```

```
let isContains = boxEl.classList.contains( 'active' );
```

```
console.log( isContains ); // true
```

```
boxEl.classList.remove( 'active' );
```

```
isContains = boxEl.classList.contains( 'active' );
```

```
console.log( isContains ); // false
```

```
// HTML 요소(Element) 모두 검색/찾기
const boxEls = document.querySelectorAll('.box');
console.log(boxEls);
```



```
// 찾은 요소들 반복해서 함수 실행!
// 익명 함수를 인수로 추가!
boxEls.forEach(function () {});
```

```
// 첫 번째 매개변수(boxEl): 반복 중인 요소.
// 두 번째 매개변수(index): 반복 중인 번호
boxEls.forEach(function (boxEl, index) {});
```

```
// 출력!
boxEls.forEach(function (boxEl, index) {
    boxEl.classList.add(`order-${index + 1}`);
    console.log(index, boxEl);
});
```

```
const boxEl = document.querySelector( '.box' );
```

```
// Getter, 값을 얻는 용도
```

```
console.log(boxEl.textContent); // Box!!
```

```
// Setter, 값을 지정하는 용도
```

```
boxEl.textContent = 'HEROPY?!';
```

```
console.log(boxEl.textContent); // HEROPY?!  

```

메소드 체이닝

Method Chaining



```
const a = 'Hello~';
```

```
// split: 문자를 인수 기준으로 쪼개서 배열로 반환.
```

```
// reverse: 배열을 뒤집기.
```

```
// join: 배열을 인수 기준으로 문자로 병합해 반환.
```

```
const b = a.split('').reverse().join(''); // 메소드 체이닝...
```

```
console.log(a); // Hello~
```

```
console.log(b); // ~olleH
```

Q.

The quick brown fox

위 문장을 camelCase(낙타 표기법)로 작성하시오!

A.

theQuickBrownFox

Q.

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

위 데이터를 활용해 'Banana'를 **콘솔 출력**하시오!

A.

```
console.log(fruits[1]);
```

Q.

불린 데이터(Boolean)에서
거짓을 의미하는 데이터는?

A.

false

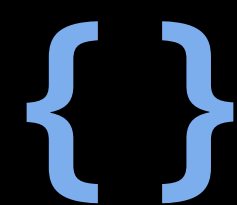
Q.

'값이 의도적으로 비어있음'을 의미하는 데이터는?

A.

null

Q.



위 데이터의 종류는?

A.

Object(객체 데이터)

Q.

```
let obj = { abc: 123 };  
console.log(obj.xyz);
```

위 코드를 통해 **콘솔 출력될 값**(데이터)은?

A.

undefined

Q.

값(데이터)을 재할당할 수 없는
변수 선언 키워드는?

A.

const

Q.

함수에서 값(데이터)을 반환하기 위해
사용하는 키워드는?

A.

return

Q.

`sum(2, 4);`

위 함수 호출에서 2, 4를 무엇이라 하는가?

A.

인수(Arguments)

Q.

```
function sum(a, b) {  
    return a + b;  
}
```

위 함수 선언의 **a, b**와 같이,

함수 호출에서 전달받은 **인수**를

함수 내부로 **전달**하기 위한 **변수**를 무엇이라 하는가?

A.

매개변수(Parameters)

Q.

이름이 없는 함수를 무엇이라 하는가?

A.

익명 함수(Anonymous Function)

Q.

hello 이름의 함수 표현을 작성하고 호출하시오!

A.

```
const hello = function () {};  
hello();
```

Q.

```
const user = {  
  getName: function () {}  
}
```

위 코드의 `getName`과 같이,
함수가 할당된 객체 데이터의
속성(Property)을 무엇이라 하는가?

A.

메소드(Method)

Q.

조건이 참(true)인 조건문을 작성하시오!

A.

```
if (true) { }
```

Q.

가져온 JS 파일을 HTML 문서 분석 이후에 실행하도록
지시하는 HTML 속성(Attribute)은?

A.

defer

Q.

```
<div class="box">Box!!</div>
```

위 HTML 요소의 내용(Content)을 콘솔 출력하시오!

A.

```
const boxEl = document.querySelector('.box');  
console.log(boxEl.textContent);
```

Q.

값(데이터)을 재할당할 목적의
변수 선언 키워드는?

A.

let

Q.

```
const boxEl = document.querySelector('.box');
```

위 코드의 `boxEl` 요소에 **클릭(Click) 이벤트**를 **추가**해,
클릭 시 **'Hello~'**를 **콘솔 출력**하시오!

A.

```
boxEl.addEventListener('click', function () {  
    console.log('Hello~');  
});
```

Q.

```
<div>1</div>
```

```
<div>2</div>
```

위 2개의 DIV 요소에 JS로
class="hello"를 추가하시오!

A.

```
const divEls = document.querySelectorAll('div');  
divEls.forEach(function (divEl) {  
    divEl.classList.add('hello');  
});
```

Q.

```
'HEROPY'.split(' ').reverse().join(' ');
```

위와 같이, 메소드를 이어 작성하는 방법을
무엇이라 하는가?

A.

메소드 체이닝(Method Chaining)

Q.

```
const boxEl = document.querySelector('.box');
```

위 코드의 `boxEl` 요소에 HTML 클래스 속성의
값으로 `'active'`가 포함되어 있으면,
`'포함됨!'`을 **콘솔 출력**하시오!

A.

```
if (boxEl.classList.contains('active')) {  
    console.log('포함됨!');  
}
```