4_JS(ES6)_PART_02

> - let, const, var
> - ⭐연산자

1. 변수를 선언하는 세가지 방법

let	const	var
ES6 이후	ES6이후	ES5 이전
변수	상수(사용 권장)	변수 (사용 권장 X)
블록 스코프	블록 스코프	함수 스코프

(1) let

```
// 1. let, ES6, block scope
let age = 30;
age = 30;
console.log(age);

let howiswhethertoday;
console.log(howiswhethertoday);
//undefined
howiswhethertoday = 20;
console.log(howiswhethertoday);
```

(2) const

```
// 2. const ES6
const birthYear = 2009;

// 2-1) birthYear = 2990;
// TypeError: Assignment to constant variable. at Object.<anonymous>
// 타입에러 : 상수화된 변수에 할당이 되었다.
// 상수 : 변하지 않는 값
// 즉, const를 올바른 문법으로 사용하지 않았기 때문에 에러 발생

// 2-2) const defaultBirthYear;
// let은 처음에 값을 할당해도 되고 나중에 할당해도 상관없지만
// const 는 반드시 처음에 어떤 값이 할당이 되어야 한다.
// 또한 2-1 의 설명처럼 그 값을 바꿔 줄 수 없다
```

(3) var

```
// 3. var ES5, function scope
var address = "Seoul";
address = "Busan";
```

2. ⭐연산자

> - 연산을 가능하게 하는 기호
> - 종류가 더 많지만 자주 쓰이는 5가지
>
> 1. 할당 연산자
> 2. 비교 연산자
> 3. 산술 연산자
> 4. 논리 연산자
> 5. 삼항 연산자

(1) 할당연산자 (Assignment Operators)

```
const myAppleCount = 3;
const yourAppleCount = 49;
const totalAppleCount = myAppleCount + yourAppleCount;

let AppleCount = 4;
// 단축 연산자
AppleCount += 20;
console.log(AppleCount);

appleCount -= 65;

appleCount *= 3;

appleCount /= 3;
console.log(appleCount);
```

■ 타입 변환과 타입 강제

(1) String으로 선언된 1997 값을 Number로 wrapper 한다.

```
console.log(Number(birthYear));
console.log(Number(birthYear) + 20);
```

(2) 만약 Number로 바뀔 수 없는 값이라면?

```
console.log(Number('Hello'));
```

> Nan 이라는 결과값이 나온다 (오류는 아님)

(3) 숫자를 문자열로 wrapper 한다.

```
console.log(String(30), 30);
```

> 30 전자 string, 후자 30 숫자 타입(각각의 타입이 다르다.)

(4) 자신의 타입을 버리고 다른 타입으로 변환 = 타입 강제 변환

```
console.log("나는 " + 40 + "살 입니다.");
```

> string 이지만 40은 number이다.
>
> 하지만 결과를 보면 string으로 변환이 된다. => 타입 강제

(5) 다른 타입을 가진 두개를 연산을 하려면?

```
console.log("20" - 5.5);
```

> 결과값 : 14.5
>
> 자바스크립트에서 이러한 연산도 가능하다 (타입변환과 타입강제 때문)

---

(2) 비교연산자 (Comparison Operators)

> 협업에서는 `===` 보다 `==` 을 권장한다.

```
const stringThree = '3';
const numberThree = 3;
```

```
console.log(stringThree == numberThree); //true
```

> - `==` : 단순하게 값이 같은지에 집중을 한다 (타입 신경x)

```
console.log(stringThree === numberThree); //false
```

> - `===` : 값과 타입이 모두 같아야 한다 (타입 신경 o)

```
const stringIsTrue = 'true';
const booleanIsTrue = true;
const numberIsTrue = 1;s
```

- 결과 (1)

```
console.log(stringIsTrue === booleanIsTrue); //false
```

```
console.log(booleanIsTrue == stringIsTrue); // false
```

> 🤔 `console.log(booleanIsTrue == stringIsTrue);` 의 결과값이 false 값이 나온 이유
>
> : boolean 이나 string 타입은 number로 변환이 된다.
>
> : 따라서 booleanIsTrue의 true 는 number 1로 변환이 되고
>
> : stringIsTrue 의 'true'는 Nan(not a number)으로 변환이 된다.
>
> : 그래서 false 결과 값이 나오는 것이다.

- 결과 (2)

```
console.log(booleanIsTrue == numberIsTrue); // true
```

■ **Truthy vs Falsy**

Truthy	**Falsy**
Boolean에서 true로 인식	Boolean에서 false로 인식
Falsy한 값을 빼고 전부	false
"0"	0, 0n
"false"	'', " "
[ ] / { }	Null / undefined / Nan

---

(3) 산술연산자 (Arithmetic Operators)

```
let num = 3;
num++;
num--;
console.log(num);
```

(4) 논리연산자 (Logical Operator)

```
const a1 = true && true; // true
const a2 = true && false // false
const a3 = false && true // false

const a4 = false && 3 == 4; // false
```

■ &&

(1) 왼쪽값이 true 일때

```
const a5 = 'cat' && 'dog'; // true
```

> 비교 연산자에서 양쪽에 불린이 아닌 다른 값으로 이루어져 있을 때는
>
> 왼쪽 값이 true이면 오른쪽에 있는 값을 바로 return 한다.
>
> 'dog' 뒤에 있는 연산자의 타입을 불린이 아닌 온전한 값인 타입을 유지한다.
>
> `console.log(a5);` 의 결과 값 : 'dog'

(2) 왼쪽 값이 false 일때

```
const a6 = false && 'cat'; // false
```

> 왼쪽 값이 false 일때 무조건 false 이다.

(3) 왼쪽 값이 true 이지만 오른쪽 값이 false 일때

```
const a7 = 'cat' && false; //false
```

■ ||

```
const o1 = true || true; // true
const o2 = false || false; // true
const o3 = true || false; // true
const o4 = false || 3 == 4; // false || false -> false
```

- 오른쪽 값이 true라면 왼쪽 값 return

```
const o5 = 'cat' || 'dog'; //'cat'
```

> or 연산자는 한쪽이 true 라면 true 값을 반환하되
>
> 반환되는 과정에서 형 변환을 하지 않고
>
> 그 값을 온전하게(=문자열이라면 문자열) return 한다.
>
> ```
> const o6 = false || 'cat'; //'cat'
> ```
>
> ```
> const o7 = 'cat' || false; // 'cat'
> ```

(5) 삼항연산자 (Conditional Operator)

```
const age = 17;
const AdultCheck = age >= 18 ? '성인' : '미성년'
```

> 다음과 같은 코드와 같은 뜻이다.
>
> ```
> if (age >= 18) return '성년';
> else return '미성년';
> ```

■ Summary

> - 변수를 선언하는 방법에는 let, const, var 가 있고 const를 권장
> - 연산자의 종류 5가지
> - Falsy한 값은 0, fasle, " ", undefined, null, NaN