6_JS(ES6)_PART_11

> - 객체 리터럴
> - 옵셔널 체이닝
> - map
> - set

1. 객체 리터널 (객체를 문자 그대로 표현할 수 있다.)

: ES6 버전에서 객체를 표현하는 방식의 변화

(1) 변수명과 객체 프로퍼티 같은 경우 중복 표현 생략

```
//Object Literal
const objectName = {
    a:1,
    b:'bb'
};

const outsideObject = {
    objectName: objectName
};

//ES6버전
const es6OutsideObject = {
    objectName
};

// key, value 모두 가지고 있을 경우 축약해서 사용 가능
// 코드는 짧으면 짧을수록 편리
```

(2) 개체 메서드 함수표현식 function 키워드 생략

```
const objetMethods = {
    method1 : function () {
        console.log('this is method');
    }
}
objectMethods.method1();

const es6ObjectMethods = {
    method1() {
         console.log('this is method');
    }
}

es6ObjectMethods.method1();

//짧은 코드를 선호
```

(3) 오브젝트 프로퍼티 명 안에 연산 가능

```
const arr = ['a', 'b', 'c'];
const objectWithOperation = {
    [arr[0]] : 'this is a'
    [`${1+4+'ha'}`] : 'this is 4ha'
}

console.log(objectWithOperation);
```

2. 옵셔널 체이닝 (optional chaining)

```
const weekdays = {
    mon: {
        open: 10,
        close: 22
    },
    tue: {
        open: 11,
        close: 20
    }
}
```

```
console.log(weekdays.wed.open);
//TypeError : Cannot read properies of undefinded(reading 'open')

//ES6 이전에는 어떻게 핸들링을 했을까?
if(weekdays.wed) console.log(weekdays.wed.open);
```

```
//옵셔널 체이닝
console.log(weekdays.wed?.open);
```

> `weekdays.wed` 앞 값이 있으면 `open` 을 참조해서 출력을 하고 `weekdays.wed` 값이 없으면 `undefinded` 출력

: `undefined`, `null` 에서는 어떤 프로퍼티를 참조할 수 없다. (TypeError)

: 이 에러를 방지하기 위해 만들어진 ES6 문법 옵셔널 체이닝이다.

```
const thisIsNull = null;
console.log(thisIsNull?.a);
```

```
objectMethods.method2();
//TypeError : objectMethods.method2 is not a function
```

```
objectMethods.method2.() ?? console.log('no method');
```

> 메서드가 없는 경우에는 `console.log('no method');` 오른쪽 으로 갈 수 있게 한다.

3. Map

: ES6 에서 추가된 KEY-VALUE 타입의 집합

: KEY 는 중복해서 들어갈 수 없다.

: 이터러블(iterable)_ 순회가능한 구조

```
//map
const thisIsMap = new Map();
thisIsMap.set(1, 'this is 1');
thisIsMap.set(true, 'this is boolean');
thisIsMap.set('aa', 'this is aa');
console.log(thisIsMap.get(1));
```

```
//iterable

for (const element of thisIsMap) {
    console.log(element);
}

//delete
thisIsMap.delete(1)
console.log(thisIsMap);
```

4. Set

: ES6에서 추가된 집합 객체

: 순회가 가능하고 값이 중복해서 들어갈 수 없음

: 이터러블(iterable)_ 순회가능한 구조

```
const thisIsSet = new Set();
thisIsSet.add('pizza');
thisIsSet.add('pasta');
thisIsSet.add(1);
thisIsSet.add(true)
console.log(thisIsSet);
// 한번 더 추가한다고 해도 변화가 없다.
thisIsSet.add('pizza')

//has
console.log(thisIsSet.has('pizza'));
console.log(thisIsSet.has('pizza2'));
```

```
for (const element of thisIsSet) {
    console.log(element);
}
```