6_JS(ES6)_PART_04

> - 배열
> - 배열의 메서드
> - 오브젝트
> - 오브젝트의 메서드

■ Array

> : 배열은 리스트와 비슷한 객체로서 순회와 변형 작업을 수행하는 메서드를 갖는다.
> : 배열은 요소 인덱스로 문자열을 사용할 수 없으며 무조건 정수만 허용한다.

*** 키(key)를 가지고 있지 않다.**
*** 순서를 고려한다.**
*** 인덱스로 값을 참조한다.**
const newArray = [1, 2, 3];

(1) 기본 구조

```
const fruit1 = 'apple';
const fruit2 = 'banana';
const fruit3 = 'kiwi';
const fruits = [fruit1, fruit2, fruit3];

console.log(fruits);
```

(2) 타입 상관 없다.

```
const diffTypeArray = ['hello', 23, true, null];
console.log(diffTypeArray);
```

(3) 인덱스 기반으로 참조하기

```
console.log(fruits[2]);
```

- **자주 사용하는 클래스 속성 `length`**

```
console.log(fruits.length);
```

- **만약 배열의 값을 바꾸고 싶다면?**

```
fruits[2] = 'mango';
```

> 🤔 const로 선언 될 때 값을 바꿀 수 없다고 했는데..?

> **원시타입 vs 객체 타입 =====> 서로 다르게 동작함을 다시 생각해보기**
>
> > array는 객체 타입에 속한다 따라서 값을 바꿀 수 있다.

- **만약 나이를 계산하고 싶다면? ⭐ `map`**

```
const birthYearArray = [1990, 1992, 1994];
const ageArray = birthYearArray.map(
    birthYearArray => 2023 - birthYearArray
);
console.log(ageArray);
```

> birthYearArray 출생년도 배열에서 나이를 구하는 공식은 동일하기에,
>
> 배열의 그 값을 같은 함수 로직으로 처리하여 return을 하고 싶다면
>
> map을 사용하면 된다.
>
> 순서대로 각각의 배열들의 원소들을 공식에 적용하여 처리한다.

(4) 자주 사용하는 메서드

```
const cars = ['hyundai', 'bmw', 'audi'];
```

4-(1) add element

```
cars.push('toyota');
cars.unshift(cars);
console.log(cars);
```

4-(2) remove element

```
cars.pop();
cars.shift();
console.log(cars);
```

4-(3) get index

```
console.log(cars.indexOf("bmw"));
```

> 같은 값이 배열 안에 있을 시, 앞에 있는 값을 반환한다.

```
console.log(cars.lastIndexOf("bmw"));
```

> 같은 값이 배열 안에 있을 시, 뒤에 있는 값을 반환한다.

4-(4) is contain

```
console.log(cars.includes("audi"));
console.log(cars.includes("volvo"));
```

> 배열에 찾고자 하는 값이 들어가 있는지 확인하고 싶을 때

4-(5) 여러개의 배열 연결

```
const array1 = [1, 2, 3];
const array2 = [4, 5, 6];
const combineArray = [...array1, ...array2];
console.log(combineArray);
```

■ Object

> : 키 값을 기본적으로 가진다. 다양한 키 모듬 및 더 복잡한 엔티티들을 저장하는데 사용된다.
>
> : 객체는 object() 생성자 또는 객체 초기자/리터럴 구문을 통해 생성할 수 있다.

*** 키(key)를 가지고 있다.**
*** 순서를 고려하지 않는다.**
*** 키로 값을 참조한다.**
const newObject = {a : 1, b : 2};

(1) 기본 구조

```
const profile = {
    firstName: "Kevin",
    lastName: "Kim",
    age: 30,
    job: 'engineer',
    isMarried: false,
};
console.log(profile);
console.log(profile.firstName);
console.log(profile['firstName']);
```

(2) 수정

```
profile.age = 31;
console.log(profile);
```

(3) 여러개의 오브젝트 연결

```
const obj1 = {
    name: 'Own',
    age: 56
};

const obj2 = {
    address: "seoul",
```

```
    job: "teacher"
};

const combineObject = {
    ...obj1,
    ...obt2
};
```

(4) 모든 키 값 확인하기

```
console.log(Object.keys(combineObject));

// map
console.log(Object.keys(combineObject)
    .map(key => combineObject[key]));
```

> 모든 키 값을 배열의 형태로 알 수 있다.

■ Summary

- 배열은 인덱스 기반의 자료구조이고, 순서를 고려한다.
- 오브젝트는 키 기빈의 자료구조이고, 순서를 고려하지 않는다.