6_JS(ES6)_PART_09

- 원시 타입과 객체 타입
- 얕은 복사와 깊은 복사

```
let myAddress = 'Seoul';
let oldMyAddress = myAddress;
myAddress = "Busan";
console.log(myAddress, oldMyAddress);
```

```
const profiel = {
    address: 'Seoul'
}
const afterProfile = profile;
afterProfile.address = 'Busan';
console.log(profile);
console.log(afterProfile);
```

> `address: 'Seoul'` 객체 값은 힙 메모리 공간에 저장이 된다.
>
> 그리고 `profile` 과 `afterProfile` 변수는 그 힙의 오브젝트가 저장되어 있는데,
>
> 그 메모리 주소값을 참조하고 있다.
>
> - `afterProfile` 과 `profile` 은 같은 메모리 값을 참조하고 있을까?
>
> > 같은 메모리 주소값을 참조하고 있다.
>
> - 결과값
>
> ```
> { address: 'Busan' }
> { address: 'Busan' }
> ```

```
const profile2 = {
    address: 'Seoul',
    family: ['Tony', 'Chris']
}

const profile2Copy = profile2;
// 이럴 경우에는 profile2의 주소를 복사를 한것 그럼, 값을 복사하고 싶은 경우에는?
// 다음과 같이 코드를 작성한다.
const profile2Copy = Object.assign({}, profile2);
```

> - assign 메서드
>
> : assign 메서드는 target과 source 2개의 인자를 받는다,
>
> `Object.assign({}, profile2);` 에서 첫번째 `{}` 가 target 이고 두번째 `profile2` 가 source 이다.

> 쉽게 설명하면, `{}` , `profile2` 이 2개를 합치는 것이다.
>
> 하지만 복사하는 경우는 첫번째에 값을 넣지 말고 두번째에 값을 넣어주면, `Object.assign` 를 통해서 빈객체(`{}`) 와 `profile2` 를 합치는 것이다.
>
> 그 결과가 `profile2Copy` 가 되는 것이다.

```
const profile2Copy = Object.assign({}, profile2)

profile2.address = 'Daegu';
console.log(profile2);
console.log(profile2Copy);
```

> - 결과값
>
> ```
> {address: 'Deage', family: ['Tony', 'Chris']}
> {address: 'Seoul', family: ['Tony', 'Chris']}
> ```
>
> `profile2Copy`값은 `profile2` 에 의존하지 않고 `Seoul` 값을 잘 가져오고 있음을 알 수 있다.
>
> ==========> 얕은 복사
>
> > 얕은 복사를 통해서 값을 복사해 올 수 있다.

🤔 왜 얕은 복사인가...?

- 배열도 객체 타입에 포함이 된다. 다음과 같은 코드를 살펴보며 왜 얕은 복사 인지 알아본다.

```
profile2Copy.family.push('Levin');

console.log(profile2);
console.log(profile2Copy);
```

> - 결과값
>
> ```
> {address: 'Deage', family: ['Tony', 'Chris', 'Levin']}
> {address: 'Seoul', family: ['Tony', 'Chris', 'Levin']}
> ```
>
> 얕은 복사는 기본적으로, 오브젝트에서 프로퍼티(property)의 첫번째 프로퍼티(property)값만 복사한다.
>
> 즉, 첫번째 프로퍼티(property)에 원시타입이 들어가는 경우라면 복사가 잘 되지만 첫번째 프로퍼티(property)에 오브젝트(객체 타입)이 들어가 있으면 그 값은 값 자체를 복사하는 것이 아닌 메모리 값을 복사하게 되는 것이다. 그렇기 때문에 중첩된 오브젝트 경우에는 얕은 복사로 복사를 하게 되면 객체들이 온전하게 값이 복사하지 않게 되지 않는 문제가 발생한다.
>
> 따라서 이러한 문제를 해결해서 나온 방안은 깊은 복사이다.
>
> 깊은 복사는 여러가지 할 수 있는 방법들이 많지만, `lodash` 라는 것을 활용하여 살펴볼 것이다.

👀 Lodash

> A modern JavaScript utilty library delivering modularity, performance & extras.
>
> : 자바스크립트에서 utilty성 모듈, 기능들 등을 Lodash 라는 것을 통해 작업을 하면, 쉽고 빠르게 작업이 가능하다.

```
_.cloneDeep(value)
```

> `.cloneDeep(value)` 메서드
>
> This method is like `_.clone` except that is recurslvely clones value.
>
> ```
> var objects = [{'a':1}, {'b':2}];
>
> var deep = _.cloneDeep(objects);
> console.log(deep[0] === objects[0]);
> //=> false
> ```

- 원시타입은 값을 콜 스택에, 객체 타입은 값을 힙에 저장
- 얕은 복사는 Object.assign()
- 깊은 복사는 lodash 사용