JS-중급 3

태그	
날짜	@2023년 5월 31일

객체 메소드(Object method), 계산된 프로퍼티(Computed property)

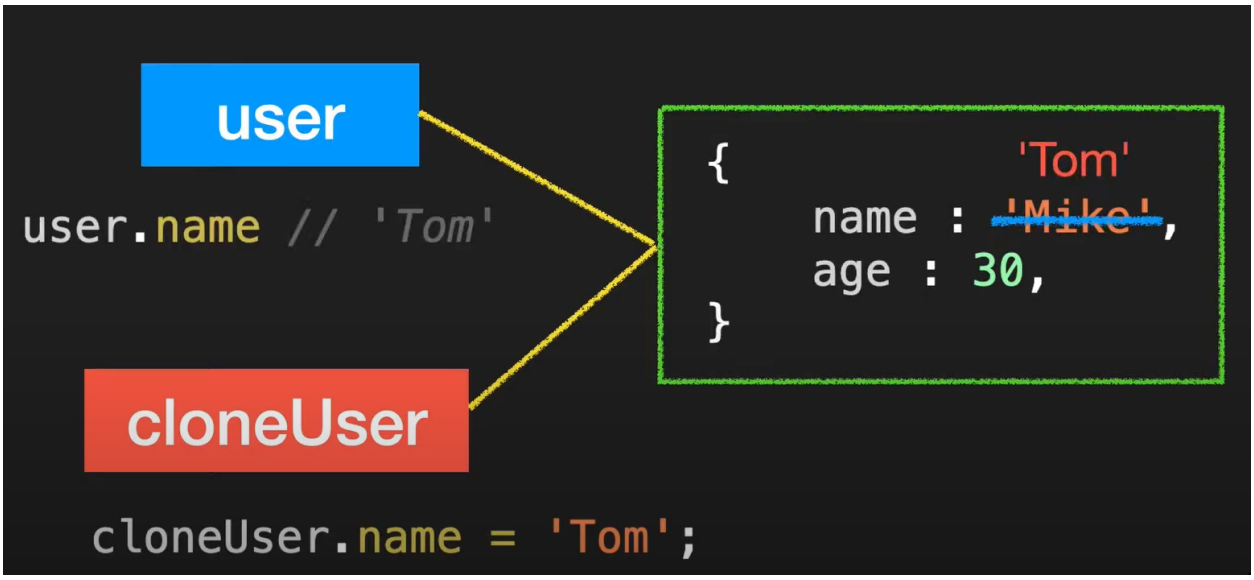
```
let a  = 'age';

const user = {
    name : 'Mike',
    age : 30,
}

age 키 대신 [a]를 사용해도 된다.

let a  = 'age';

const user = {
    name : 'Mike',
    age : [a],
}

대괄호로 묶어주면 a라는 문자열이 아니라 변수 a에 할당된
값이 들어간다. 이를 Computed property 라고 한다.

-------------------------------------------------

const user = {
    [1 + 4] : 5;
    ["안녕" + "하세요"] : "Hello"
}

이렇게 식 자체를 넣는 것도 가능하다.
```

```
예제

let n = "name";
```

```
let a = "age";

const user = {
  [n] : 'Mike',
  [a] : 30,
  [1 + 5] : 6
 };

console.log(user);

// [object Object]
{
  "6": 6,
  "name": "Mike",
  "age": 30
}
-----------------------------------------------------

function makeObj (key, val) {

   return {
     [key] : val,
   };
}

const obj = makeObj("이름", 33);

console.log(obj);

// [object Object]
{
  "이름": 33
}

// 어떤게 키가 될지 모르는 객체에 유용하다.
```

```
Methods

Object.assign()
Object.keys()
Object.values()
Object.entries()
Object.fromEntries()
-----------------------------------------------------

Object.assign() : 객체를 복제

const user = {
    name : 'Mike',
```

```
    age : 30,
}

const cloneUser = user; 를 만들어서 넣어주면 복제가 되는걸까?
아니다. 저 유저 변수에는 객체 자체가 들어가있는게 아니라
객체가 저장되어 있는 메모리 주소인 객체에 대한 참조값이 저장된다.
그러니 클론 유저를 만들어서 유저를 넣으면 객체가 복사되면서 들어가는게
아니라 그 참조값만 복사되는 것이다.
아래와 같은 모습이다.
```




```
cloneUser의 이름을 바꿨는데 user의 이름도 바뀌었다.
하나의 객체를 두 변수가 접근하고 있는것이다.

동일하게 복제하려면

const newUser = object.assign({}, user);

를 사용해야한다. 여기서 빈 객체({})는 초기 값이다.
두번째 매개변수(user)로 부터 들어온 객체들이 초기값에 병합된다.

{} + {name : 'Mike', age : 30 } =

이렇게 병합된다.

const newUser = Object.assign({},user);
newUser.name = 'Tom';
console.log(user.name); //'Mike'

이름을 바꿔도 user는 변함이 없다.
같은 객체가 아니다.
```

예제

```
const user = {
    name : 'Mike',
    age : 30,
};

const user2 = user;
user2.name = "Tom";

console.log(user);
console.log(user2);

// [object Object]
{
  "name": "Tom",
  "age": 30
}
// [object Object]
{
  "name": "Tom",
  "age": 30
}
```

하나의 객체를 보고 있기 때문에 둘다 값이 변경됨

그러므로 복사를 할때는 Object.assign() 을 사용

```
const user = {
    name : 'Mike',
    age : 30,
};

const user2 = Object.assign({},user);
user2.name = "Tom";

console.log(user);
console.log(user2);
// [object Object]
{
  "name": "Mike",
  "age": 30
}
// [object Object]
{
  "name": "Tom",
  "age": 30
}

-------------------------------------------------
Object.assign({gender : 'male' } , user);
```

같은 상황이라면
user를 병합하여

```
gender : 'male',
name : 'Mike',
age : 30,

이렇게 총 3개의 프로퍼티를 가지게 되는것이다.

만약 병합을 하는데 키값이 같다면 어떻게 될까?
Object.assign({name : 'Tom'}, user);

name : 'Tom',
name : 'Mike',
age : 30,

덮어쓰게 된다.

-------------------------------------------------

2개 이상의 객체도 합칠 수 있다.

const user = {
  name : 'Mike'
}

const info1 = {
  age : 30,
}

const info2 = {
  gender : 'male',
}

Object.assign(user,info1, info2)
각각 다른 객체가 있다면 info1과 info2를 user에 합친다.
```

```
Object.keys() : 키 배열 반환

const user = {
  name : 'Mike',
  age : 30,
  gender : 'male',
}

Object.keys(user);
// ["name", "age", "gender"]

객체 프로퍼티 키를 배열로 반환한다.

-------------------------------------------------
Object.values() : 값 배열 반환
```

```
const user = {
  name : 'Mike',
  age : 30,
  gender : 'male',
}

Object.values(user);
// ["Mike", 30 , "male"]

-----------------------------------------------------

Object.entries() : 키/값 배열 반환

const user = {
  name : 'Mike',
  age : 30,
  gender : 'male',
}

Object.entries(user);

[
  ["name", "Mike"],
  ["age", 30],
  ["gender", "male"]
]

배열안에 키와 값이 같이 들어있는 배열을 생성

-----------------------------------------------------

Object.fromEntries() : 키/값 배열을 객체로 생성
Object.entries()와 반대로 배열을 넣어주면 객체로 만들어준다.

const arr =
[
  ["name", "Mike"],
  ["age", 30],
  ["gender", "male"]
];

Object.fromEntries(arr);

{
  name : 'Mike',
  age : 30,
  gender : 'male',
}
```