ES6 객체와 배열

2023.06.13.

01. 구조 분해를 사용한 대입
02. 배열 구조 분해하기
03. 객체 리터럴 개선
04. 스프레드 연산자

01. 구조 분해를 사용한 대입

2023.06.13.

구조 분해(destructuring)를 사용하면 객체 안에 있는 필드 값을 원하는 변수에 대입할 수 있다.

sandwich를 분해하여 bread와 meat 필드를 같은 이름의 변수에 대입한다.

```
const sandwich = {
  bread:"더치 크런치",
  meat:"참치",
  cheese:"스위스",
  toppings: ["상추", "토마토", "머스타드"]
};

let {bread, meat} = sandwich;

console.log(bread, meat);
```

두 변수의 값은 sandwich의 같은 이름의 필드 값으로 초기화 되지만, 두 변수를 변경해도 원래의 필드 값은 바뀌지 않는다.

```
const sandwich = {
  bread:"더치 크런치",
  meat:"참치",
  cheese:"스위스",
  toppings: ["상추", "토마토", "머스타드"]
};

let {bread, meat} = sandwich;

bread = "마늘";
meat = "칠면조";

console.log(bread);
console.log(meat);

console.log(sandwich.bread, sandwich.meat);
```

```
const lordify = function(regularPerson) {
  console.log("켄터베리의 "+ regularPerson.firstName);
}

var regularPerson = {
  firstName:"유",
  lastName:"재석"
}

loadify(regularPerson); //켄터베리의 재석
```

객체의 필드에 접근하기 위한 점(.)을 사용하는 대신 구조 분해로 호출하기

```
const lordify = function({firstName}) {
  console.log("켄터베리의 "+ firstName);
}

var regularPerson = {
  firstName:"유",
  lastName:"재석"
}

loadify(regularPerson); //켄터베리의 재석
```

객체 안의 객체를 이용한 호출

```
const lordify = function({spouse: {firstName}}) {
  console.log("켄터베리의 "+ firstName);
}

var regularPerson = {
  firstName:"유",
  lastName:"재석",
  spouse: {
    firstName:"경은",
    lastName:"나"
  }
}

loadify(regularPerson); //켄터베리의 경은
```

02. 배열 구조 분해하기

2023.06.13.

배열의 첫 번째 요소를 변수에 대입하고 싶을 경우

```
const [firstAnimal] = ["캥거루", "웜벳", "코알라"];

console.log(furstAnimal); //캥거루
```

리스트 매칭(List matching) : 불필요한 값을 콤마(,)를 사용해 생략

```
const [,,thirdAnimal] = ["캥거루", "웜벳", "코알라"];

console.log(thirdAnimal); //코알라
```

03. 객체 리터럴 개선

2023.06.13.

객체 리터럴 개선(Object literal enhancement) : 구조 분해의 반대. 구조를 다시 만들어내는 과정 또는 내용을 한데 묶는 과정. 객체 리터럴을 사용하여 변수를 객체의 필드로 묶을 수 있다.

```
const name = "한라";
const elevation = 1950;

const funHike = {name, elevation};

console.log(funHike); //{name: "한라", elevation: 1950}
```

객체 리터럴 개선을 통해 메서드를 만드는 것도 가능하다.

```
const name = "한라";
const elevation = 1950;
const print = function() {
  console.log(this.name +"산의 높이는 "+ elevation +"m 입니다.");
}

const funHike = {name, elevation, print};

funHike.print(); //한라산의 높이는 1950m 입니다.
```

객체 메서드를 정의할 때 더 이상 function 키워드를 사용하지 않아도 된다.
옛날 방식

```
var name = "park";
var sound = "go fast";

var skier = {
  name:name,
  sound:sound,
  powerYell:function() {
    var yell = this.sound.toUpperCase();
    console.log(yell +" "+ yell +" "+ yell +"!!!");
  },
  speed:function(kmph) {
    this.speed = kmph;
    console.log("속력(km/h): "+ kmph);
  }
}

skier.powerYell();
skier.speed(56);
console.log(skier);
```

새로운 방식

```
var skier = {
  name,
  sound,
  powerYell() {
    var yell = this.sound.toUpperCase();
    console.log(yell +" "+ yell +" "+ yell +"!!!");
  },
  speed(kmph) {
    this.speed = kmph;
    console.log("속력(km/h): "+ kmph);
  }
}
```

04. 스프레드 연산자

2023.06.13.

스프레드 연산자 : 3개의 점(...)으로 이루어진 연산자로 몇 가지 다른 역할을 담당한다.

1. 배열의 내용을 조합한다.
두 개의 배열이 있는 경우, 두 배열의 모든 원소가 포함된 세 번째 배열을 만들 수 있다.

```
const peaks = ["대청봉", "중청봉", "소청봉"];
const canyons = ["천불동계곡", "가야동계곡"];
const seoraksan = [...peaks, ...canyons];

console.log(seoraksan.join(","));
```

사용례) peaks 배열에서 마지막 요소를 변수에 담고 싶다. Array.reverse()를 사용해 배열을 뒤집은 다음 구조 분해를 사용해 첫 번째 요소를 변수에 담을 수 있다.

```
const peaks = ["대청봉", "중청봉", "소청봉"];
const [last] = peaks.reverse();

console.log(last); //소청봉
console.log(peaks.join(",")); //소청봉, 중청봉, 대청봉
```

reverse()는 원본 배열을 뒤집는다. 스프레드 연산자를 사용하여 원본 배열을 뒤집지 않은 채 마지막 요소를 변수에 담을 수 있다.

```
const peaks = ["대청봉", "중청봉", "소청봉"];
const [last] = [...peaks].reverse();

console.log(last); //소청봉
console.log(peaks.join(",")); //대청봉, 중청봉, 소청봉
```

2. 배열의 나머지 원소 얻기

```
const lakes = ["경포호", "화진포", "송지호", "청초호"];
const [first, ...rest] = lakes;

console.log(rest.join(",")); //화진포, 송지호, 청초호
```

3. 레스트 파라메타(Rest parameter) : 함수의 인자를 배열로 받기

```
function directions(...args) {
  let [start, ...remaining] = args;
  let [finish, ...stops] = remaining.reverse();

  console.log(args.length +" 도시를 운행 합니다.");
  console.log(start +"에서 출발 합니다.");
  console.log("목적지는 "+ finish +"입니다.");
  console.log("중간에 "+ stops.length +" 군데를 들립니다.")
}

directions("서울", "수원", "천안", "대전", "대구", "부산");
```

4. 객체에서 스프레드 연산자 사용 : 객체 내용 조합

```
const morning = {
  breakfirst :"미역국",
  lunch:"삼치구이와 보리밥"
}

const dinner = "스테이크 정식";

const backpackingMeals = {
  ...morning,
  dinner
};

console.log(backpackingMeals);
//{"breakfirst": "미역국", "lunch": "삼치구이와 보리밥", "dinner": "스테이크 정식"}
```