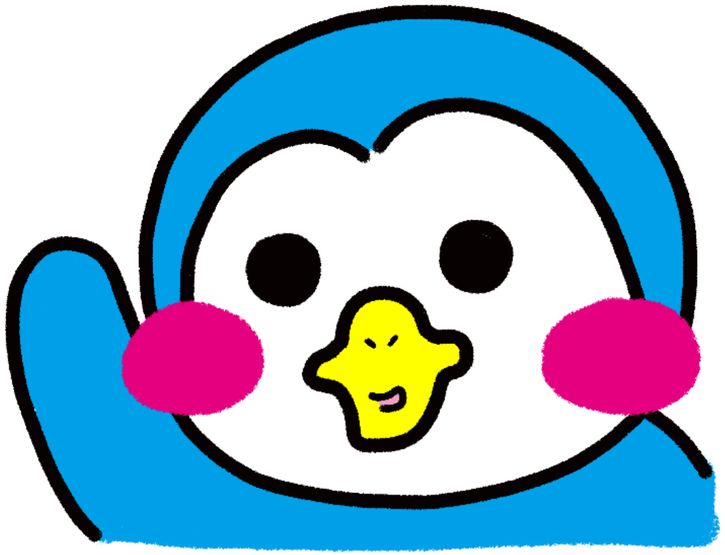
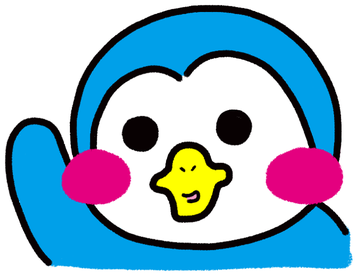
배열/객체를 해체한다



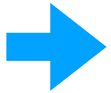
구조 분해 할당 & 객체 초기화

구조 분해 할당



구조 분해라고도 한다. 구조 분해란 배열이나 객체의 속성을 해체하여 그 값을 개별 변수에 담을 수 있게 하는 자바스크립트 표현식이다.

```
const cafe = {
  starbucks : "스타벅스",
  ediya : "이디야",
  pascucci : "파스쿠찌",
  coffeebean : "커피빈"
}
```

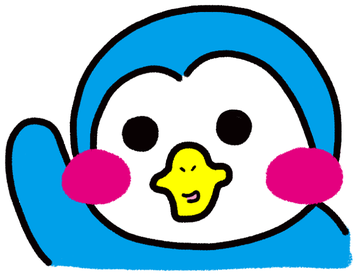


```
const star = cafe.starbucks
const edi = cafe.ediya
```

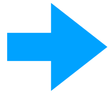
객체에서 스타벅스와 이디야의 값만 따로 복사한다.

=> 이렇게 하지 말고 Perl이나 Python의 방식을 따와서 복사해보자!

구조 분해 할당



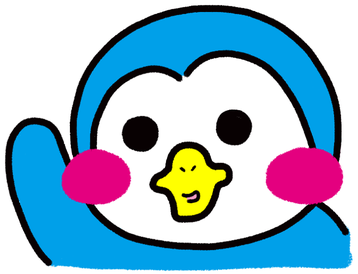
```
const cafe = {
  starbucks : "스타벅스",
  ediya : "이디야",
  pascucci : "파스쿠찌",
  coffeebean : "커피빈"
}
// 구조 분해를 통한 객체 값 복사!
const { pascucci, coffeebean } = cafe
console.log(pascucci, coffeebean)
```



파스쿠찌 커피빈

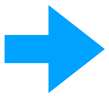
=> 출력된 변수 pascucci와 coffeebean은 객체 cafe와 별개이다!

배열의 구조 분해



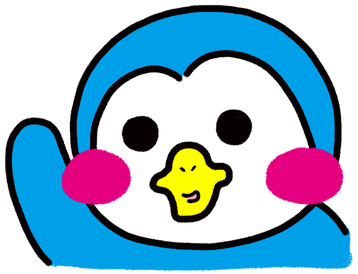
배열을 구조 분해하는 기본적인 방법은 다음과 같다.

```
const pet = ["강아지", "고양이", "햄스터"];

const [dog, cat, hamster] = pet;
console.log(dog);
console.log(cat);
console.log(hamster);
```



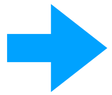
강아지
고양이
햄스터

선언과 초기화 분리



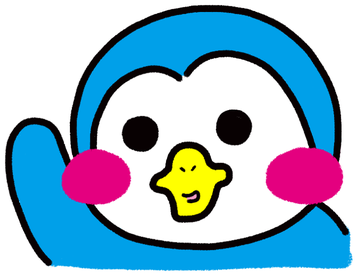
변수의 선언이 분리 되어도 구조 분해를 통해 값을 할당할 수 있다.

```
const pet = ["강아지", "고양이", "햄스터"];

let dog, cat;
[dog, cat] = pet;

console.log(dog);
console.log(cat);
```



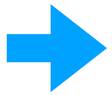
강 아 지
고 양 이

구조 분해 시 디폴트 선언



분해한 값이 undefined인 경우를 대비해 디폴트 값을 지정할 수도 있다.

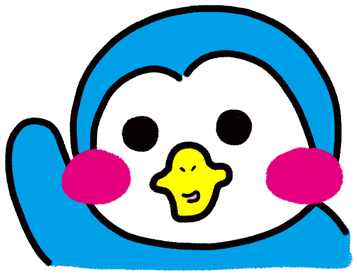
```
const pet = ["강아지", "고양이", "햄스터"];

let dog, cat, hamster, parrot;

// 배열 length는 3인데?!
[dog, cat, hamster, parrot] = pet;

console.log(dog); // 강아지
console.log(cat); // 고양이
console.log(hamster); // 햄스터
console.log(parrot); // undefined
```



```
const pet = ["강아지", "고양이", "햄스터"];

let dog, cat, hamster, parrot;

// 배열 length는 3인데?!
[dog, cat, hamster, parrot="앵무새"] = pet;

console.log(dog); // 강아지
console.log(cat); // 고양이
console.log(hamster); // 햄스터
console.log(parrot); // 앵무새
```

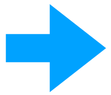
=> 디폴트 값이 있어도, 분해할 값이 있다면 디폴트 값은 무시된다!

객체의 구조 분해



객체를 구조 분해하는 기본적인 방법은 다음과 같다(앞에서 이미 봄).

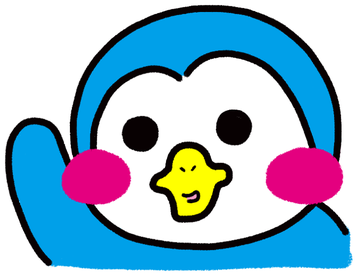
```
const cafe = {
  starbucks : "스타벅스",
  ediya : "이디야",
  pascucci : "파스쿠찌",
  coffeebean : "커피빈"
}
// 구조 분해를 통한 객체 값 복사!
const { pascucci, coffeebean } = cafe
console.log(pascucci, coffeebean)
```



```
파스쿠찌 커피빈
```

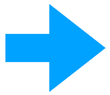
=> 키가 이름 그대로 '키' 역할을 한다(아무튼 중요하다는 뜻).

새로운 이름을 할당하자



객체의 원래 속성명(키)과는 다른 이름의 변수에 할당할 수도 있다.

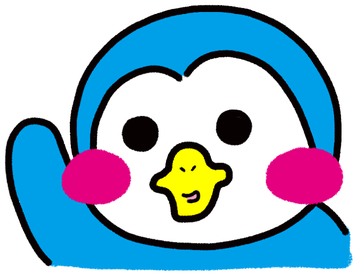
```
const cafe = {
  starbucks : "스타벅스",
  ediya : "이디야",
  pascucci : "파스쿠찌",
  coffeebean : "커피빈"
}

// 이름 바꿔줘
const { starbucks: sb } = cafe
// console.log(starbucks) // not defined!
console.log(sb) // 스타벅스
```



스타벅스

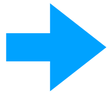
=> 키가 startbucks인 밸류를 복사하고, 거기에 sb라고 이름붙여 사용하자!

디폴트 선언 가능



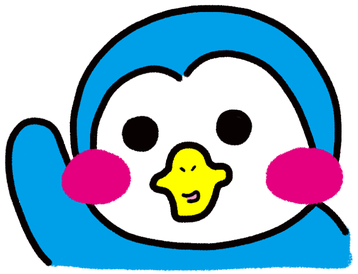
객체 구조 분해 시에도 디폴트 선언 가능하다.

```
const cafe = {
  starbucks : "스타벅스",
  ediya : "이디야",
  pascucci : "파스쿠찌",
  coffeebean : "커피빈"
}

// 투썸 별로라서 안 넣었음
const { twosome } = cafe
console.log(twosome) // undefined
```



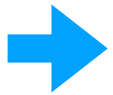
```
const cafe = {
  starbucks : "스타벅스",
  ediya : "이디야",
  pascucci : "파스쿠찌",
  coffeebean : "커피빈"
}

// 디폴트 값
const { twosome = "투썸플레이스" } = cafe
console.log(twosome) // 투썸플레이스
```

향상된 객체 리터럴



향상된 객체 리터럴이란 기존에 사용하던 객체 정의 방식을 개선한 문법이다.

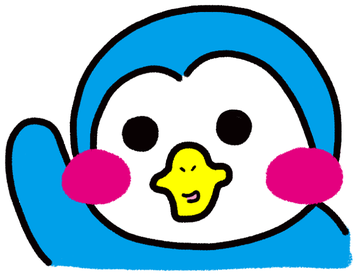
```
const avante = "아반떼"
const sonato = "소나타"

const car = {
  avante: avante,
  sonato: sonato
}

console.log(car)
```



```
const avante = "아반떼"
const sonato = "소나타"

const car = {
  avante, sonato
}

console.log(car)
```

=> 두 코드는 동일한 동작을 수행하는 코드이다.

내용 정리



- 구조 분해 할당을 사용하면 객체나 배열의 멤버를 변수로 선언할 수 있다.
- 여느 변수와 마찬가지로 선언과 초기화를 분리할 수 있다.
- 구조 분해 시 디폴트 값을 선언하여 undefined 값에 대비할 수 있다.
- 객체 초기화 시 변수의 이름만 기입하여 키와 밸류를 생성할 수 있다.