Current Skill Destructuring

Object access chain

When working with objects, we can use the dot notation (also called **object access chain**) to get the property of an object.

Let's see an example :

```
let student = { address: { city: 'New York', country: 'USA' } }


console.log(student.address.city) // New York
console.log(student.address.country) // USA
```

Use of variable

But when dealing with **deeply nested objects**, we might end up with a **long object access chain**. Naturally, repeating code leads to duplicates and duplicates are a problem. Some even say that duplicates are the most disliked annoyances when it comes to programming. So one of the alternatives is to use variables like we can see below:

```
let student = { address: { city: 'New York', country: 'USA' } }


let city = student.address.city
let country = student.address.country


console.log(city) // New York
console.log(country) // USA
```

ES6 Destructuring

Variables solve the issue of redundant code. But, it makes the code longer and harder to read. This is where **destructuring** comes into play. In simpler terms, the destructuring assignment syntax is a JavaScript expression that makes it possible to unpack values from arrays, or properties from

objects, into distinct variables.

All we need to do is to put the variable inside two **curly brackets** in the declaration.

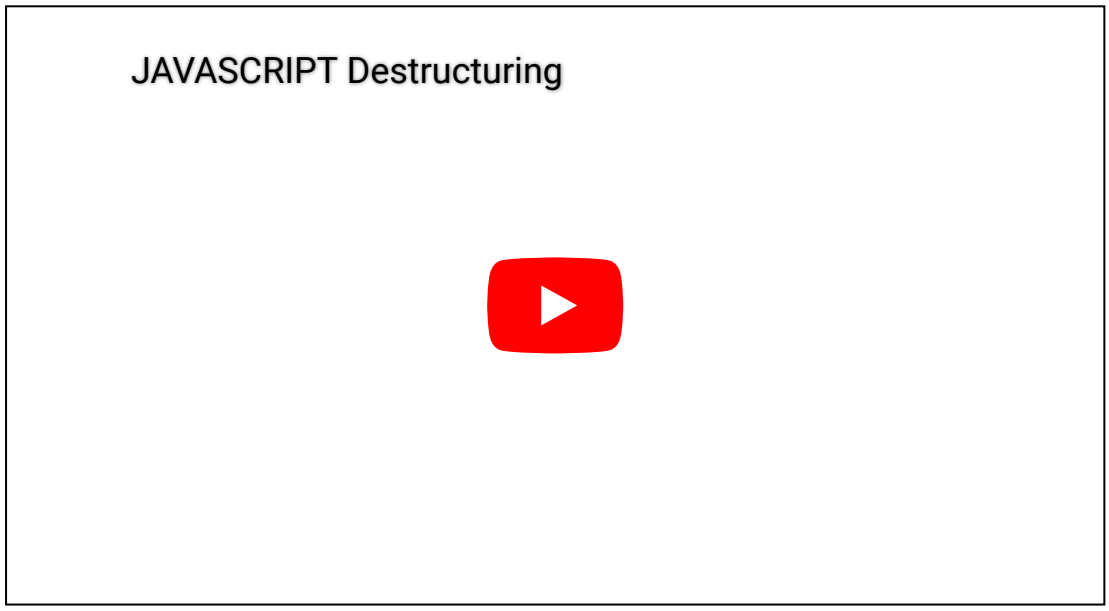
Notice : the variable **must have the same name** as the property inside the object.

You would do something like this:

```
let { address } = {address:{city:"New York", country:"USA"}};


let {city, country} = address;


console.log(city) // New York
console.log(country) // USA
```



Previous next