Destructuring

Dealing with objects and arrays has never been easier! Learn how to use destructuring to improve your code.

WE'LL COVER THE FOLLOWING

MDN defines **destructuring** like this:

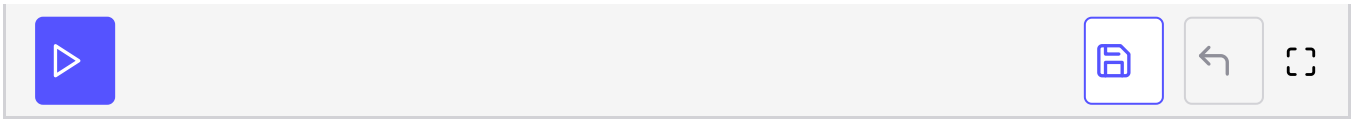
> The **destructuring** assignment syntax is a `JavaScript` expression that makes it possible to unpack values from arrays, or properties from objects, into distinct variables.

Let's start with **destructuring objects** first.

Destructuring Objects

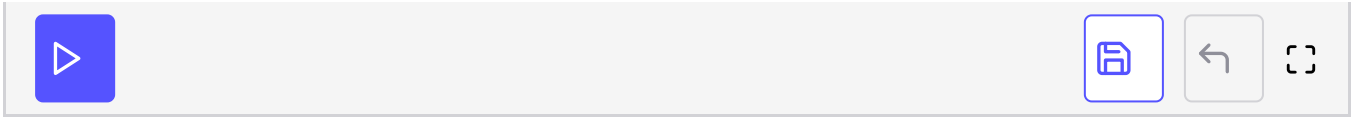
To create variables from an object, we used to do the following:

```
var person  = {
  first: "Alberto",
  last: "Montalesi"
}

var first = person.first;
var last = person.last;
console.log(first,last);
```



In ES6 we can instead do it as following:

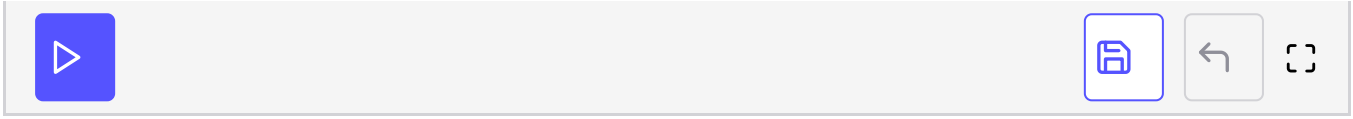
```
const person = {
  first: "Alberto",
  last: "Montalesi"
}

const { first, last } = person;
console.log(first,last);
```



Since our `const` variable- `person`- has the same name as the properties we want to grab, we don't have to specify `person.first` and `person.last` anymore.

The same applies even when we have nested data, such as what we could get from an API.

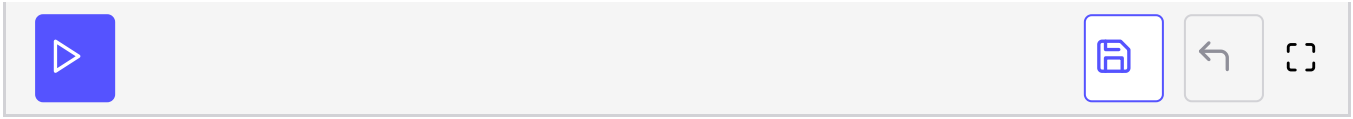
```
const person = {
  name: "Alberto",
  last: "Montalesi",
  links:{
    social: {
      facebook: "https://www.facebook.com/alberto.montalesi",
    },
    website: "http://albertomontalesi.github.io/"
  }
}

const { facebook } = person.links.social;
console.log(facebook);
```



We're not limited to name our variables the same as the property of the object. We can also rename as the following:

```
const person = {
  name: "Alberto",
  last: "Montalesi",
  links:{
    social: {
      facebook: "https://www.facebook.com/alberto.montalesi",
    },
    website: "http://albertomontalesi.github.io/"
  }
}
```

```
const { facebook:fb } = person.links.social;
// it will look for the property person.links.social.facebook and name the variable fb
console.log(fb); // https://www.facebook.com/alberto.montalesi
console.log(facebook); //ReferenceError: facebook is not defined
```



We are using the syntax `const { facebook:fb }` to specify that we want to target the property `facebook` of the object `person.links.social`, and that we want the `const` variable to be called `fb`. That is why when we try to log `facebook` we get an error.

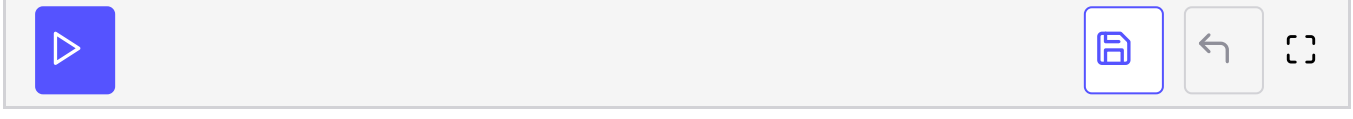
We can also pass in **default values** like this:

```
const { facebook:fb = "https://www.facebook.com"} = person.links.social;
// we renamed the variable to *fb* and we also set a default value to it
```

Destructuring Arrays

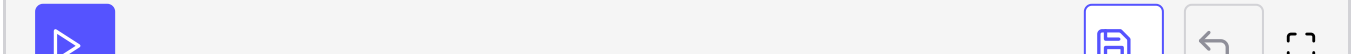
The first difference we notice when **destructuring arrays** is that we are going to use `[]` and not `{}`.

```
const person = ["Alberto","Montalesi",25];
const [name,surname,age] = person;
console.log(name);
// Alberto
```



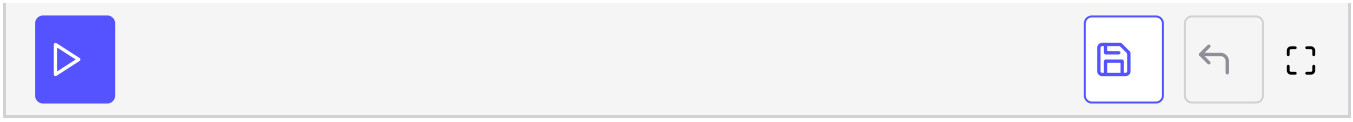
What if the number of variables that we create is less than the elements in the array?

```
const person = ["Alberto","Montalesi",25];
// we leave out age, we don't want it
const [name,surname] = person;
//the value of age will not be bound to any variable.
console.log(name,surname);
// Alberto Montalesi
```



Let's say we want to grab all the other values remaining. We can use the **rest operator** for that:

```
const person = ["Alberto", "Montalesi", "pizza", "ice cream", "cheese cake"];
// we use the **rest operator** to grab all the remaining values
const [name,surname,...food] = person ;
console.log(food);
// Array [ "pizza", "ice cream", "cheese cake" ]
```



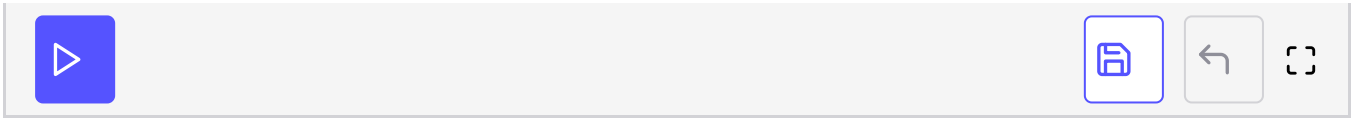
In the example above, the first two values of the array were bound to `name` and `surname`, while the rest (that's why it's called the **rest operator**) get assigned to `food`

The `...` is the syntax for the **rest operator**.

Swapping variables with destructuring

The destructuring assignment makes it **extremely easy** to swap variables, just look at this example:

```
let hungry = "yes";
let full = "no";
// after we eat we don't feel hungry anymore, we feel full, let's swap the values

[hungry, full] = [full, hungry];
console.log(hungry,full);
// no yes
```



It can't get easier than this to swap values.

Moving on to the quiz and another challenge now.