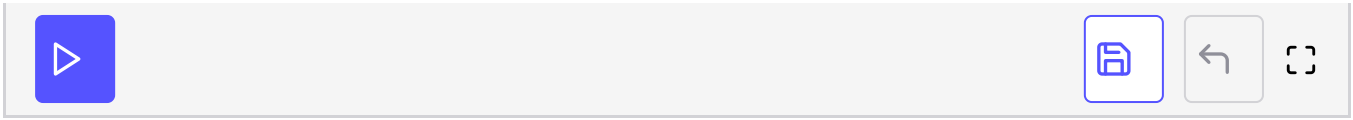
Destructuring

Destructuring is a way for us to extract data from arrays and objects without having to manually create variables. Take this object for example.

```
let person = {
    name: "Ryan",
    age: 30,
    location: "Toronto"
};
```

An example of the destructuring syntax would be this.

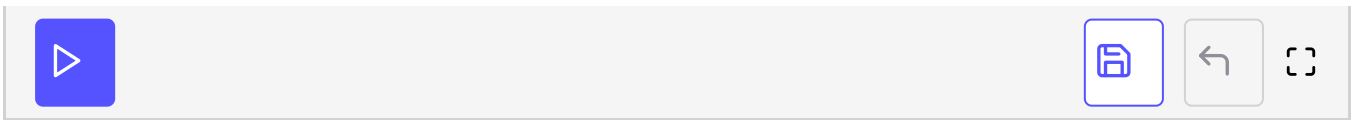
```
let { age : personAge } = person;
console.log(personAge);
```



In the destructuring expression the left-hand value is the key from the object, and the right is the new variable we would like to create. Don't want to create a new variable? We can use a special shorthand to just create a new variable called `age`.

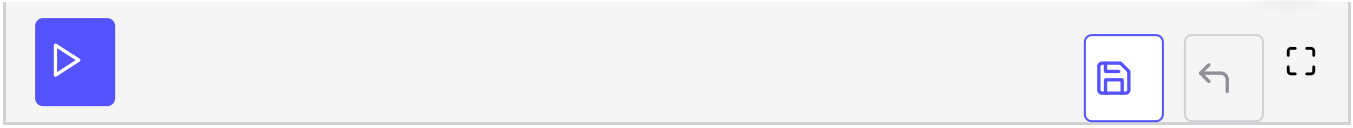
> this paragraph is confusing

```
let { age } = person;
console.log(age);
```



The great thing about this is that we can do this on multiple keys.

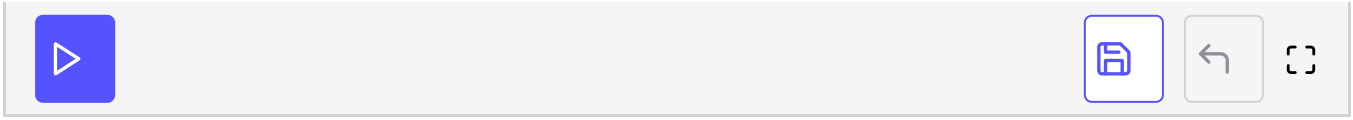
```
let { age , location: currentLocation} = person;
```



Note that we mix and match the shorthand with the longer assignment. If we had done this with just the shorthand `location` key as the variable, there would be an issue! Since `location` refers to the `window.location` property, changing that will change the location of the page! So in this case we have to assign a new variable name.

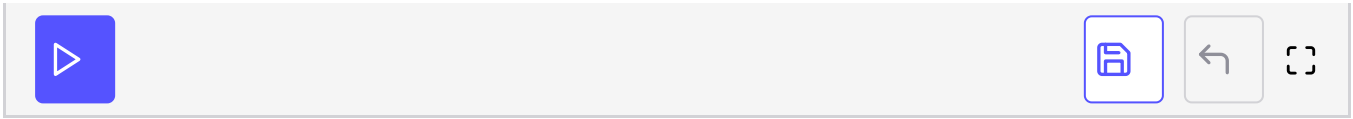
We can even use computed properties (like we saw in the Objects chapter) to pull out the values.

```
let key = 'age';

let { [key]: keyAge } = person;
```



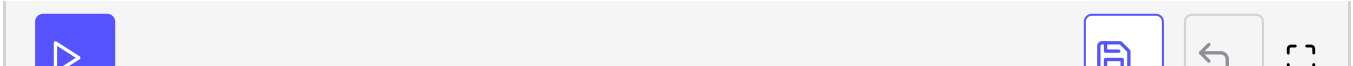
In this case we have a string value stored on our variable, and this will now act as our key. Along with objects we can also destructure arrays. Instead of being able to extract the values based on the keys from an object, we can just use variables that represent elements in the array.

```
let numbers = [1,2,3,4,5];

let [first,second] = numbers;
console.log(first,second); //1 2
```



Unlike destructuring an object, we can't just use the key name to select any element in the array. Luckily we have the ability to skip elements in the array.

```
let numbers = [1,2,3,4,5];

let [first,second,,fourth] = numbers;
console.log(first,second,fourth); // 1 2 4
```



Destructuring is a pretty common practice when pulling from a module, and we will see this a little later on. But if you have done any React work with React-Router you will have probably seen code like this (look at the highlighted code).

```
import React from 'react';
import ReactDom from 'react-dom';
import { Router, Route, Link } from 'react-router'

let App = React.createClass({
    render() {
        return (
            <div>
                <h1>My App</h1>
                <Link to="/somepage">A Link</Link>
            </div>
        );
    }
});
```

This is just a look at how we can use destructuring and modules together, more to come later!