

# the Master Course

{C0DENATION}

# Using the Map method.

{CODENATION}

React.js

# Learning Objectives

**To be familiar with using the map method in React.js**

**When using JavaScript we have worked heavily with lists (arrays) of items. We use arrays when we have a group of things that are related to each other.**

```
const bookPrices = [2.5, 4.5, 3.5, 6.0, 8.0];
```

**Let's take this array which stores some prices for books we are selling.**

Say we wanted to have a sale, and reduce the price of each book by 50%. I'd like an array of the new sale prices. I could use a **for loop** to make that happen.

```
const bookPrices = [2.5, 4.5, 3.5, 6.0, 8.0];
```

```
let salePrices = [];
```

```
for (i = 0; i < bookPrices.length; i++) {  
  let sale = bookPrices[i] * 0.5;  
  salePrices.push(sale);  
}
```

```
console.log(salePrices) // returns [ 1.25, 2.25, 1.75, 3, 4 ]
```

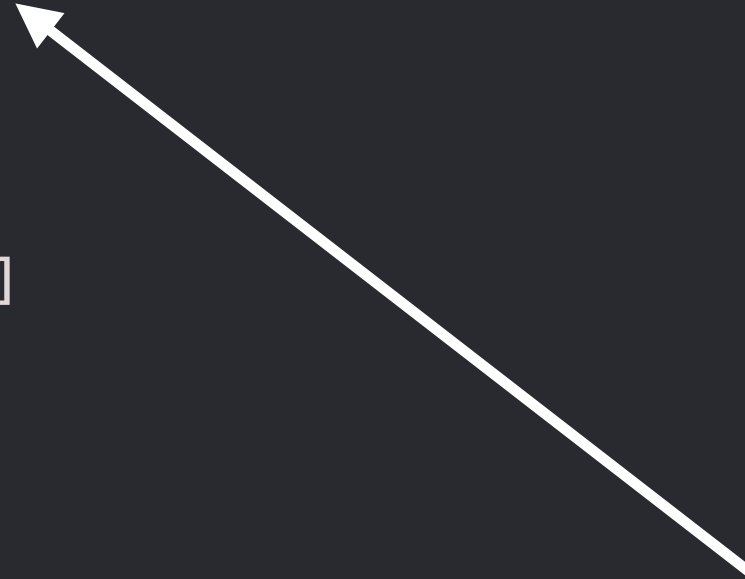
But there is a nicer way using an **array method**.

We can make use of the JavaScript **map method**. This array method takes an array, and will create a new array – with the results of calling a function for every element.

```
const bookPrices = [2.5, 4.5, 3.5, 6.0, 8.0];
```

```
const salePrice = bookPrices.map((price) => {  
  return price * 0.5;  
});
```

```
console.log(salePrice)  
// returns [ 1.25, 2.25, 1.75, 3, 4 ]
```



**This parameter is named by you. Try to name it something relevant**

```
array.map(function(currentValue, index, arr), thisValue)
```

# Parameter Values

Parameter	Description
<i>function(currentValue, index, arr)</i>	Required. A function to be run for each element in the array. Function arguments:



```
array.map(function(currentValue, index, arr), thisValue)
```

Parameter Values

Parameter	Description								
<i>function(currentValue, index, arr)</i>	Required. A function to be run for each element in the array. Function arguments: <table><tr><th>Argument</th><th>Description</th></tr><tr><td><i>currentValue</i></td><td>Required. The value of the current element</td></tr><tr><td><i>index</i></td><td>Optional. The array index of the current element</td></tr><tr><td><i>arr</i></td><td>Optional. The array object the current element belongs to</td></tr></table>	Argument	Description	<i>currentValue</i>	Required. The value of the current element	<i>index</i>	Optional. The array index of the current element	<i>arr</i>	Optional. The array object the current element belongs to
Argument	Description								
<i>currentValue</i>	Required. The value of the current element								
<i>index</i>	Optional. The array index of the current element								
<i>arr</i>	Optional. The array object the current element belongs to								
<i>thisValue</i>	Optional. A value to be passed to the function to be used as its "this" value. If this parameter is empty, the value "undefined" will be passed as its "this" value								

```
const bookPrices = [2.5, 4.5, 3.5, 6.0, 8.0];
```

```
const salePrice = bookPrices.map((price) => {  
  return price * 0.5;  
});
```

```
console.log(salePrice)  
// returns [ 1.25, 2.25, 1.75, 3, 4 ]
```

**So basically we can run a function for each element in an array, and whatever the return value is will form an element within a new array. Pretty cool.**

**Why should we care about this so much when working with React?**

**What if the return value from our map function was some JSX?**



```
class App extends React.Component {  
  render() {  
    const bookPrices = [2.5, 4.5, 3.5, 6.0, 8.0];  
  
    const salePrice = bookPrices.map(price => {  
      return <h1>The price is {price}</h1>; ←  
    });  
  
    return (  
      <div>  
        {salePrice}  
      </div>  
    )  
  }  
}
```

The map method runs the function for each array item (5 times in total), **each time returning a h1 element**. If we render this array, we will have 5 h1 elements on the screen!

**The price of the book is 2.5**

**The price of the book is 4.5**

**The price of the book is 3.5**

**The price of the book is 6**

**The price of the book is 8**

**But as we are using  
a different array  
element each time,  
each h1 is showing  
different data!**



```
class App extends React.Component {
```

```
  render() {
```

```
    const bookPrices = [2.5, 4.5, 3.5, 6.0, 8.0];
```

```
    const salePrice = bookPrices.map(price => {  
      return <h1>The price is {price}</h1>;  
    })
```

```
    return (  
      <div>
```

```
        {salePrice}
```

```
      </div>
```

```
    )
```

```
  }
```

```
}
```

Move this section into the JSX

A white arrow originates from the bottom-left corner of the magenta-bordered code block and points towards the `{salePrice}` placeholder within the `<div>` JSX element.

**We can actually tidy this code up a little bit. Instead of creating a new constant to store the new array in, we can just render it directly in the JSX.**



```
class App extends React.Component {  
  
  render() {  
    const bookPrices = [2.5, 4.5, 3.5, 6.0, 8.0];  
  
    return (  
      <div>  
        {bookPrices.map(price => {  
          return <h1>The price is {price}</h1>;  
        })}  
      </div>  
    )  
  }  
}
```

**So in your user interface, you will have a div with 5 h1 elements inside.**



```
class App extends React.Component {  
  state = {  
    bookPrices: [2.5, 4.5, 3.5, 6.0, 8.0]  
  }  
  
  render() {  
  
    return (  
      <div>  
        {this.state.bookPrices.map(price => {  
          return <h1>The price of the book is {price}</h1>;  
        })}  
      </div>  
    )  
  }  
}
```

**Sometimes you might be mapping through an array you store in your state. Just remember to use the dot notation correctly.**



```
✖ Warning: Each child in a list should have index.js:1  
a unique "key" prop.  
  
Check the render method of `App`. See https://fb.me/react-warning-keys for more information.  
    in h1 (at App.js:9)  
    in App (at src/index.js:7)
```

**If I opened up my browser console after writing the previous code, I would likely see this error. What does it mean?**

# From the React docs:

## Keys

Keys help React identify which items have changed, are added, or are removed. Keys should be given to the elements inside the array to give the elements a stable identity:

**We need to give the (parent) element in our return statement a key, so React can uniquely identify it. This happens behind the scenes. Also, it only has to be unique amongst its siblings (so different to the other other elements within the map)**



```
class App extends React.Component {  
  state = {  
    bookPrices: [2.5, 4.5, 3.5, 6.0, 8.0]  
  }  
  
  render() {  
  
    return (  
      <div>  
        {this.state.bookPrices.map(price => {  
          return <h1 key = { } >The price of the book is {price}</h1>;  
        })}  
      </div>  
    )  
  }  
}
```



**What unique piece of information can we use as a key for each element in our returned array? What's actually unique about each of the original array items? In this example, the book prices themselves are unique, but we can't guarantee this is always the case.**

# Syntax

# Remember the optional parameters for the map method.

```
array.map(function(currentValue, index, arr), thisValue)
```

## Parameter Values

Parameter	Description
<i>function(currentValue, index, arr)</i>	Required. A function to be run for each element in the array. Function arguments:

# Syntax

# Remember the optional parameters for the map method.



```
array.map(function(currentValue, index, arr), thisValue)
```

## Parameter Values

Parameter	Description
<i>function(currentValue, index, arr)</i>	Required. A function to be run for each element in the array. Function arguments:



```
class App extends React.Component {  
  state = {  
    bookPrices: [2.5, 4.5, 3.5, 6.0, 8.0]  
  }  
  
  render() {  
  
    return (  
      <div>  
        {this.state.bookPrices.map((price, index) => {  
          return <h1 key = {index}>The price of the book is {price}</h1>;  
        })}  
      </div>  
    )  
  }  
}
```



**We can use the index of each item as the key  
for react to uniquely identify it.**



```
class App extends React.Component {
  state = {
    bookPrices: [2.5, 4.5, 3.5, 6.0, 8.0]
  }
  render() {

    return (
      <div>
        {this.state.bookPrices.map((price, index) => {
          return (
            <div key={index}> ←
              <h1>Hello I'm a h1 element</h1>
              <p>Oh hi, I'm a p element</p>
              <p>{price}</p>
              <button>I'm a button</button>
            </div>)
          })}
        </div>)
      )
    }
  }
}
```

**If your map  
function returns  
multiple  
elements, you  
only need to put  
a key on the  
parent.**

## News headlines >



**Police declare major incident  
after Wales flooding**

UK



**China reports third straight drop  
in virus cases**

CHINA



**Budget may be delayed, says  
cabinet minister**

UK POLITICS

## Sport headlines >



**LIVE Scottish Premiership:  
Aberdeen v Celtic**

SCOTTISH



**Skinner & Fagerson added to  
Scotland squad**

SCOTTISH RUGBY



**England beat NZ in T20 World  
Cup warm-up**

WOMEN'S CRICKET

**What if our map function returned entire  
components? Let's have a look at a few  
more examples**



```
class App extends React.Component {
```



```
  render() {  
    return (  
      <div>  
        <Person name="Dan" age={34} />  
        <Person name="Ben" age={21} />  
      </div>  
    );  
  }  
}
```

```
const Person = props => {  
  return (  
    <div>  
      <h1>{props.name}</h1>  
      <p>{props.age}</p>  
    </div>  
  );  
};
```

**Here we are  
rendering the Person  
component twice.  
Each time giving it  
multiple bits of data.  
Let's see how we can  
use the map method  
here.**

```
class App extends React.Component {
  state = {
    people: [
      { name: "Dan", age: 34 },
      { name: "Ben", age: 21 }
    ]
  };

```

First, we need an array to store our props. It has to be an array because map is an array method! I have stored my props as objects within an array.

```
  render() {
    return (
      <div>
        {this.state.people.map((person, index) => {
          return <Person key={index} name={person.name} age={person.age} />;
        })}
      </div>
    );
  }
}

```

```
const Person = props => {
  return (
    <div>
      <h1>{props.name}</h1>
      <p>{props.age}</p>
    </div>
  );
};

```

The person parameter represents each item in our array. As each of these items is an object, we can use dot notation to access its values. But these will be different each time.

**Dan**

34

**Ben**

21

**This is what would  
render to the screen.  
Two Person  
components, with  
different data each  
time.**

```
const Card = props => {  
  return (  
    <div className="card-container">  
      <img src={props.image} />  
      <p>{props.text}</p>  
    </div>  
  );  
};
```

**Say I have a Card component like this one here. It has an image and some text. I want to render this component to the screen multiple times, each time giving it different data as props.**

```
import Bowser from "./images/bowser.jpg";
import BMario from "./images/babymario.jpg";

class App extends React.Component {
  state = {
    cards: [
      { image: Bowser, text: "Bowser has bad acceleration." },
      { image: BMario, text: "Baby Mario has crazy acceleration" }
    ]
  };

  render() {
    return (
      <div className="container">
        {this.state.cards.map((card, index) => {
          return <Card key={index} image={card.image} text={card.text} />;
        })}
      </div>
    );
  }
}
```

## News headlines >



**Police declare major incident after Wales flooding**

UK



**China reports third straight drop in virus cases**

CHINA



**Budget may be delayed, says cabinet minister**

UK POLITICS

## Sport headlines >



**LIVE Scottish Premiership: Aberdeen v Celtic**

SCOTTISH



**Skinner & Fagerson added to Scotland squad**

SCOTTISH RUGBY



**England beat NZ in T20 World Cup warm-up**

WOMEN'S CRICKET

**This is kind of similar. 6 story components being rendered, each time with different data. In the previous examples, our data was hard coded by us. But eventually we might be using data from an external source.**



Bowser has bad acceleration.



Baby Mario has crazy acceleration



**\*Subject to css**

**Remember, when you are passing methods as props inside the map function, you can use the element value and/or the index as an argument.**

```
class App extends React.Component {

  clickHandler = button => {
    console.log(button);
  };

  render() {
    const buttons = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, "+", "-", "x", "/"];
    return (
      <div className="container">
        {buttons.map((button, index) => {
          return <Button key={index} label={button} click={() => this.clickHandler(button)} />;
        })}
      </div>
    );
  }
}

const Button = props => {
  return <button onClick={props.click}>{props.label}</button>;
};
```



**Common Problem:** You have used map to render a list of items to the screen. When you click on one of them, you want to identify which one exactly you have clicked on, so you can do something with it in a function.

**Solution:** pass the index as an argument to your function, from within the map array method. You can then use the index within your function logic to isolate the item you need.

**Example of use case:** You have a to-do list, and each item has a delete button. How would you be able to identify the one in the list you want to delete?

# Revisiting Learning Objectives

To be familiar with using the map method in React.js