MISSION READY

DARE TO

DEVELOP

React Props, State and Hooks

Reuben Simpson

Recap

- So far, we’ve looked at
  - React
  - JSX
- We’ve looked at the basics of React and how to create a react application.
- Today we will be looking at a few more aspects of React
  - Props
  - State
  - Hooks

React Props

- We know that in a React app we break up different parts of our app into components, often times, these components need information from other components in order to work properly.
- We can pass information to our components through Props.
- Let's look at an example of a component called Car that says

"The car is a {car_name}" where car name gets passed in as a prop.

Passing in a prop

- To pass in a prop to a component, we add an attribute to our component in the JSX.
- In this example we are importing the Car component in the App.js file and inserting it in our JSX
- To pass in a prop we can add an attribute to the Car component with the prop name.

```
import logo from "./logo.svg";
import "./App.css";
import Car from "./components/Car";

// parent/top component
function App() {
  return (
    <div> <Car /> </div>
  );
}

export de[illegible]lt App;
```

```
<div>  <Car carName="Toyota" /> </div>
```

Using props

- The Car.js component will look like this on the right
- Props will get passed to the component and can be used as a parameter.
- The prop we pass in can be accessed through the ***props*** object

```
//child component which uses the props
function Car(props) {
  return (
    <div>
      The car is a {props.carName}
    </div>
  );
}

export default Car;
```

Your turn

App.js

```
import Car from "./Car";

function App() {
  return (
    <div>  <Car carName="Toyota" />
</div>
  );
}

export default App;
```

Car.js

```


function Car(props) {
  return (
    <div>
      The car is a {props.carName}
    </div>
  );
}

export default Car;
```

More on ES6

Revisiting the array `map( )` + Object Destructuring

Array.map()

- The *map* calls a provided **function** once for each element in an array, in order, and constructs a new array from the results.

```
const numbers = [1, 4, 9];
const addOne = numbers.map((num) => num + 1); // [2, 5, 10]
```

- The **function** passed to the *map* method also takes in an optional ***index*** parameter, which gives the index of the current element being processed in the array.

Your turn

Run the following code and share the output

0 1 2

```
const fruits = ["apple", "mango", "orange"];

const mappedFruits = fruits.map((fruit, index) => {
  return `${index}. ${fruit}`;
});

console.log(mappedFruits);
```

Exercise 1

Given an array of objects on the right, use the map method to generate the following array.

```
[
  "1) 3 units of Apple costs $0.75",
  "2) 6 units of Mango costs $2.10",
  "3) 4 units of Banana costs $0.60",
];
```

[*Hint: Start with a hardcoded string logged to your console and breakdown the dynamic parts afterwards.*

```
const fruitOrder = [
  {
    name: "Apple",
    qty: 3,
    singlePrice: 0.25,
  },
  {
    name: "Mango",
    qty: 6,
    singlePrice: 0.35,
  },
  {
    name: "Banana",
    qty: 4,
    singlePrice: 0.15,
  },
];
```

Using map in *JSX*

- The map() method is used to loop over an array and manipulate or change its elements.
- To display contents of an array in the DOM, you need to use the ***map()*** method and return JSX from the function passed to map.
    - This is the most common use case of the map() function in React.

```
const users = [
  { id: 1, name: "Sally Rogers" },
  { id: 2, name: "Buddy Sorrell" },
  { id: 3, name: "Millie Helper" },
];
return (
  <div>
    {users.map((user) => (
      <div>{user.name}</div>
    ))}
  </div>
);
```

```jsx
const users = [
  { id: 1, name: "Sally Rogers" },
  { id: 2, name: "Buddy Sorrell" },
  { id: 3, name: "Millie Helper" },
];

function App() {
  return (
    <>
      {users.map((user) => (
        <div>{user.name}</div>
      ))}
    </>
  );
}
```

Renders

```html
<div>Sally Rogers</div>
<div>Buddy Sorrell</div>
<div>Millie Helper</div>
```

Object Destructuring – ES6

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

```
const { propertyName } = objectName;
```

```
let person = {
  firstName: "John",
  lastName: "Doe",
};

// Before ES6
let fName = person.firstName;
let lName = person.lastName;

// with ES6 Object destructuring
let { firstName: fName, lastName: lName } = person;
```

The identifier before the colon (:) is the property of the object and the identifier after the colon is the variable.

- If the variables have the same names as the properties of the object, you can make the code more concise as follows:

```
let { firstName, lastName } = person;
console.log(firstName); // 'John'
console.log(lastName); // 'Doe'
```

- Another example:

```
const user = {
  id: 42,
  isVerified: true,
};
const { id, isVerified } = user;
console.log(id); // 42
console.log(isVerified); // true
```

Exercise 2

- Extract the username and email from the object using the destructuring syntax.
- Console log the values and share a screenshot.

```
const myObj = {
  id: 1,
  name: "Leanne Graham",
  username: "Bret",
  email: "Sincere@april.biz",
  phone: "1-770-736-8031 x56442",
  website: "hildegard.org",
  company: {
    companyName: "Romaguera-Crona",
    catchPhrase: "It's collaboration time",
    bs: "harness real-time e-markets",
  },
};
```

Array Destructuring – ES6

- A simplified method of extracting multiple properties from an array by taking the structure and deconstructing it down into its own constituent parts through assignments.
  - Create a pattern that describes the kind of value you are expecting and make the assignment.

```
const [first, second, third] = ["Laide", "Gabriel", "Jets"];
let a, b;
[a, , b] = ["Lordy", "Crown", "Roses"];
console.log(a); //Output: Lordy
console.log(b); //Output: Roses
```

Using props - *Continued*

- There is another way to access our props in our component that uses the destructuring ES6 syntax.

Way 1: Accessing the prop through the *props* object

```
export default function Car(props) {
  return (
    <div>
      The car is a {props.carName}
    </div>
  );
}
```

Way 2: Destructuring

```
function Car({carName}) {
  return (
    <div>
      The car is a {carName}
    </div>
  );
}

export default Car;
```

Handling multiple props

- We can pass in multiple props to a component by adding more attributes

```
<Car carName="Toyota" model="Camry" />
```

- To access these attributes we can do

```
export default function Car(props) {
  return (
    <div>
      The car is a {props.carName} {props.model}
    </div>
  );
}
```

- Or

```
export default function CarWithDestruc({ carName, model }) {
  return (
    <div>
      The car is a {carName} {model}
    </div>
  )
}
```

Exercise 3

- Create a ***Person*** component that takes in a ***name*** and an ***age*** as props and has an h1 tag with the text

  “Hi I’m {*name*}, I am {*age*} years old”

- where ***name*** and ***age*** come from the props passed to the component

MISSION READY
www.missionreadyhq.com

DARE TO
DEVELOP

Thank you | Reuben Simpson