COURSEWARE

Static Data

Contents

Overview

In this module, we will look at using static data.

Data Sources

React is for producing front end components, so we need to be able to convert data from data sources into state and/or props - for example, displaying a list of items in a table from a HTTP search response.

We can convert a data source into an array, or create one from it.

This array can then be stored in state, and the component can then use the JavaScript `map` function to return the state array as a modified array of Components.

These can then be displayed to the screen!

Importing Static Data

Given a JSON file; we can use the information by importing it into our component.

An alias is given so that the array can be used.

Often in React, an array of data is mapped to return a Component.

Each element in the array is passed to the Component as a prop and used to generate a unique component, and the mapped array is used to display the contents of the array:

```
import Data from './Data';
import someData from './someData.json';
const AllData = () => {
    const allData = someData.map(data => <Data item={data} key={data.id}/>);
    return (
           <>
             <h2>The data supplied is:</h2>
              <div>
                {allData}
              </div>
            </>
        );
};

export default AllData;
```

Using a Dataset with map

Imagine we have an array called `people` that stores a person object holding an ID, their first name and their country of origin, imported from an external source such as a JSON file.

An array can then be used in the `render` function to produce JSX code for each person in the array through the `map()` function:

```
{people.map(person =>
    <p key={person.id.toString()}>This is {person.name}, they’re
from{person.country}
    </p>
)};
```

`map()` works in the following way:

- For every item in the array, take the item itself and its index
- Create a new anonymous function that receives item and index, and returns a value of the modified data

Sub-Components

Data can be passed from the state of a parent component to a child using props.

This is done through sub-components and initially changed within the parent's `render()` function:

```
…
    {people.map(person => (
        <Person key={person.id} name={person.name}
        country= {person.country} />
    ))};
…
```

Then, within the child, create a Person Function Component:

```
const Person = (props) => (
    <p>This is {props.name}, they're from {props.country}</p>
);

export default Person;
```

Conditional Rendering

We can use conditionals to only render certain items depending on certain conditions.

Conditional rendering in React works in the same way as conditional statements in JavaScript.

`If` statements can be used to create a return for a Function component.

Ternary statements can also do this, and can also be used to conditionally set properties, such as `className`.

Below is an example that will render a button based on whether a user is logged in:

```
const Login = (props) => {
    if(props.isLoggedIn) {
      return <button onClick={props.logout}> Log Out </button>
    } else {
        return <button onClick={props.login}>Log In</button>
    }
};

export default Login;
```

Tutorial

In this tutorial, we are going to take a look at importing static data.

1. Copy the information from this link and store into a file called `'sampleData.json'`

2. Create a function component called `Content`

```
const Content = () => {

}
export default Content;
```

3. Inside the `Content` component, in the return function, write a `map()` function to recurse through the `sampleData` and print the person's name.

```
return(
  <div>
    {sampleData.map((person)=>(
      <p> Hello, {person.name} </p>
    ))}
  </div>
)
```

4. Add the Person's city to the `<p>` using by accessing the 'address' property.

```
<p> Hello, {person.name} from {person.address.city}!</p>
```

5. Export the component, import into `App.js` and run using `npm start`

6. You should see the following output:

Hello, Leanne Graham from Gwenborough!

Hello, Ervin Howell from Wisokyburgh!

Hello, Clementine Bauch from McKenziehaven!

Hello, Patricia Lebsack from South Elvis!

Hello, Chelsey Dietrich from Roscoeview!

Hello, Mrs. Dennis Schulist from South Christy!

Hello, Kurtis Weissnat from Howemouth!

Hello, Nicholas Runolfsdottir V from Aliyaview!

Hello, Glenna Reichert from Bartholomebury!

Hello, Clementina DuBuque from Lebsackbury!

▶ Content.jsx

Now, let's have a look at how we can modify this code into a sub-component example.

1. Create a `SubContent.jsx` component.

2. Import `sampleData` from `sampleData.json`

3. In the return statement, write a map function to loop over the `sampleData`, instead of returning a `<p>` element, return a `<Person>` component, with the following attributes:

    1. `key = {person.id}`

    2. `name = {person.name}`

    3. `city = {person.address.city}`

```
...
return(
<>
  {sampleData.map(person => (
    <Person key={person.id} name={person.name} city=
{person.address.city}/>
  ))};
</>
)
...
```

4. Now create a `Person` component which takes in `props` as argument

```
const Person = (props) => {

}
```

5. In the body of the component, write a `return` function which returns a `<p>` element with the attributes from the `props` argument. It should say '`This is <NAME>, they are from <CITY>`'

```
return <p> This is {props.name}, they are from {props.city}<p>
```

6. Export as default, make sure it is imported in the `SubContent` Content, then run the application.

7. The output you should see is:

This is Leanne Graham, they are from Gwenborough

This is Ervin Howell, they are from Wisokyburgh

This is Clementine Bauch, they are from McKenziehaven

This is Patricia Lebsack, they are from South Elvis

This is Chelsey Dietrich, they are from Roscoeview

This is Mrs. Dennis Schulist, they are from South Christy

This is Kurtis Weissnat, they are from Howemouth

This is Nicholas Runolfsdottir V, they are from Aliyaview

This is Glenna Reichert, they are from Bartholomebury

This is Clementina DuBuque, they are from Lebsackbury

▶ Code

Exercises

1. Create a component `LoginControl` which displays:
    1. Header with "Please sign up" as text and a Login Button
    2. Header with "Welcome Back!" as text and a Logout Button
    3. Onclick on each of the buttons should take the user between the two displays above.

▶ Solution