COURSEWARE

Props

Contents

Overview

In this module we will be discussing React Props.

React and Data

React only supports **uni-directional** data flow.

Data flows from the top of a component tree to the bottom - data cannot flow back up the component tree.

Data that does not change over the lifetime of the component should be considered as **props**.

Data that can change should be considered as **state**.

`State` should be the single source of truth for changing data.

All components that rely on this should receive the data as `props`.

State should be in the highest common component of those that require the data.

What are props?

According to Facebook's GitHub documentation, (http://facebook.github.io/react/docs/thinking-in-react.html) “props are a way of passing data from parent to child”.

Think of this as a communication channel between components that always moves from the top (parent) to the bottom (child).

props are immutable - once they are set, they cannot change.

```
<App headerProp = "Header from attr" />
```

props can be added as attributes in the component used when rendering it from `ReactDOM.render`

Default props can also be defined under the component declaration in the `.jsx` file:

```
App.defaultProps = {
    headerProp : `Header from default`,
    contentProp : `Content from default`
}
```

Using Props in Components

props must be passed to a Function component as an argument for it to be aware of them.

props are rendered to the browser through the component return – either the default, or overriding value (if supplied).

```
const App = props => (
  <div>
    <h1>{props.headerProp}</h1>
    <p>{props.contentProp}</p>
  </div>
  );
```

Here is a full example of how props can be used

```
import ReactDOM from 'react-dom';
const App = props => {
    return (
        <div>
            <h1>{props.headerProp}</h1>
            <p>{props.contentProp}</p>
        </div>
    );
};

// props rendered in the component here

App.defaultProps = {
    headerProp: `This is the default heading`, contentProp: `This is default
content`
}

// defaultProps set here
ReactDOM.render(
    <App headerProp = "Header from attribute" />, document.querySelector('#app')
    );
```

This component will be rendered as expected, with the header being displayed from the overriding attribute setting, and the content being rendered from the default.

Prop types

The sub-object `propTypes` can be used for both typing and validation (uses the PropTypes class from the prop-types npm package).

Prop types are useful for ensuring the correct usage of components.

Any valid JavaScript type can be used, and it will produce a console warning if the correct type is not used for a prop.

We can also validate that a prop has a value supplied with `.isRequired` being chained to a propTypes declaration.

This will ensure that a console warning is produced if prop is not available, and any undeclared props are ignored by the browser.

```
//importing all of the relevant packages
import ReactDOM from 'react-dom';
import PropTypes from 'prop-types';

const App = props => {
    return (
        <>
          <h1>{props.headerProp}</h1>
          <p>{props.contentProp}</p>
          <p>Value of numberProp is: {props.numberProp}</p>
        </>
    );
};

App.defaultProps = {
    headerProp: `This is the default heading`,contentProp: `This is default
content`
}

App.propTypes = {
    headerProp: PropTypes.string.isRequired,
    contentProp: PropTypes.string.isRequired,
    numberProp: PropTypes.number
}

ReactDOM.render(<App numberProp={10} />, document.querySelector('#app'));
```

Tutorial

1. Create a file with a `.jsx` extension.
2. Create a const called Hello as an arrow function that takes `props` arguments

```
const Hello = props => {

}
```

3. Make the function return a `<h1>` heading, containing a name that is de-structured from the `props` parameter

```
return (
    <h1>Hello, {props.name}</h1>
);
```

4. Now, we create a `PropComp` Component that renders `Hello` many times

```
const PropComp = () =>{
    return(
        <>
            <Hello name="Chris"/>
            <Hello name="P."/>
            <Hello name="Bacon"/>
        </>
    );
}
```

5. In your `App.js` file - import the `<PropComp/>`
6. Remove EVERYTHING in the return and replace it with `<PropComp/>`

```
import logo from './logo.svg';
import './App.css';
import PropComp from './Hello';

const App = () => {
return (
    <PropComp/>
);
}

export default App;
```

7. Run the app using `npm start`

Exercises

Create a new file called ComponentWithProps.jsx with a defined Function component called `ComponentWithProps`

It should have props as an argument, and a return that has:

- A wrapping React Fragment;
- A <h1> that uses header from props as its content;
- A <p> that uses content from props as its content;
- A <p> that uses number from props as its content along with some text;
- A <p> that uses nonexistent from props as its content along with some text.

Following this, make sure to do the following

- export ComponentWithProps as default.
- Import the new component into your MyComponent.jsx file
- Ensure that you wrap the return of MyComponent in a React Fragment

Save and run the file to see the results.