codecademy

Components and Props

Returning HTML Elements and Components

A class component's `render()` method can return any JSX, including a mix of HTML elements and custom React components.

In the example, we return a `<Logo />` component and a "vanilla" HTML title.

This assumes that `<Logo />` is defined elsewhere.

```
class Header extends React.Component {
  render() {
    return (
      <div>
        <Logo />
        <h1>Codecademy</h1>
      </div>
    );
  }
}
```

React Component File Organization

It is common to keep each React component in its own file, `export` it, and `import` it wherever else it is needed. This file organization helps make components reusable. You don't need to do this, but it's a useful convention.

In the example, we might have two files: **App.js**, which is the top-level component for our app, and **Clock.js**, a sub-component.

```
// Clock.js
import React from 'react';

export class Clock extends
React.Component {
  render() {
    // ...
  }
}

// App.js
import React from 'react';
import { Clock } from './Clock';

class App extends React.Component {
  render() {
    return (
      <div>
        <h1>What time is it?</h1>
        <Clock />
      </div>
    );
  }
}
```

this.props

React class components can access their props with the `this.props` object.

In the example code below, we see the `<Hello>` component being rendered with a `firstName` prop. It is accessed in the component's `render()` method with `this.props.firstName`.

This should render the text "Hi there, Kim!"

```
class Hello extends React.Component {
  render() {
    return <h1>Hi there,
{this.props.firstName}!</h1>;
  }
}

ReactDOM.render(<Hello firstName="Kim"
/>, document.getElementById('app'));
```

defaultProps

A React component's `defaultProps` object contains default values to be used in case props are not passed. If a prop is not passed to a component, then it will be replaced with the value in the `defaultProps` object.

In the example code, `defaultProps` is set so that profiles have a fallback profile picture if none is set. The `<MyFriends>` component should render two profiles: one with a set profile picture and one with the fallback profile picture.

```
class Profile extends React.Component {
  render() {
    return (
      <div>
        <img src=
{this.props.profilePictureSrc} alt="" />
        <h2>{this.props.name}</h2>
      </div>
    );
  }
}

Profile.defaultProps = {
  profilePictureSrc:
'https://example.com/no-profile-
picture.jpg',
};

class MyFriends extends React.Component {
  render() {
    return (
      <div>
        <h1>My friends</h1>
        <Profile
          name="Jane Doe"
          profilePictureSrc="https://exam
ple.com/jane-doe.jpg"
        />
        <Profile name="John Smith" />
      </div>
    );
  }
}
```

`props`

Components can pass information to other components. When one component passes information to another, it is passed as *props* through one or more *attributes*.

The example code demonstrates the use of attributes in props. `SpaceShip` is the component and `ride` is the attribute. The `SpaceShip` component will receive `ride` in its props.

```
<SpaceShip ride="Millennium Falcon" />
```

`this.props.children`

Every component's `props` object has a property named `children`. Using `this.props.children` will return everything in between a component's opening and closing JSX tags.

```
<List>  // opening tag
  <li></li> // child 1
  <li></li> // child 2
  <li></li> // child 3
</List> // closing tag
```