React 101
Codecademy
React Components

Components and Advanced JSX

Use Multiline JSX in a Component

This can be achieved by using parentheses in the return statement.

```
import React from 'react';
import ReactDOM from 'react-dom';

class App extends React.Component {
  render() {
    return (
    <blockquote>
      <p>What is important now is to recover our senses.</p>
      <cite>
        <a                                               target="_blank"
href="https://en.wikipedia.org/wiki/Susan_Sontag"> Susan Sontag </a>
        </cite>
    </blockquote>
    );
  }
};

ReactDOM.render(<App />,document.getElementById('app'));
```

Use a Variable Attribute in a Component

```
import React from 'react';
import ReactDOM from 'react-dom';

const redPanda = {
  src:
'https://upload.wikimedia.org/wikipedia/commons/b/b2/Endangered_Red_Pan
da.jpg',
  alt: 'Red Panda',
  width:  '200px'
};

class RedPanda extends React.Component {
  render() {
    return (
      <div>
        <h1>Cute Red Panda</h1>
        <img
          src={redPanda.src}
          alt={redPanda.alt}
          width={redPanda.width} />
      </div>
    );
  }
}

ReactDOM.render(
  <RedPanda />,
  document.getElementById('app')
);
```

Using variables to define the component properties we can make our components dynamic, easy to change, and read.

Put Logic in a Render Function

A render() function must have a return statement. But that's not all.

A render() function can also be a fine place to put simple calculations that need to happen right before a component renders.

```
class Random extends React.Component {
  render() {
    // First, some logic that must happen
    // before rendering:
    // be careful not to put this outside the render function
    const n = Math.floor(Math.random() * 10 + 1);
    // Next, a return statement
    // using that logic:
    return <h1>The number is {n}!</h1>;
  }
}
```

If the const n is declared outside the render function but inside the component then it will give a syntax error as it should not be a part of the class declaration itself, but should occur in a method like render().

Use a Conditional in a Render Function

We can use the if statement inside the render function before the return statement. This is pretty much the only way one would see an if statement used in a render function.

```
import React from 'react';
import ReactDOM from 'react-dom';

class TodaysPlan extends React.Component {
  render() {
    let task;
    if (!apocalypse) {
      task = 'learn React.js'
    } else {
      task = 'run around'
    }

    return <h1>Today I am going to {task}!</h1>;
  }
}

ReactDOM.render(
      <TodaysPlan />,
      document.getElementById('app')
);
```

Use this in a Component

this keyword is used a lot in React a lot!

```
class IceCreamGuy extends React.Component {
  get food() {
    return 'ice cream';
  }

  render() {
    return <h1>I like {this.food}.</h1>;
  }
}
```

this refers to an instance of IceCreamGuy. The less simple answer is that this refers to the object on which this’s enclosing method, in this case .render(), is called. It is almost inevitable that this object will be an instance of IceCreamGuy, but technically it could be something else.

Use an Event Listener in a Component

Render functions often contain event listeners.

An event handler is a function that gets called in response to an event.

In React, we define event handlers as methods on a component class.

```
class MyClass extends React.Component {
  myFunc() {
    alert('Stop it.  Stop hovering.');
  }

  render() {
    return (
      <div onHover={this.myFunc}>
      </div>
    );
  }
}
```

We can also invoke multiple event handler calls by passing an anonymous function in response to a single event.

```
class MyButton extends React.Component {
  eventHandler1() {
    console.log('eventHandler1 called!');
  }

  eventHandler2() {
    console.log('eventHandler2 called!');
  }

  render() {
    return (
      <button onClick={() => {
        this.eventHandler1();
        this.eventHandler2();
      }}>Here's a button!</button> // here `onClick` is set to an anonymous function where the function body contains multiple function calls that will be triggered on the click event of the button
    )
  }
};
```

We can also use a named function.

```
class MyButton extends React.Component {
  eventHandler1() {
    console.log('eventHandler1 called!');
  }

  eventHandler2() {
    console.log('eventHandler2 called!');
  }

  handleClick = () => {
    this.eventHandler1();
    this.eventHandler2();
  }

  render() {
    return (
      <button  onClick={this.handleClick}>Here's  a  button!</button>  //
here `onClick` is set to a named function `handleClick` where the function
body contains multiple function calls that will be triggered on the click
event of the button
    )
  }
};
```