Defining components

src/components/Button/Button.js

```
import React, { Component } from "react";
import "./Button.css";
class Button extends Component {
  render() {
    <button className="Button"
      onClick={this.props.onClick}>
      {this.props.children}
    </button>
  }
}

// If no state, can be re-written as
// "functional component" short-hand
const Button = (props) => (
  <button className="Button"
    onClick={props.onClick}>
    {props.children}
  </button>
);
export default Button;
```

React Terms

component One discrete, re-usable, self-contained portion of React code that can be used multiple times in a project for repeatable graphical components

props Short for "properties", props are *immutable* and represent the data passed down to components from the parent of a component as attributes

lifecycle methods Methods that have special names in React which are triggered at certain points in a React component's lifecycle

function / stateless component Many components might not need any state, only props, and can be re-written in a function syntax short-hand

unidirectional data-flow The idea that parents pass data to children via props, while children can never interact with siblings or with their parents directly

Virtual DOM Novel technique to speed up rendering while seemingly rerenders entire page (does "dry run" to render a "virtual DOM", compares what changed with the real DOM, and only makes minimum tweaks)

Destructuring

```
const info = {name: "jane", age: 35};
const name = info.name;
const age = info.age;
// Equivalent to
const info = {name: "jane", age: 35};
const {name, age} = info;
```

Using components

```
import React, { Component } from "react";
import Button from "./components/Button/Button.js";
class App extends Component {
  state = {
    count: 0,
  }

  increment() {
    this.setState({
      count: this.state.count + 1,
    });
  }

  render() {
    return (
      <Button onClick={this.increment}>
        Click me {this.state.count}
      </Button>
    )
  }
}
```

Useful snippets

Conditional rendering

```
render() {
  if (!this.props.text) {
    return (
      <p><em>No text found...</em></p>
    );
  }

  return (
    /*... full render method here ... */
  );
}
```

Using `map` to loop through data

```
<div>{
    this.props.data.map((item, index) => (
      <p onClick={this.performAction(index)}>
        {index}: {item}
      </p>
    ))
}</div>
```

Using `?` `:` (ternary operator) for an "if-statement"

```
<div>{
    this.props.image ? (
      <img src={this.props.image} />
    ) : <em>No image provided.</em>
}</div>
```