Styles

Direct Import

There are multiple ways to add styles to a React app. One way is to simply import a CSS file directly and use its classes. For example, assume the following CSS file and component file are in the same directory.

```css
/* styles.css */
.btn {
    padding: 12px 15px;
    font-family: Arial, sans-serif;
    font-weight: bold;
    background: linear-gradient(30deg, rebeccapurple, magenta);
    color: #fff;
    border: none;
}
```

```jsx
// MyButtonComponent.js
import React, { Component } from 'react';
import './styles.css';

const MyButton = (props) => {
    return (
        <button className="btn">{ props.children }</button>
    );
}

export default MyButton;
```

One drawback of this approach is that styles are not encapsulated to the component. In other words, by importing this styles.css file, if we use the "button" class anywhere else in our application, these same styles will be applied, which may not be the desired outcome. Note that we use `className` instead of `class` when writing JSX, as class is a reserved word in JavaScript.

Inline Styles

A second way we can incorporate styles is by directly passing an object to the style attribute of an HTML element. For example, see the following:

```jsx
// MyButtonComponent.js
import React, { Component } from 'react';

const btnStyle = {
    padding: '12px 15px',
    fontFamily: 'Arial, sans-serif',
    fontWeight: 'bold',
    background: 'linear-gradient(30deg, rebeccapurple, magenta)',
    color: '#fff',
    border: 'none'
};

const MyButton = (props) => {
    return (
        <button style={btnStyle}>{ props.children }</button>
    );
}

export default MyButton;
```

While this approach does a good job of encapsulating the styles to the specific component, it has the drawback that media queries can't be used for responsiveness and it doesn't support pseudo-classes. Further, note that property names which are hyphenated in

standard CSS must be camel cased when used in objects like this. Also, all the values (including integers like 0) must be strings.

CSS Modules

A third approach overcomes a number of the problems of the first two: CSS Modules. Create-react-app supports CSS Modules by default, so we don't need to make any special adjustments to use them. Here's an example of the above button component using CSS Modules.

```
/* MyButtonComponent.module.css */
.btn {
    padding: 12px 15px;
    font-family: Arial, sans-serif;
    font-weight: bold;
    background: linear-gradient(30deg, rebeccapurple, magenta);
    color: #fff;
    border: none;
}
```

```
// MyButtonComponent.js
import React, { Component } from 'react';
import styles from './MyButtonComponent.module.css';

const MyButton = (props) => {
    return (
        <button className={styles.btn}>{ props.children }</button>
    );
}

export default MyButton;
```

Note that the name of the CSS file needs to end in "module.css" for this to work. Importing the styles gives us an object with all the different classes as key names. A small drawback of CSS Modules is that class names cannot be hyphenated, so by convention, camel casing is used. Two big advantages of CSS Modules are that (1) you can use media queries in them as usual, and (2) they are completely encapsulated at the component level. If you have another component using a different "btn" class, there is no conflict between them, as the class names are given unique hashes at build time to keep them isolated.