CSS WITH COMPONENT IN REACT

To use CSS with a component in React, you have a few different options. Here are three common approaches:

Inline Styling:

React allows you to apply styles directly to individual JSX elements using the style attribute. Styles are defined as JavaScript objects. ***You can define a JavaScript object containing CSS properties and values, and then pass it as the value of the style attribute.***

e.g.

```
import React from 'react';
const MyComponent = () => {
const myStyle = {
backgroundColor: 'blue',
   color: 'white',
   padding: '10px',
  };

  return (
<div style={myStyle}>
<h1>Hello, I am a styled component!</h1>
<p>I have inline styles applied to me.</p>
</div>
  );
}
export default MyComponent;
```

Internal CSS (CSS Modules):

You can use CSS Modules, which are a popular approach for localizing styles in React. This allows you to create a CSS file for each component, and the class names are scoped to that component, preventing style conflicts.

First, install the necessary dependencies (e.g., css-loader and style-loader) if not already included in your project. :**npm install --save style-loader css-loader**

How CSS Modules (Internal CSS) Work in React

- **File Naming Convention:** CSS Modules often use a naming convention that includes the component name and the ***.module.css*** extension. For example, if you have a React component called "MyComponent," you would create a corresponding CSS Module file named "MyComponent.module.css." The ".module" part is a convention used by many build tools to identify CSS Modules.
- **Local Scoping**: CSS Modules automatically scope the CSS styles to the component with which they are associated. This means that the class names and styles defined in a CSS Module are specific to that component. As a result, you won't run into naming conflicts or styling issues when you use the same class names in different components.

- **Importing Styles:** In your React component JavaScript file, you import the CSS Module using the import statement, just like you would for other JavaScript or CSS files.
  e.g.
  import React from 'react';
  import **styles** from '**./MyComponent.module.css'**;

- **Using Styles:** To apply styles to your component's elements, you use the class names from the imported CSS Module.
  e.g.
  const MyComponent = () => {
    return <div className**={styles.myClass}>**Hello, World!</div>;
  };

Benefits of CSS Modules in React:

- **Scoped Styles**: CSS Modules ensure that styles are encapsulated within the component, reducing the risk of global styling conflicts.

- **Maintainability:** Styles are co-located with their corresponding components, making it easier to understand and maintain the code.

- **Readability:** CSS Modules make it clear which styles are applied to a particular component by looking at the imported class names.

- **Performance:** Modern build tools can optimize CSS Modules to improve performance, such as tree-shaking and generating minimal CSS bundles.

- **Predictable Class Names:** You don't need to worry about creating unique class names; the build tool generates them automatically.

- **Tool Compatibility:** CSS Modules are supported by various build tools, such as Webpack, Parcel, and Create React App.

e.g.

MyComponent.module.css

```css
.container {
  background-color: blue;
  color: white;
  padding: 10px;
}

.heading {
  font-size: 24px;
  color: aqua;
}

.paragraph {
  font-size: 16px;
  color: brown;
}

#p1{
  color:black;
}
```

MyComponent.js

```
import React from 'react';
import styles from "./MyComponent.module.css" ;
const MyComponent = () => {
 return (
<div className={styles.container}>
<h1 className={styles.heading}>Hello, I am a styled component!</h1>
<p id={styles.p1}>I have inline styles applied to me.</p>
</div>
 );
}
export default MyComponent;
```

App.js

```
import React from 'react';
import MyComponent from './MyComponent';
const App = () => {
 return (
<div>
<h1>Welcome to my app</h1>
<MyComponent />
</div>
 );
}
export default App;
```

External CSS:

You can include external CSS files in your React application. Import the CSS file directly into your component file:

MyComponent.styles.css

```
.container {
   background-color: blue;
   color: white;
   padding: 10px;
  }

.heading {
   font-size: 24px;
  }

.paragraph {
   font-size: 16px;
  }
  #p1{
   color: pink;
  }
```

MyComponent.js

```
import React from 'react';
import './MyComponent.css'.;

const MyComponent = () => {
  return (
<div className={ 'container'}>
<h1 className={'heading'}>Hello, I am a styled component!</h1>
<p className={'paragraph'}>I have CSS module styles applied to me.</p>
</div>
  );
}
```

CSS-in-JS Libraries:

You can also use CSS-in-JS libraries like styled-components or Emotion, which allow you to define styles using JavaScript or template literals. These libraries offer a more dynamic way to style React components and have gained popularity in the React ecosystem.
Installation of package :**npm install styled-components**

e.g.

```
import React from 'react';
import styled from 'styled-components';
```

```
// Create a styled component
const StyledButton = styled.button`
  background: ${props => (props.primary ? 'blue' : 'white')};
  color: ${props => (props.primary ? 'white' : 'blue')};
  font-size: 16px;
  padding: 10px 20px;
  border: 2px solid blue;
  cursor: pointer;
`;

//const Button = ({primary,children}) => {
const Button = (props) => {
  return <StyledButton primary={props.primary}>{props.children}</StyledButton>;
};

// Usage of the Button component
function App() {
  return (
<div>
<Button>Secondary Button</Button>
<Button primary>Primary Button</Button>
</div>
  );
}
export default App;
```

Description :

1. We import styled from 'styled-components' to create styled components.

2. We create a styled button component called StyledButton using the tagged template literal syntax. The template literal allows you to define the CSS styles directly within the JavaScript code. The props can be used to conditionally apply styles based on the component's props.

3. The Button component takes a primary prop that determines the background and text colors. It renders the StyledButton with the appropriate styling based on the primary prop.

4. In the App component, we demonstrate how to use the Button component with different styles (primary and secondary).

styled-components offers several advantages, including scoped styles, dynamic styling, and better encapsulation of styles within components, making it a popular choice for styling React applications