**A Few Ways to Style a React App**

**1. Create Vite React Project**

Open your terminal and run:

npm create vite@latest my-react-app --template react

Then:

cd my-react-app

npm install

**2. Install SCSS Support**

SCSS is built into Vite. Just install Sass:

npm install sass

Create a file like App.scss:

.app {

background-color: #f5f5f5;

padding: 2rem;

}

Use it in App.jsx

import './App.scss';

**3. Install Material UI (MUI)**

npm install @mui/material @emotion/react @emotion/styled

Example:

import Button from '@mui/material/Button';

export default function App() {

return <Button variant="contained">Hello Material UI</Button>;

}

**4. Install styled-components**

npm install styled-components

Optionally install TypeScript types (safe even in JS):

npm install --save-dev @types/styled-components

Example:

import styled from 'styled-components';

const Title = styled.h1`

color: teal;

font-size: 2rem;

`;

export default function App() {

return (

<div className="app">

<Title>Styled Title</Title>

</div>

);

}

**App.jsx using all three**

import './App.scss';

import Button from '@mui/material/Button';

import styled from 'styled-components';

const StyledDiv = styled.div`

padding: 1rem;

border: 2px dashed pink;

`;

function App() {

return (

<div className="app">

<StyledDiv>

<h1>Hello from styled-components</h1>

<Button variant="contained" color="primary">Material UI</Button>

</StyledDiv>

</div>

);

}

export default App;

**Benefits of styled-components in React:**

**1. Scoped Styles (No Global Leaks)**

Each styled-component is scoped to its component, so you never have to worry about class name collisions.

const Button = styled.button`color: red;`; // styles won’t leak outside

**2. Dynamic Styling with Props**

You can use JavaScript to apply styles based on props or state.

const Button = styled.button`

background: ${props => props.primary ? 'blue' : 'gray'};

`;

**3. Everything in One File**

Styles live alongside components, which makes code more cohesive and easier to maintain.

// No need to manage separate .css or .scss files

**4. Automatic Critical CSS**

Only the styles needed for the page are included in the output, improving performance.

**5. Theming Support Built-In**

Easily share design tokens (colors, spacing, fonts) across your app using the <ThemeProvider>.

<ThemeProvider theme={{ primaryColor: 'hotpink' }}>

<App />

</ThemeProvider>

**6. Component-based Style Reuse**

Create base components and extend them with styles:

const BaseButton = styled.button`padding: 1rem;`;

const DangerButton = styled(BaseButton)`background: red;`;

**7. No Manual Class Naming Needed**

It generates unique class names for you—no more long btn--primary\_\_highlighted class names.