Ways to Style Your React App

Williams Godsfavour

When it comes to styling your React app, we have a ton of different options.

Which do you choose?

I have broken down the 5 primary ways you can use to style your React app.

LET'S GET STARTED

Inline Styles

Inline styles are the most direct away to style any React application.
Styling elements inline doesn't require you to create a separate stylesheet.
Despite a few quick benefits, inline styles are only an acceptable choice for very small applications.

Williams Godsfavour

```
export default function App() {
  return (
     <div
      style={{
        fontFamily: '-apple-system',
        fontSize: "1rem",
        fontWeight: 1.5,
        lineHeight: 1.5,
        color: "#292b2c",
        backgroundColor: "#fff",
        padding: "0 2em"}}>
       This is a Div
     </div>
)}
```

Plain CSS

it's common to import a CSS stylesheet to style a component's elements. Writing CSS in a stylesheet is probably the most common and basic approach to styling a React application. Writing styles in "plain" CSS stylesheets is getting better all the time, due to an increasing set of features available in the CSS standard. This includes features like CSS variables to store dynamic values, all manner of advanced selectors to select child elements with precision,

Williams Godsfavour

```
/* src/styles.css */
.form-input {
  width: 100%;
  padding: 0.5rem 1rem;
  font-size: 1rem;
  font-weight: 400;
  line-height: 1.5;
  color: #293240;
  background-color: #ffffff;
  border-radius: 4px;
 }

// src/App.js
import "./styles.css";
export default function App() {
return <input type="text" className="form-input" /> }
```

SASS / SCSS

SASS gives us some powerful tools, many of which don't exist in normal CSS stylesheets. It includes features like variables, extending styles, and nesting.
SASS allows us to write styles in two different kinds of stylesheets, with the extensions .scss and .sass.

Williams Godsfavour

```scss
/* src/styles.scss */

nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
   } li {
      display: inline-block;
   } a {
    display: block;
    padding: 6px 12px;
    text-decoration: none;
}}
```

SASS / SCSS

SCSS styles are written in a similar syntax to normal CSS, however SASS styles do not require us to use open and closing brackets when writing style rules.

Compare this with the same code written in a SASS stylesheet:

Williams Godsfavour

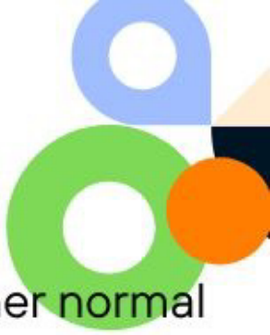
```
/* src/styles.sass */

nav
  ul
    margin: 0
    padding: 0
    list-style: none

  li
    display: inline-block

  a
    display: block
    padding: 6px 12px
    text-decoration: none
```

CSS MODULES



CSS modules are another slight alternative to something like CSS or SASS. What is great about CSS modules is that they can be used with either normal CSS or SASS. Our CSS file has the name .module in it before the extension .css. Any CSS module file must have the name "module" in it and end in the appropriate extension (if we are using CSS or SASS/SCSS).

Williams Godsfavour

```
/* src/styles.module.css */
.qoute {
  line-height: 24px;
  padding: 15px 40px;
  font-size: 14px;
  font-weight: 700;
  color: #4154f1;
  text-transform: uppercase;
  background: #4154f1;
  box-shadow: 0px 5px 25px rgba(65, 84, 241, 0.3);
}

import styles from './styles.module.css';
export default function App() {
return ( <p className={styles.quote}>I'm a Frontend developer</p>
)}
```

CSS-in-JS

The styled components allows us to write CSS styles directly in our components' javascript (.js) files Not only does it allow you write CSS style rules without having to make a single .css file, but these styles are scoped to individual components.
it often makes use of a special type of JavaScript function called a tagged template literal. Which we can still write plain CSS style rules directly in our JS!

Williams Godsfavour

```
/* src/styles.module.css */
import styled from "styled-components";
const Button = styled.button`
  color: blue;
  border: 2px solid blue;
  font-size: 14px;
  margin: 20px 10px;
  padding: 5px 10px;
  border-radius: 3px;
  &:hover {
    opacity: 0.9;
  }`;

export default function App() {
    return <div><Button>Click me</Button></div>
}
```