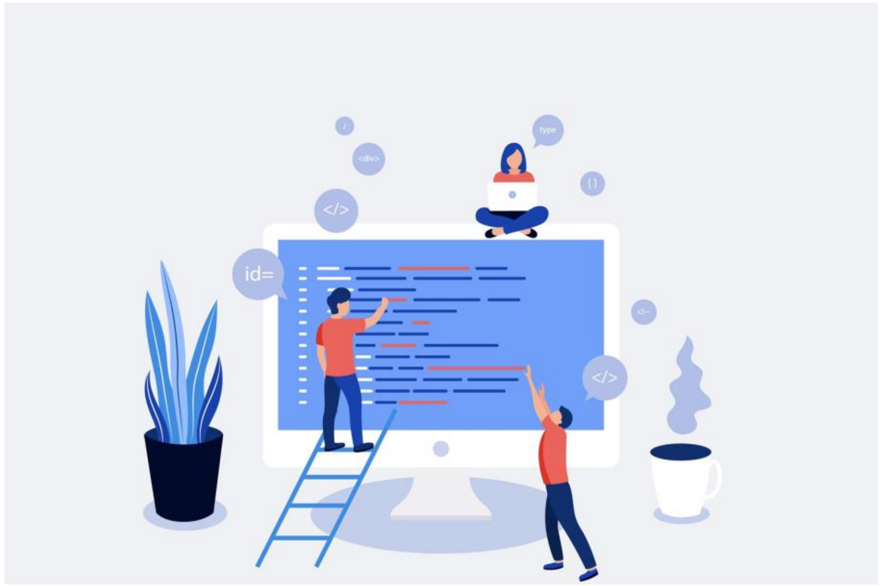
Styling in React

React Session-3



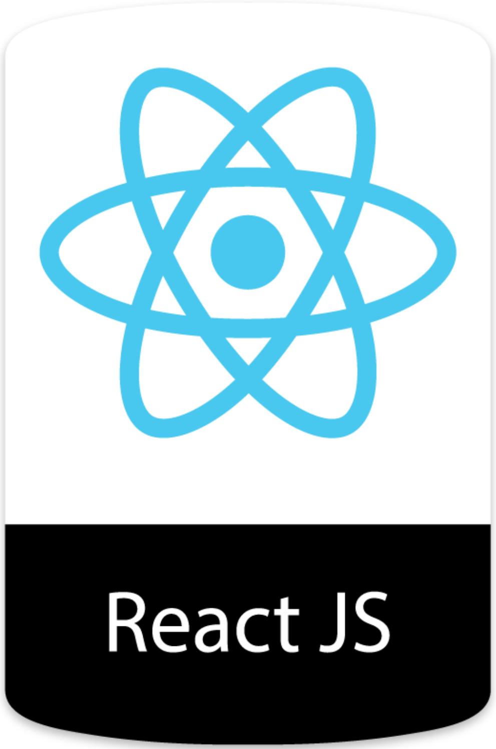
CLARUSWAY
WAY TO REINVENT YOURSELF

Table of Contents

- Styling Components in React
- a. Inline Styling
- b. CSS Stylesheet
- c. CSS Modules
- d. SASS

Did you finish React pre-class material?

YES NO



Kahoot!

1

Styling Components in React

There is a famous saying, "*It is the box, that helps to sell the jewelry*".

Styling Components in React

- As visual elements, styling them is a big part of how applications actually meet our users, and composes the way our brand and product looks and feels.

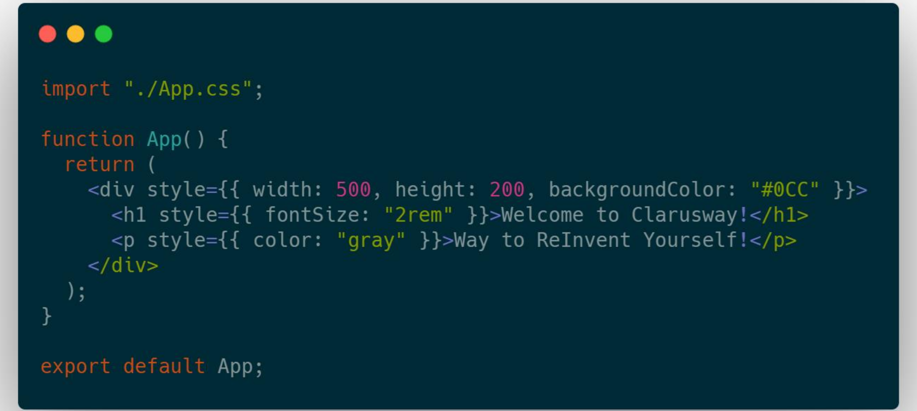
- Choosing the right method for styling components isn’t an absolute truth. It’s a relative choice that should serve your use case, personal preferences and above all architectural goals of the way you work: Global namespacing, dependencies, reusability, scalability, dead-code elimination and so on.

Styling Components in React

- Using Inline Styling
- Importing CSS Stylesheets
- Importing CSS Modules
- Using CSS pre-processor Sass
- Using Styled-Components

2

a. Inline CSS



```
import "./App.css";

function App() {
  return (
    <div style={{ width: 500, height: 200, backgroundColor: "#0CC" }}>
      <h1 style={{ fontSize: "2rem" }}>Welcome to Clarusway!</h1>
      <p style={{ color: "gray" }}>Way to ReInvent Yourself!</p>
    </div>
  );
}

export default App;
```

CLARUSWAY
WAY TO REINVENT YOURSELF

a. Inline CSS

- Change the CSS property name to its `camelCase` version like "`background-color`" to "`backgroundColor`", "`font-size`" to "`fontSize`", etc.
- Create an object with all the CSS properties as keys and their CSS values.
- Assign that object to the style attribute.

Disadvantages of Inline CSS

- Duplication of CSS properties
- CSS properties will be limited to a component scope only, so there is zero reusability
- We will not be able to utilize the full power of CSS, for example, no pseudo-classes, pseudo-element, media queries, keyframe animations, etc.
- It is hard to maintain or edit/update, and lot of inline CSS can reduce the code readability
- It hampers the performance, on each re-rendering the style object will be recomputed

CLARUSWAY
WAY TO REINVENT YOURSELF

Inline CSS Example

```
import "./App.css";


function App() {
  return (
    <div style={{ width: 500, height: 200, backgroundColor: "#0CC" }}>
      <h1 style={{ fontSize: "2rem" }}>Welcome to Clarusway!</h1>
      <p style={{ color: "gray" }}>Way to ReInvent Yourself!</p>
    </div>
  );
}
export default App;
```

Inline CSS Example

```
const myStyles = {
  div: { width: 500, height: 200, backgroundColor: "#0CC"},
  headerOne: {fontSize: "2rem"},
  par: {color: "gray"}};
function App() {
  return (
    <div style={myStyles.div}>
      <h1 style={myStyles.headerOne}>Welcome to Clarusway!</h1>
      <p style={myStyles.par}>Way to ReInvent Yourself!</p>
    </div>
)}
export default App;
```

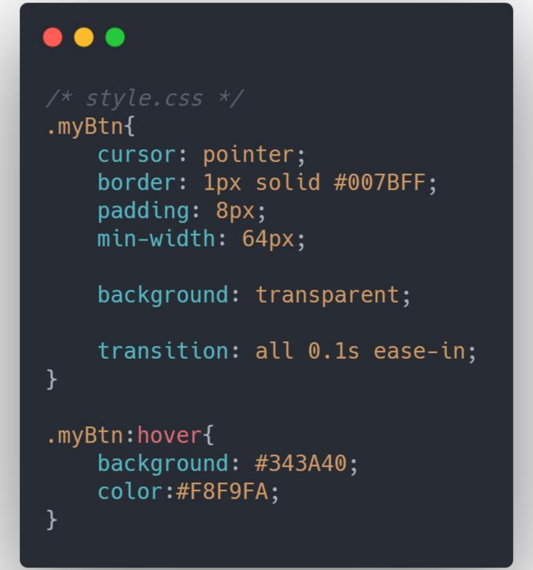
3

b. CSS Stylesheet

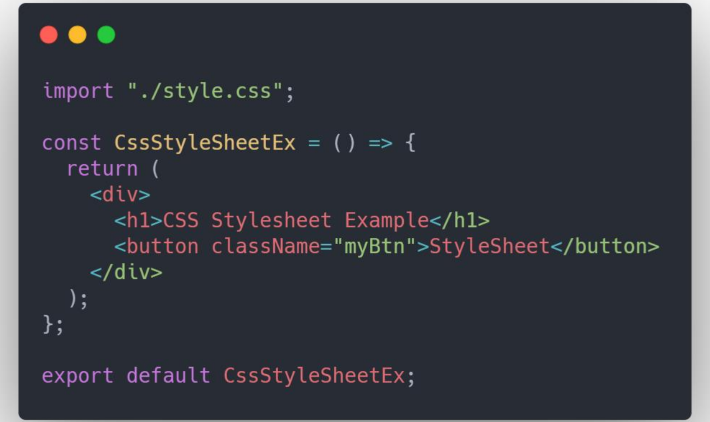
CLARUSWAY
WAY TO REINVENT YOURSELF

b. CSS Stylesheet

You can write your CSS styling in a separate file, just save the file with the `.css` file extension, and import it in your application.



```css
/* style.css */
.myBtn{
    cursor: pointer;
    border: 1px solid #007BFF;
    padding: 8px;
    min-width: 64px;

    background: transparent;

    transition: all 0.1s ease-in;
}

.myBtn:hover{
    background: #343A40;
    color:#F8F9FA;
}
```

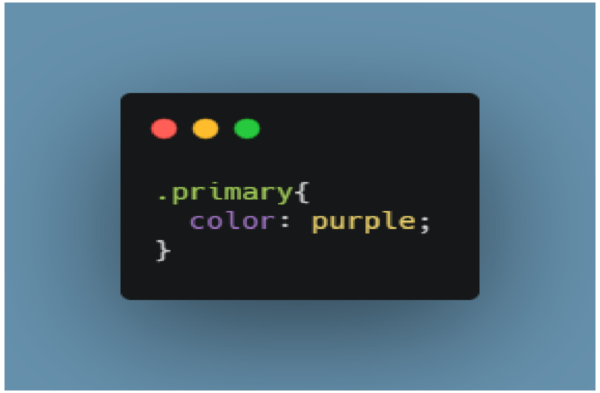


```jsx
import "./style.css";

const CssStyleSheetEx = () => {
  return (
    <div>
      <h1>CSS Stylesheet Example</h1>
      <button className="myBtn">StyleSheet</button>
    </div>
  );
};

export default CssStyleSheetEx;
```

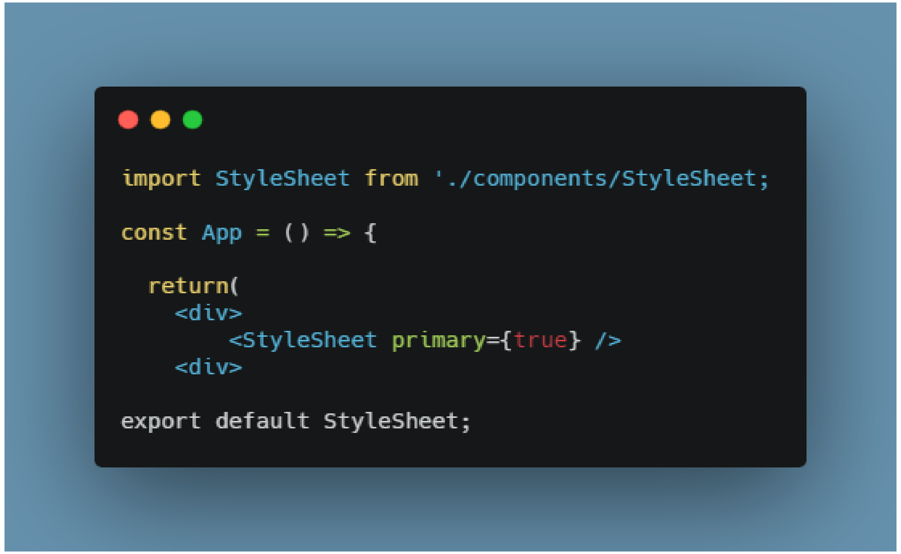
CLARUSWAY
WAY TO REINVENT YOURSELF

b. CSS Stylesheet

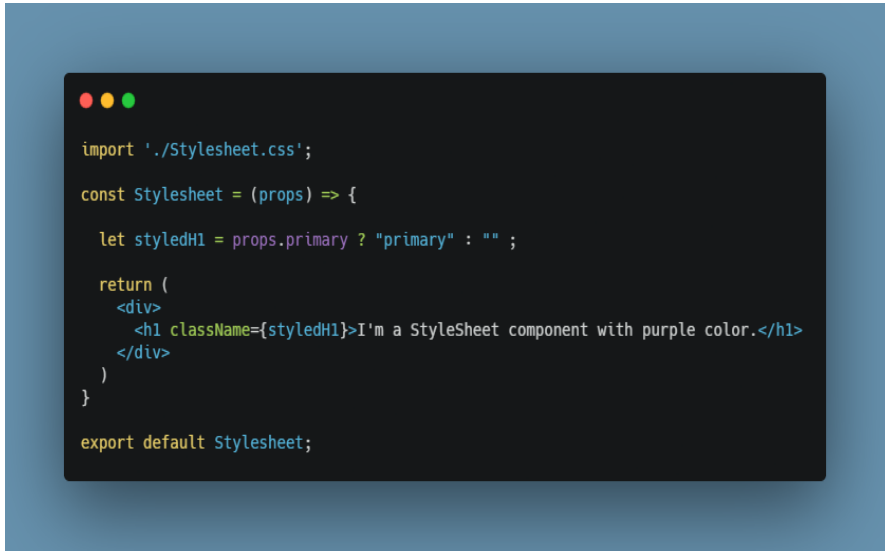
You can also conditionally apply a class, based on props or state of the component.



```css
.primary{
  color: purple;
}
```



```jsx
import StyleSheet from './components/StyleSheet;

const App = () => {

  return(
    <div>
        <StyleSheet primary={true} />
    <div>

export default StyleSheet;
```

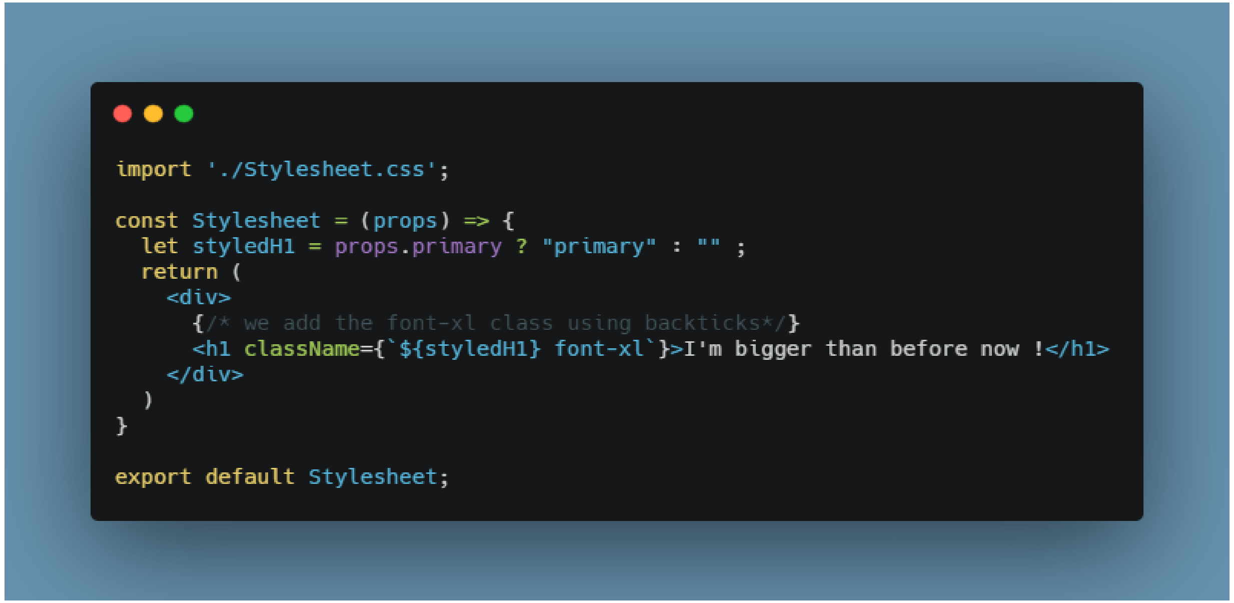


```jsx
import './Stylesheet.css';

const Stylesheet = (props) => {

  let styledH1 = props.primary ? "primary" : "" ;

  return (
    <div>
      <h1 className={styledH1}>I'm a StyleSheet component with purple color.</h1>
    </div>
  )
}

export default Stylesheet;
```

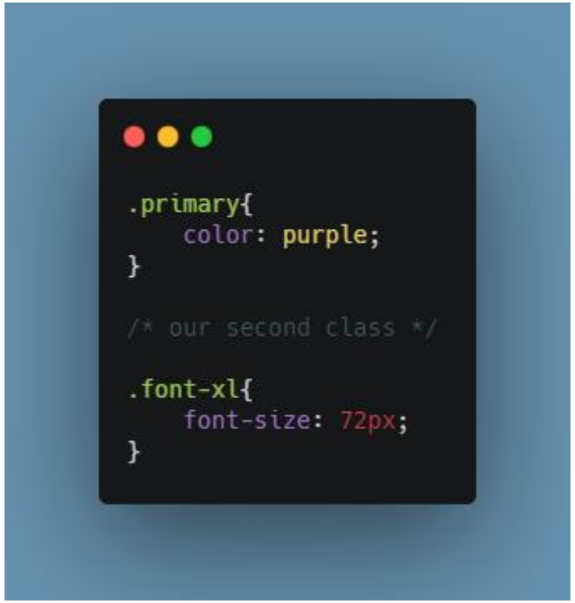
CLARUSWAY
WAY TO REINVENT YOURSELF

b. CSS Stylesheet

If you want to specify multiple classes, the simplest option is to use template literals.



```
import './Stylesheet.css';

const Stylesheet = (props) => {
  let styledH1 = props.primary ? "primary" : "" ;
  return (
    <div>
      {/* we add the font-xl class using backticks*/}
      <h1 className={`${styledH1} font-xl`}>I'm bigger than before now !</h1>
    </div>
  )
}

export default Stylesheet;
```

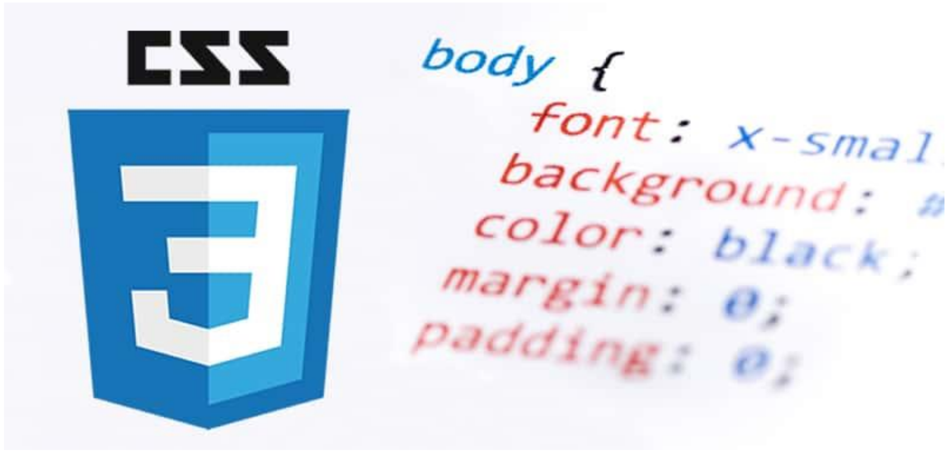


```
.primary{
    color: purple;
}

/* our second class */

.font-xl{
    font-size: 72px;
}
```

CLARUSWAY
WAY TO REINVENT YOURSELF

4

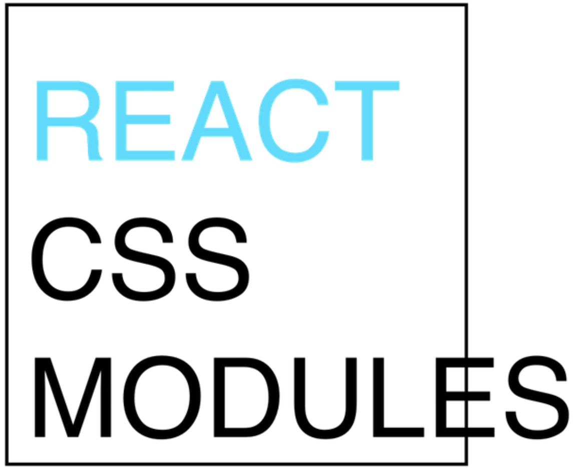
c. CSS Modules



c. CSS Modules

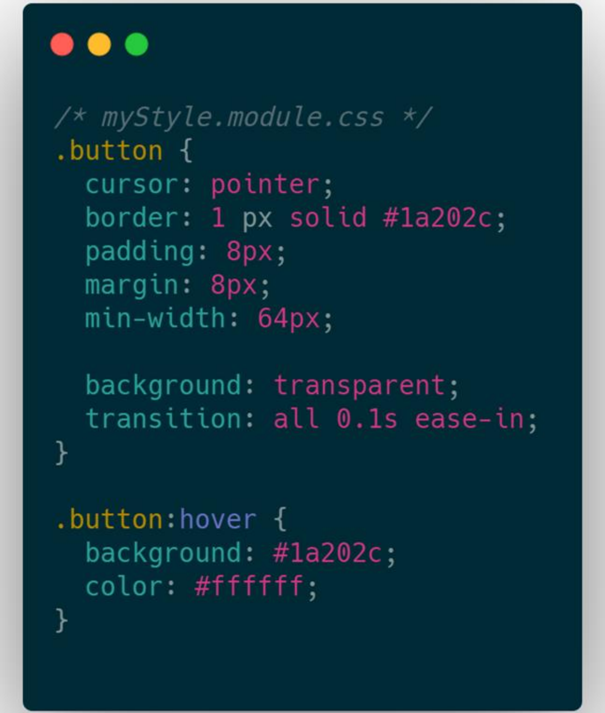
- A CSS Module is a CSS file in which all class names and animation names are scoped locally- by default.
- In React, each React component gets its own CSS file, which is scoped to that file and component. For a React component that you'd like to style, simply create a CSS file that'll contain the styles for that component.
- At build time local class names are mapped and exported as a JS object literal for React- as well as a modified version of input CSS with renamed class names. The result is, you don't have to mess as much with global styles. When scaling your projects, you have less overrides and trouble on your hands.

c. CSS Modules

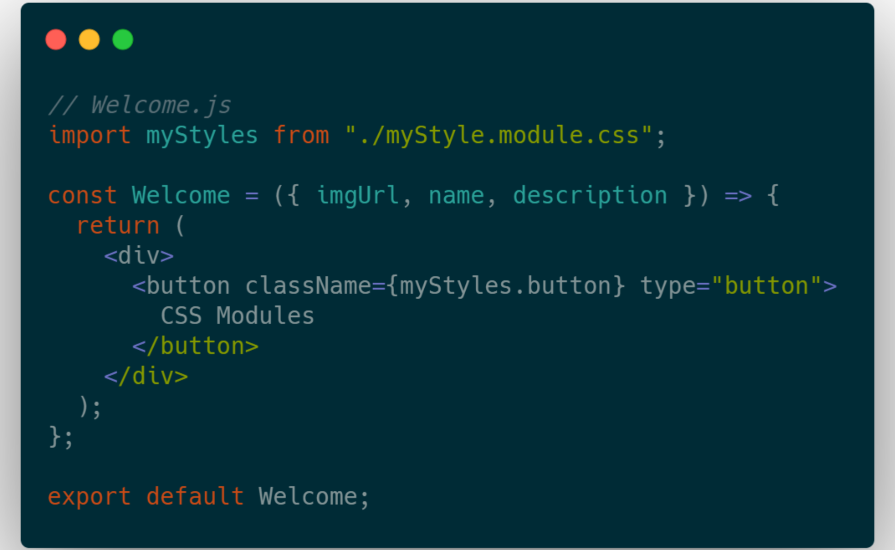


- A CSS Module feature is available with react scripts version 2 or higher.
- Css modules file name is a bit different from regular stylesheet.
- Css modules' file must be suffixed with '.module.css' such as myStyle.module.css

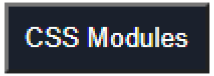
c. CSS Modules Example



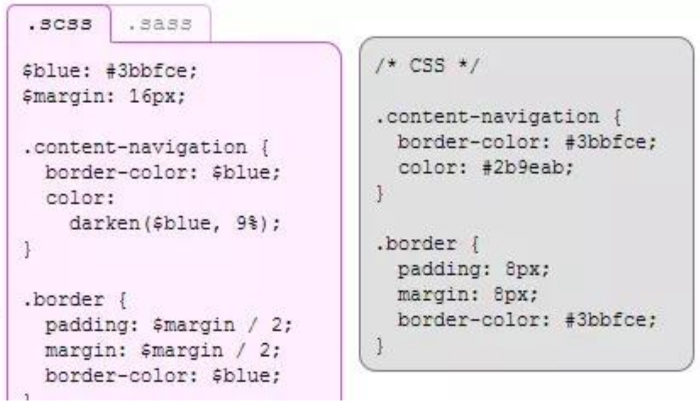
```css
/* myStyle.module.css */
.button {
  cursor: pointer;
  border: 1 px solid #1a202c;
  padding: 8px;
  margin: 8px;
  min-width: 64px;

  background: transparent;
  transition: all 0.1s ease-in;
}

.button:hover {
  background: #1a202c;
  color: #ffffff;
}
```



```jsx
// Welcome.js
import myStyles from "./myStyle.module.css";

const Welcome = ({ imgUrl, name, description }) => {
  return (
    <div>
      <button className={myStyles.button} type="button">
        CSS Modules
      </button>
    </div>
  );
};

export default Welcome;
```



5 d. SASS



```
$blue: #3bbfce;
$margin: 16px;

.content-navigation {
  border-color: $blue;
  color:
    darken($blue, 9%);
}

.border {
  padding: $margin / 2;
  margin: $margin / 2;
  border-color: $blue;
```

```
/* CSS */

.content-navigation {
  border-color: #3bbfce;
  color: #2b9eab;
}

.border {
  padding: 8px;
  margin: 8px;
  border-color: #3bbfce;
}
```

d. SASS

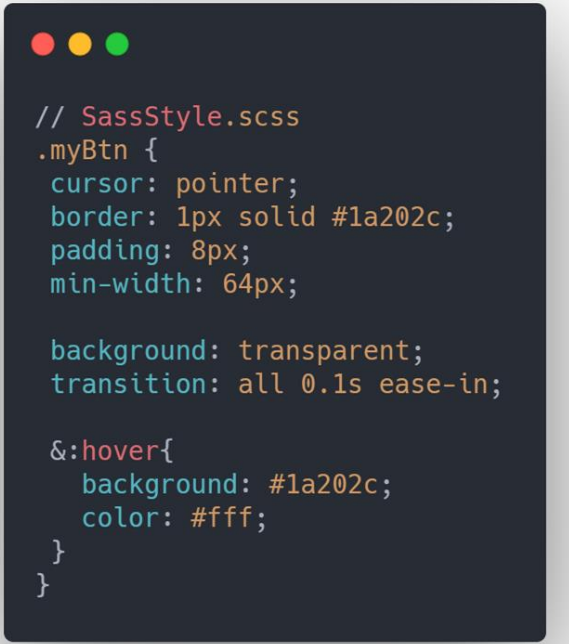
- Sass (Syntactically Awesome Style Sheets) is a CSS extension that gives you more powerful CSS.
- For example, you can define reusable CSS variables and you are able to nest your CSS.
- First install

```
npm install sass

yarn add sass
```

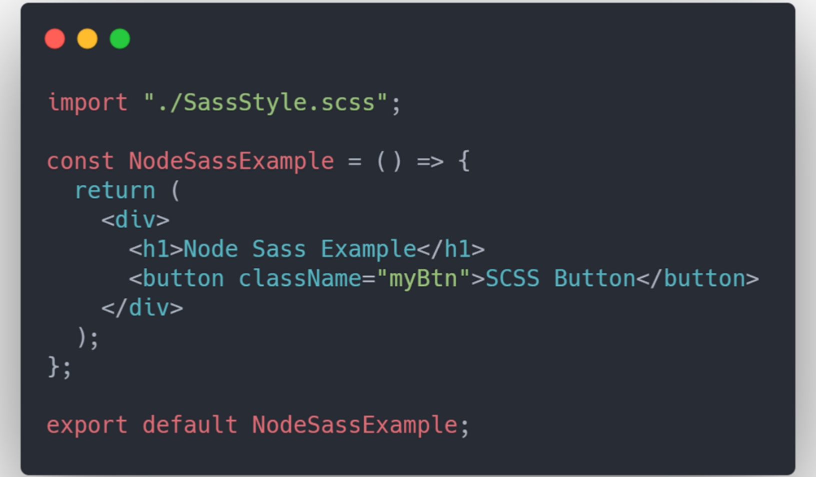
- Now you can rename `src/App.css` to `src/App.scss` and update `src/App.js` to import `src/App.scss`. This file and any other file will be automatically compiled if imported with the extension `.scss` or `.sass`.

CLARUSWAY
WAY TO REINVENT YOURSELF

d. SASS Example



```
// SassStyle.scss
.myBtn {
 cursor: pointer;
 border: 1px solid #1a202c;
 padding: 8px;
 min-width: 64px;

 background: transparent;
 transition: all 0.1s ease-in;

 &:hover{
   background: #1a202c;
   color: #fff;
 }
}
```



```
import "./SassStyle.scss";

const NodeSassExample = () => {
  return (
    <div>
      <h1>Node Sass Example</h1>
      <button className="myBtn">SCSS Button</button>
    </div>
  );
};

export default NodeSassExample;
```

CLARUSWAY
WAY TO REINVENT YOURSELF

THANKS!

Any questions?