In this guide, you will learn how to set up a site with the CSS-in-JS library Styled Components.

Styled Components lets you use actual CSS syntax inside your components. Styled Components is a variant on "CSS-in-JS"—which solves many of the problems with traditional CSS.

One of the most important problems they solve is selector name collisions. With traditional CSS, you have to be careful not to overwrite CSS selectors used elsewhere in a site because all CSS selectors live in the same global namespace. This unfortunate restriction can lead to elaborate (and often confusing) selector naming schemes.

With CSS-in-JS, you avoid all that as CSS selectors are scoped automatically to their component. Styles are tightly coupled with their components. This makes it much easier to know how to edit a component's CSS as there's never any confusion about how and where CSS is being used.

First, open a new terminal window and run the following to create a new site:

```
gatsby new styled-components-tutorial https://github.com/gatsbyjs/gatsby-starter-hello-world
cd styled-components-tutorial
```

Second, install the necessary dependencies for `styled-components`, including the Gatsby plugin.

```
npm install gatsby-plugin-styled-components styled-components babel-plugin-styled-components
```

And then add it to your site's `gatsby-config.js`:

```
module.exports = {
  plugins: [`gatsby-plugin-styled-components`],
}
```

Then in your terminal run `gatsby develop` to start the Gatsby development server.

Now create a sample Styled Components page at `src/pages/index.js`:

```
import React from "react"
import styled from "styled-components"

const Container = styled.div`
  margin: 3rem auto;
  max-width: 600px;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
`

const UserWrapper = styled.div`
  display: flex;
  align-items: center;
  margin: 0 auto 12px auto;
  &:last-child {
```

```
    margin-bottom: 0;
  }
`

const Avatar = styled.img`
  flex: 0 0 96px;
  width: 96px;
  height: 96px;
  margin: 0;
`

const Description = styled.div`
  flex: 1;
  margin-left: 18px;
  padding: 12px;
`

const Username = styled.h2`
  margin: 0 0 12px 0;
  padding: 0;
`

const Excerpt = styled.p`
  margin: 0;
`

const User = props => (
  <UserWrapper>
    <Avatar src={props.avatar} alt="" />
    <Description>
      <Username>{props.username}</Username>
      <Excerpt>{props.excerpt}</Excerpt>
    </Description>
  </UserWrapper>
)

export default function UsersList() {
  return (
    <Container>
      <h1>About Styled Components</h1>
      <p>Styled Components is cool</p>
      <User
        username="Jane Doe"

avatar="https://s3.amazonaws.com/uifaces/faces/twitter/adellecharles/128.jpg"
        excerpt="I'm Jane Doe. Lorem ipsum dolor sit amet, consectetur adipisicing
elit."
      />
      <User
        username="Bob Smith"
        avatar="https://s3.amazonaws.com/uifaces/faces/twitter/vladarbatov/128.jpg"
        excerpt="I'm Bob smith, a vertically aligned type of guy. Lorem ipsum dolor
```

```
sit amet, consectetur adipisicing elit."
      />
    </Container>
  )
}
```

Creating Global Styles

Styled-components are primarily used for a single CSS class that is isolated from other components. In some cases, you want to override global styling — for example, the default margins of your `body` element. Styled-components has your back. You can use the `createGlobalStyle` to accomplish this. It's advised to use `createGlobalStyle` in Layout components, which are shared over multiple pages rather than using it on a single page.

The example below shows how to create a `GlobalStyle` (which is a StyledComponent) for the color purple by importing `createGlobalStyle` from `styled-components`.

```
import React from "react"
import { createGlobalStyle } from "styled-components"

const GlobalStyle = createGlobalStyle`
  body {
    color: ${props => (props.theme === "purple" ? "purple" : "white")};
  }
`
export default function Layout({ children }) {
  return (
    <React.Fragment>
      <GlobalStyle theme="purple" />
      {children}
    </React.Fragment>
  )
}
```

Enabling user stylesheets with a stable class name

Adding a persistent CSS `className` to your styled components can make it easier for end users of your website to take advantage of user stylesheets for accessibility.

Here's an example where the class name `container` is added to the DOM along with the Styled Components' dynamically-created class names:

```
import React from "react"
import styled from "styled-components"

const Section = styled.section`
  margin: 3rem auto;
  max-width: 600px;
`
```

```
export default function Container({ children }) {
  return <Section className={`container`}>{children}</Section>
}
```

An end user of your site could then write their own CSS styles matching HTML elements using a class name of `.container`. If your CSS-in-JS style changes, it will not affect the end user's stylesheet.

```
.container {
  margin: 5rem auto;
  font-size: 1.3rem;
}
```