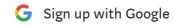
Theming React With CSS variables

How to theme a React application using CSS custom properties and hooks





Get one more story in your member preview when you sign up. It's free.



f Sign up with Facebook

Already have an account? Sign in

. . .

Getting Started

As always, let's start with <code>create-react-app</code>. We can bootstrap our new application as follows:

```
npx create-react-app css-theming
```

For this demo, we won't need to install any other dependencies. React, react-dom, and react-scripts will be enough.

Open the new application in your favourite editor.

Let's start by declaring some CSS variables in App.css.

```
:root {
   --color-surface: white;
   --color-solid: black;
   --color-primary: teal;
}
```

Next, we create some basic styles for our app. Notice how we consume our variables with the syntax var(--variable-name). This should be familiar to you if you are used to using PostCSS custom properties.

```
html {
background: var(--color-surface);

color: var(--color-solid):

Get one more story in your member preview when you sign up. It's free.

Gign up with Google

Sign up with Facebook

Already have an account? Sign in
```

Finally, let's modify our App.js to include our button.

We are ready to use JavaScript to modify our custom properties.

. . .

Modifying Custom Properties With JavaScript

The syntax for modifying custom properties is quite simple.

Getting a variable's value

```
window
  .getComputedStyle(document.documentElement)
  .getPropertyValue("--color-surface");
```

Setting a variable's value

Get one more story in your member preview when you sign up. It's free.

Sign up with Google

Sign up with Facebook

Already have an account? Sign in

```
"--color-solid": "black",
"--color-surface": "white",
"--color-primary": "teal"

const darkTheme = {
"--color-solid": "white",
"--color-surface": "black",
"--color-primary": "purple"
};

theme.js hosted with \(\sigma\) by GitHub

view raw
```

Next we modify our app's render function to toggle its theme, using usestate. We keep track of our current theme, and we update it in the onclick handler.

```
const App = () => {
       const [currentTheme, setTheme] = React.useState("light");
     const onClick = () => {
         const nextTheme = currentTheme === "light" ? "dark" : "light";
         setTheme(nextTheme);
       };
 7
     return (
        <div className="App">
8
           <h1>{currentTheme === "light" ? "Light theme" : "Dark theme"}</h1>
           <button onClick={onClick}>Toggle theme</button>
10
11
       );
13
     };
app.js hosted with ♥ by GitHub
                                                                                             view raw
```

Then we create a function which updates our CSS variables.

```
const applyTheme = nextTheme => {
    const theme = nextTheme === "dark" ? lightTheme : darkTheme;

Get one more story in your member preview when you sign up. It's free.

Sign up with Google

Sign up with Facebook

Already have an account? Sign in
```

Finally, we add a call to our applyTheme function in our button's onClick handler.

```
const onClick = () => {
  const nextTheme = currentTheme === "light" ? "dark" : "light";
  setTheme(nextTheme);
  applyTheme(nextTheme);
};
```

And there you have it — We've created a theme which can be configured using JavaScript. Let's see it in action.

Light theme

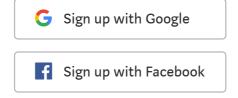


I may not win any awards for the design, but at least it works as expected.

. . .

Adding Tests

Get one more story in your member preview when you sign up. It's free.



Already have an account? Sign in

X

```
exherriberrate right rheme \1. coneinthenormment()
       });
     });
9
     describe("when toggling the theme", () => {
       it("should update theme", () => {
11
         const { getByText } = render(<App />);
12
13
         const button = getByText("Toggle theme");
         button.click();
         expect(getByText("Dark theme")).toBeInTheDocument();
15
16
       });
17
     });
testing.library.js hosted with ♥ by GitHub
                                                                                               view raw
```

Here, we test that when the app starts, it defaults to the light theme, and displays the text "Light theme" in the document.

Next, we click on our button to toggle the theme, and test to make sure that the text in our app updates.

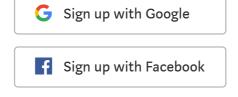
While these tests are good, and provide some value, they don't do anything to make sure that our CSS variables have changed.

To test this, let's mock one of the methods that we used to set the CSS variables.

We take the function document.documentElement.style.setProperty and replace it with a jest.fn() mock.

```
it("should update the css variables", () => {
   const setProperty = jest.fn();
   document.documentElement.style.setProperty = setProperty;
   const { getByText } = render(<App />);
   const button = getByText("Toggle theme");
   button.click();
   expect(setProperty).toHaveBeenCalledTimes(3);
```

Get one more story in your member preview when you sign up. It's free.



Already have an account? Sign in

X

. . .

Conclusion

We were able to set up theming in minutes, without adding any extra dependencies, or changing our build setup.

We can use these variables all over our CSS, and our React components don't need to know about them.

And finally, we were able to set up some tests that will give us confidence in our app's functionality.

Check out the code on code sandbox to see it in action.

css-theming - CodeSandbox

This project was bootstrapped with [Create React App](https://github.com/facebook/create-react-...

codesandbox.io

Thanks for reading!

React JavaScript CSS Design Programming

About Help Legal

Get one more story in your member preview when you sign up. It's free.

G Sign up with Google

f Sign up with Facebook

Already have an account? Sign in

X