Defining and Using Theme Props

In this lesson we'll discuss how to define our themes and how to use a theme's props in a simple example.

WE'LL COVER THE FOLLOWING

Having CSS variables change values between different themes is the main way by which our theming platform will work. This lesson includes a simple example that toggles between a light and a dark theme of an app. This example will also show what we mean by *defining a theme* and how we'll use the theme props in our app.

Example: Toggling between dark and light mode

Let's assume our themes have 2 props, `--background` (background color) and `--color` (text color), and that we have 2 themes we want to define, *light* and *dark*.

It's easy to switch themes by assigning a unique CSS class for each theme and attaching the theme we want to activate on the HTML element. This, in turn, applies the CSS variables to the target container of the theme, a notion we'll talk more about later.

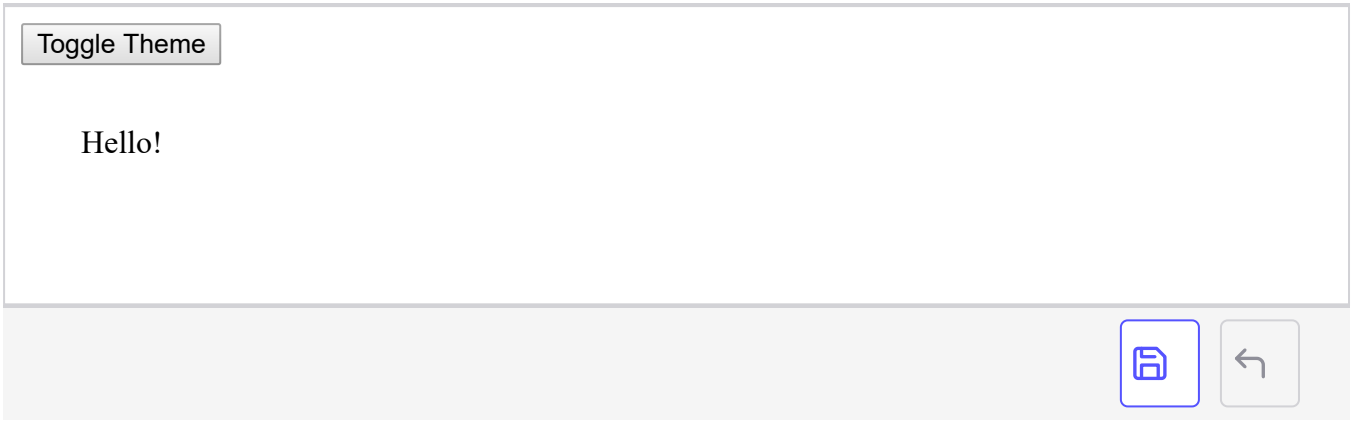
The most basic example of this is:

Output

JavaScript

HTML

CSS (SCSS)



Let's start with the SCSS. We first define our theme variables:

```
// Definition of our light theme:
.theme-light body {
    --background: white;
    --color: black;
}

// Definition of our dark theme:
.theme-dark body {
    --background: black;
    --color: white;
}
```

This is how we define our themes and their props. It's that simple! It's the act of defining CSS variables that we'll be using throughout our app. We usually target the **body** element that's inside a `theme-xxx` class.

> **Note:** The *target* of a theme (in this case, the **body** element) can, of course, be anything. As we'll see later when we talk about multiple theme categories, we'll be using other targets.

Next, we're making it so that clicking the button toggles between our two defined themes. This is easy! We just add the new theme's CSS class name and remove the previous one:

```
var currentTheme = 'theme-light';

button.addEventListener('click', () => {
  // We're just toggling between 2 themes here: 'theme-light' and 'theme-d
```

```
ark'

  var prevTheme = currentTheme;
  currentTheme = currentTheme == 'theme-light' ? 'theme-dark' : 'theme-lig
ht';

  // Remove the previous theme from the html class list
  document.documentElement.classList.remove(prevTheme);

  // Add the new theme name to the html class list
  document.documentElement.classList.add(currentTheme);
});
```

All that's left is to use the theme props we defined wherever we need them:

```
.root {
    background: var(--background);
    color: var(--color);
}
```

This is a simplified but functioning theming system! Everything else builds on top of this, and while an actual implementation might be more complex, the basic idea of how it works is shown in this example.

We already have a working theme switcher in just a few lines of code! But a theme is much more than this. Starting in the next lesson, we'll be talking about all the additional components of our themes. Let's continue!