What is Theming in CSS?

Theming in CSS refers to the practice of defining a set of visual styles, such as colors, fonts, and spacing, that apply across a website or application. Different themes allow for variations in appearance, like a "light" and "dark" mode, by applying different sets of style properties.

With Sass, theming becomes more efficient because you can use variables, mixins, and functions to define and switch between different themes without duplicating code.

Implementing Theming with Sass

In Sass, theming can be done by defining variables for colors, backgrounds, and other style properties. You can then create different themes (e.g., light and dark) by changing the values of these variables and switching between them dynamically, such as through a toggle in the UI.

Example: Light and Dark Theme Switcher Using Sass

We’ll create a theme switcher that allows users to toggle between a light and dark theme using Sass variables.

1. Sass Code for Light and Dark Themes

scss

Copy code

// \_variables.scss (Default theme variables)

$primary-color: #3498db;

$background-color: #ffffff;

$text-color: #333333;

// \_light-theme.scss (Light theme overrides)

$primary-color-light: #3498db;

$background-color-light: #ffffff;

$text-color-light: #333333;

// \_dark-theme.scss (Dark theme overrides)

$primary-color-dark: #8e44ad;

$background-color-dark: #1e1e1e;

$text-color-dark: #f1f1f1;

// \_theme.scss (Theme styles using variables)

body {

background-color: $background-color;

color: $text-color;

}

.button {

background-color: $primary-color;

color: white;

padding: 10px 20px;

border: none;

border-radius: 4px;

cursor: pointer;

text-align: center;

text-decoration: none;

&:hover {

background-color: darken($primary-color, 10%);

}

}

2. Toggling Themes

We will define two theme classes: .light-theme and .dark-theme, which will override the default variable values when applied to the <body> element.

scss

Copy code

// Theme switcher styles

.light-theme {

$primary-color: $primary-color-light;

$background-color: $background-color-light;

$text-color: $text-color-light;

}

.dark-theme {

$primary-color: $primary-color-dark;

$background-color: $background-color-dark;

$text-color: $text-color-dark;

}

3. Main Styles File

scss

Copy code

// styles.scss

@import 'variables';

@import 'light-theme';

@import 'dark-theme';

@import 'theme';

HTML Structure for Theme Switcher

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Theme Switcher</title>

<link rel="stylesheet" href="styles.css">

</head>

<body class="light-theme">

<div class="container">

<h1>Theme Switcher Example</h1>

<p>This is an example of a light and dark theme switcher using Sass.</p>

<button class="button" id="theme-toggle">Toggle Theme</button>

</div>

<script>

const toggleButton = document.getElementById('theme-toggle');

const body = document.body;

toggleButton.addEventListener('click', () => {

body.classList.toggle('dark-theme');

body.classList.toggle('light-theme');

});

</script>

</body>

</html>

Explanation:

Theme Variables:

The Sass variables for each theme (light and dark) are defined in separate partial files (\_light-theme.scss and \_dark-theme.scss).

These variables control properties such as the background color, text color, and button color for each theme.

Body Class Switch:

The <body> element is initially set to the light-theme class, which applies the light theme styles.

When the user clicks the "Toggle Theme" button, JavaScript toggles the dark-theme and light-theme classes on the <body>, switching between themes dynamically.

CSS Output:

Based on which theme is applied, the styles for the background color, text color, and button color will change.

JavaScript Toggle:

The JavaScript function listens for clicks on the toggle button and switches between the light-theme and dark-theme classes.

Compiled CSS Output

After compiling the Sass code into CSS, the resulting styles will look something like this:

css

Copy code

/\* Light Theme \*/

body.light-theme {

background-color: #ffffff;

color: #333333;

}

body.dark-theme {

background-color: #1e1e1e;

color: #f1f1f1;

}

.button {

background-color: #3498db;

color: white;

padding: 10px 20px;

border: none;

border-radius: 4px;

cursor: pointer;

text-align: center;

text-decoration: none;

}

.light-theme .button {

background-color: #3498db;

}

.dark-theme .button {

background-color: #8e44ad;

}

.button:hover {

background-color: #2980b9;

}

Default Styles: Styles for the .button and other elements are defined, and their appearance will change based on which theme class (light-theme or dark-theme) is applied to the <body>.

Hover Effects: The button color darkens on hover using the darken() function.

Advantages of Using Sass for Theming

Variables: You can define theme-specific variables for colors, fonts, and spacing, and reuse them throughout the project. This centralizes theme-related styles and makes it easy to change the theme by simply switching variable values.

Maintainability: By separating theme-specific values into partial files, you can easily update themes without touching the core styles.

Reusable Components: Sass allows you to create components (e.g., buttons, cards) that are flexible and can easily adapt to different themes without rewriting styles for each theme.

Dynamic Switching: With Sass, you can predefine light and dark themes, then use JavaScript or CSS custom properties to switch themes dynamically in the UI.

Summary

Theming in CSS allows you to switch between different sets of styles (e.g., light and dark mode) across a website or app.

Sass makes theming more efficient by using variables and mixins to define and manage theme styles, which can be dynamically switched in the UI.

In this example, we created a theme switcher using Sass variables to manage both light and dark themes, and used JavaScript to toggle between them in the browser.