**A Beginner's Guide to Theming with Custom Colors in Tailwind CSS v4**

**Author:** Gemini  
**Date:** July 26, 2025  
**Theme:** Based on the *"Portfolio Design"* Figma Community File

**Introduction**

Welcome to the world of Tailwind CSS v4! If you're new to Tailwind, it's a utility-first CSS framework that enables you to build modern websites rapidly, directly within your HTML.

Version 4 introduces a significant change: a **CSS-first configuration**. Instead of managing your theme in a JavaScript file (tailwind.config.js), you now do everything directly inside your CSS. This approach is more intuitive and aligns better with standard web development practices.

This guide will walk you through implementing the professional color palette from the Figma portfolio design into a theme-aware website with both **light** and **dark modes**.

**Prerequisites**

Before we begin, ensure you have:

* **Node.js** and **npm** (or another package manager) installed
* A **code editor** like Visual Studio Code
* Basic understanding of **HTML**, **CSS**, and **JavaScript**
* A project set up with **Vite** and **Tailwind CSS v4**  
  *(Follow the official Tailwind CSS v4 installation guide for Vite)*

**1. Understanding the Core Concepts**

In Tailwind v4, two new concepts are key to theming:

**1. The @theme Directive**

Think of @theme as your new configuration file. This is where you define **design tokens**—the foundational values for your design system. By defining colors as CSS variables within @theme, Tailwind can generate utility classes like bg-primary-blue.

**2. The @custom-variant Directive**

Tailwind's default dark: variant detects if the user’s OS is set to dark mode. Using @custom-variant, you can override this to give users a manual toggle—activating dark: utilities whenever a .dark class is present, thus providing full control over themes.

**2. Step-by-Step Tutorial**

Let's implement the color palette based on the Figma design.

**Step 1: Define Your Color Palette in @theme**

Open your main CSS file (e.g., src/index.css) and define your custom color utilities:

css

/\* src/index.css \*/

@import "tailwindcss";

@theme {

/\* Define your design tokens here. Tailwind generates classes like 'bg-primary-blue', 'text-accent-yellow', etc. \*/

--color-primary-blue: var(--primary-blue);

--color-accent-yellow: var(--accent-yellow);

--color-text-primary: var(--text-primary);

--color-text-secondary: var(--text-secondary);

--color-background: var(--background);

--color-surface: var(--surface); /\* For cards, modals, etc. \*/

}

**Step 2: Set Up Light & Dark Mode Colors**

Below your @theme block, tell Tailwind how to handle dark: variants and define your theme colors:

css

/\* src/index.css \*/

/\* ... your @theme block from above ... \*/

/\* --- Inform Tailwind to look for '.dark' class to activate dark mode --- \*/

@custom-variant dark (&:where(.dark, .dark \*));

/\* --- Light Mode Colors --- \*/

:root {

--primary-blue: #6344F5;

--accent-yellow: #F5D444;

--text-primary: #282828; /\* Dark Gray \*/

--text-secondary: #5E5E5E; /\* Lighter Gray \*/

--background: #F5F5F5; /\* Off-white \*/

--surface: #FFFFFF; /\* White for surfaces \*/

}

/\* --- Dark Mode Colors --- \*/

.dark {

--primary-blue: #6344F5; /\* Same as light for consistency, or change if needed \*/

--accent-yellow: #F5D444;

--text-primary: #EAEAEA; /\* Light Gray for dark mode text \*/

--text-secondary: #B4B4B4;

--background: #0D0D0D; /\* Near black background \*/

--surface: #1A1A1A; /\* Slightly lighter for cards in dark mode \*/

}

**Step 3: Use Utility Classes in HTML**

Apply the custom utility classes in your HTML for elements:

html

<!-- index.html -->

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

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

<title>Portfolio Theming with Tailwind v4</title>

<link rel="stylesheet" href="/src/index.css" />

</head>

<body class="bg-background text-text-primary transition-colors duration-300 ease-in-out">

<div class="flex flex-col items-center justify-center min-h-screen p-8">

<div class="w-full max-w-md text-center">

<h1 class="text-4xl font-bold text-primary-blue">Portfolio Design Theme</h1>

<p class="mt-2 text-text-secondary">This design adapts to light and dark modes.</p>

<div class="mt-8 p-8 bg-surface rounded-xl shadow-lg text-left">

<h2 class="text-2xl font-semibold text-accent-yellow">About Me</h2>

<p class="mt-2 text-text-secondary">

This card uses the 'surface' background and 'accent-yellow' for the title.

</p>

<button id="theme-toggle" class="mt-6 w-full px-4 py-3 bg-primary-blue text-white rounded-lg font-semibold hover:opacity-90 transition-opacity">

Toggle Theme

</button>

</div>

</div>

</div>

<script type="module" src="/src/main.js"></script>

</body>

</html>

**Step 4: Add Theme Toggle Logic in JavaScript**

Create src/main.js with the theme toggle functionality:

js

// src/main.js

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

const setTheme = (theme) => {

if (theme === 'dark') {

document.documentElement.classList.add('dark');

localStorage.setItem('theme', 'dark');

} else {

document.documentElement.classList.remove('dark');

localStorage.setItem('theme', 'light');

}

};

// Load saved theme or check preference on page load

const savedTheme = localStorage.getItem('theme');

const prefersDark = window.matchMedia('(prefers-color-scheme: dark)').matches;

if (savedTheme) {

setTheme(savedTheme);

} else if (prefersDark) {

setTheme('dark');

} else {

setTheme('light');

}

// Toggle theme on button click

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

const isDark = document.documentElement.classList.contains('dark');

setTheme(isDark ? 'light' : 'dark');

});

**Conclusion**

Congratulations! You've integrated a professional, Figma-inspired color palette into a theme-aware website using Tailwind CSS v4's CSS-first configuration. Using semantic color names and CSS variables ensures your design is both maintainable and flexible.

Happy coding!