Chapter #10: Tailwind in Action

In this chapter, we will learn how to style our application using Tailwind CSS.

Ways to style react components -

***External CSS File -***

All the CSS styles for components are written in a single central file, such as index.css. This file is then imported into your project, and styles are applied using class names.

***External SCSS File -***

Similar to CSS, but using SCSS (Sass) syntax. All styles are written in a central file like index.scss, which allows the use of variables, nesting, and other Sass features. The SCSS file is then compiled to CSS and imported into the project.

***Inline CSS -***

Styles are defined as JavaScript objects inside your component and applied directly to elements via the style attribute in JSX. This method allows dynamic styling using JavaScript variables.

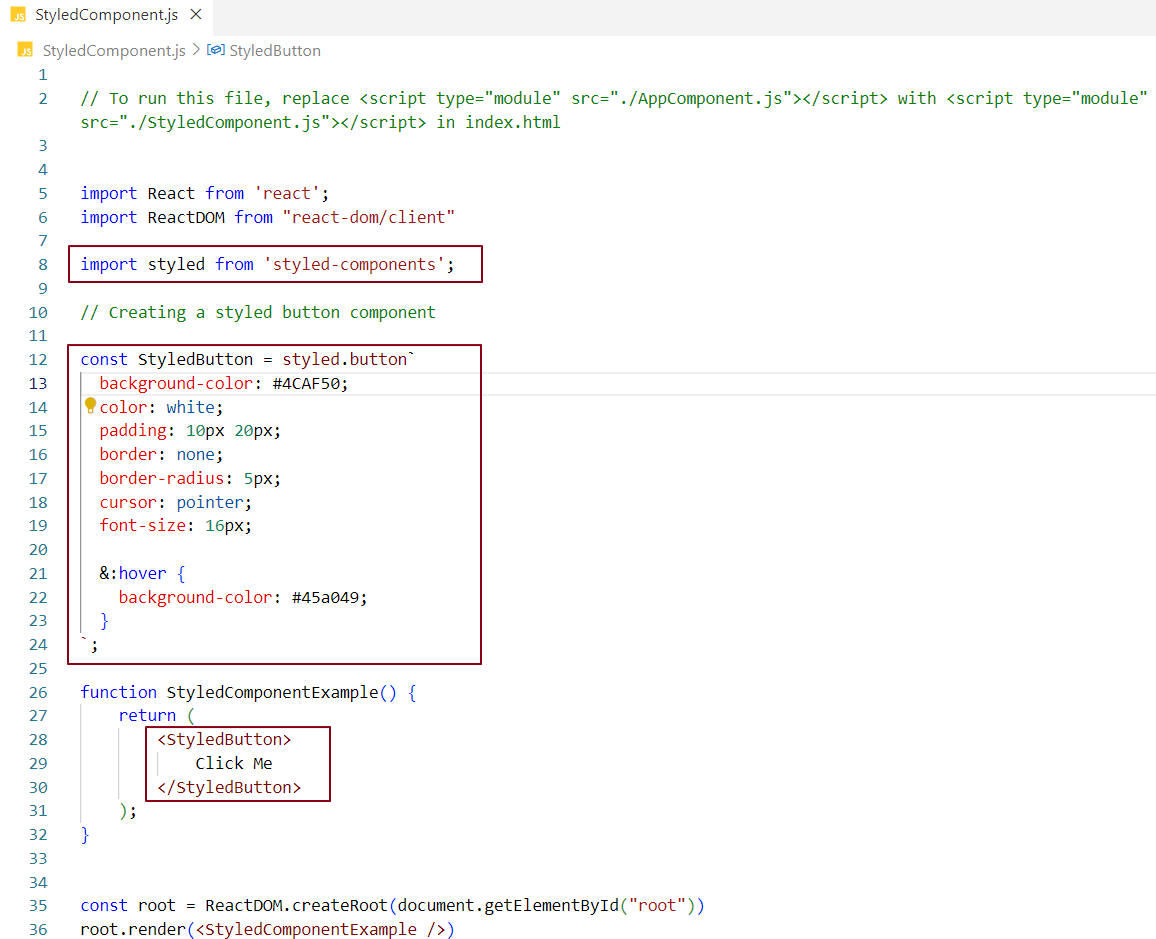


***Styling Libraries -***

Libraries such as Material UI and Ant Design offer a collection of pre-styled, ready-to-use React components along with styling utilities. These libraries help you build consistent and professional-looking user interfaces quickly without writing all styles from scratch.

***Styled Components -***

Styled Components allow you to write CSS directly inside JavaScript, similar to how JSX lets you write HTML-like code within JavaScript. This approach makes it easy to create reusable components with their own styles.



**1. Importing styled-components -** We import the styled function from the styled-components library to create styled elements in React.



**2. Creating Styled Components -** You can create styled elements by using styled.<elementName>, where <elementName> is any valid HTML tag (e.g., button, div, section, etc.). Then, you define the styles for those elements.

Syntax: const StyledElement = styled.button` /\* CSS styling here \*/ `;



Here, StyledButton is a styled <button> element. You can replace button with any other HTML tag to create different styled components.

**3. Using Styled Components -** Once you create a styled component, you can use it in your JSX just like any other React component. This lets you apply styles directly within your component.

In this example, StyledButton is a styled component that can be reused throughout your application.

Advantages and Disadvantages of Using Styling Libraries and Frameworks

***Advantages -***

* Easy to use and promotes reusability of components.
* Saves significant development time, speeding up the build process.
* Provides built-in themes that can be applied automatically.
* Ensures a consistent user interface across the application.  
  *(A consistent UI means all elements, like buttons and textboxes, share the same style, enhancing the overall user experience. It avoids having different button styles on different pages, maintaining uniformity throughout the app.)*
* Handles responsiveness, making sure the UI works well on all device sizes.

***Disadvantages -***

* Can increase the overall bundle size, making the app heavier.
* Limits design customization. For example, using Material UI buttons everywhere can make it harder to apply custom styles without extra effort. While development is faster, deep customization may take more time.
* Challenging to integrate into heavily customized or legacy front-end projects.
* May be unnecessarily complex for simple or small projects.
* Updates to the library can occasionally introduce bugs or breaking changes.
* Some features may not be fully supported on older devices or certain mobile platforms.

How Did We Use to Style Our Components?

In the past, we maintained a single CSS file - usually named index.css where all the styles for every component in the project were written. This file acted as the central place for styling. We then included index.css in our main index.html file to apply those styles across the entire application.

However, this approach had several drawbacks.

***Disadvantages -***

* All component styles were combined in one large file (index.css), which grew larger with every update.
* As the file size increased, maintaining and testing the CSS became difficult.
* The CSS was often not optimized, leading to unnecessary styles being loaded.
* This method reduced reusability since sharing styles between components was not straightforward.
* It often resulted in hardcoded styling, making flexibility and customization harder.
* Processing large amounts of inline CSS or unstructured styles can be inefficient for browsers.

To overcome these challenges, Tailwind CSS offers a better, more efficient solution.

What is Tailwind CSS?

Tailwind is an open-source CSS framework that allows us to write CSS directly within the same component JSX file, similar to inline styling. This eliminates the need to switch between the CSS file and the component JSX file.

***Perks / Why Choose Tailwind -***

* CSS on the go: Tailwind enables faster development by allowing us to write styles directly in the JSX, saving time.
* Reusability: Tailwind promotes the reuse of utility classes across components.
* Smaller bundle size: Tailwind provides minimal, optimized CSS, resulting in smaller bundle sizes.
* Flexibility and customization: Unlike Material UI, Tailwind offers greater flexibility and customization.
* Prebuilt classes: It provides a wide range of utility classes, reducing the need to write custom CSS.
* Less code and easier debugging: Since styles are defined in JSX, there’s less code to manage, making it easier to debug.
* No duplicate CSS: Tailwind’s bundler automatically removes duplicate CSS, preventing redundancy.
* Dynamic styling: It supports dynamic styling with custom CSS values.

***Disadvantages of Tailwind -***

* Initial learning curve: It may be challenging for new developers to learn, as it introduces a new way of styling.
* Code readability: Using many utility classes to style an element can make the code harder to read and cluttered.

Note: Inline styling and other styling methods each have their drawbacks. Tailwind inherits only the beneficial aspects of these approaches. For instance, "CSS on the go" is a feature that Tailwind borrowed from inline styling, enabling faster development.

How to include Tailwind into our project?

**Method 1:** Use the CDN link to allow Tailwind to control the styling of your UI elements and components.



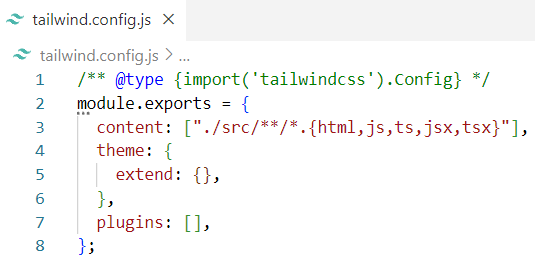
**Method 2 - Tailwind Installation with Parcel:** [Tailwind Installation Guide](https://tailwindcss.com/docs/guides/parcel)

* Install Tailwind CSS by running the command: npm install -D tailwindcss
* After installing Tailwind, install PostCSS using the command: npm install -D postcss

***Why PostCSS ?***

PostCSS is needed because it allows Parcel to understand that we are using Tailwind, so it can compile the Tailwind class properties into CSS that browsers can interpret.

* Configure Tailwind by running the command: npx tailwindcss init

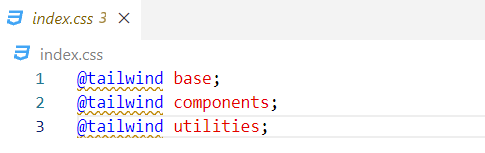


In the content section, we specify a regular expression for the files where Tailwind will be used. This configuration ensures that the specified files can use Tailwind's CSS.

* Create a new file named **.**postcssrc in the project root directory.  
  The **.**postcssrc file should contain the following configuration, which tells Parcel that our project will use many Tailwind classes. This ensures that when bundling, Parcel compiles Tailwind's CSS into regular CSS.



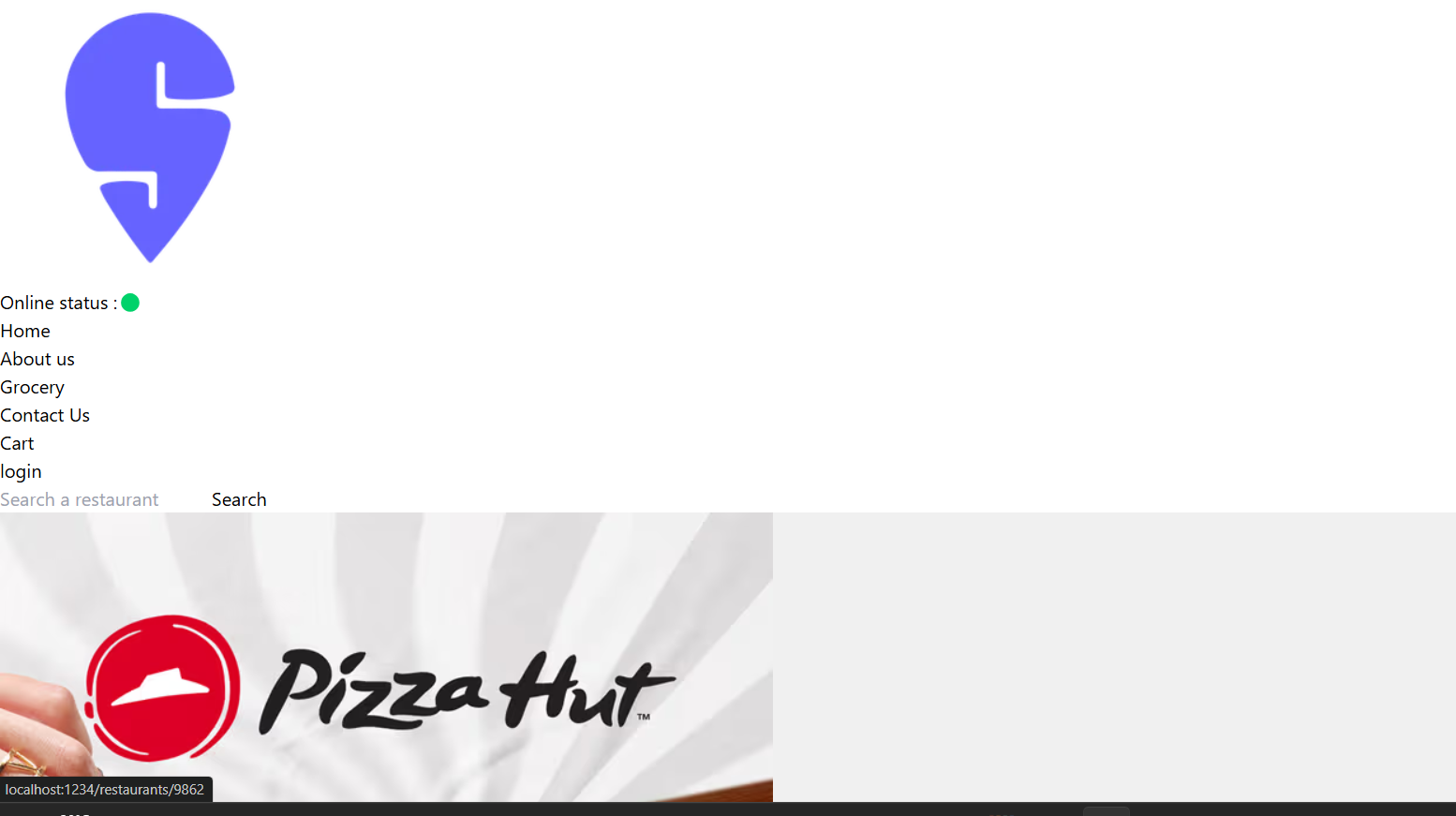
* Add the following three lines to index.css. This ensures that when Parcel processes index.css, it knows that Tailwind is being used for base components and utilities in the project.



We're done! Now, we can use Tailwind classes throughout our components.

Tailwind controlling UI elements -

When we run npm start, Tailwind styles will override the default HTML styles of the UI.



Note –- Tailwind provides a className for every CSS rule we want to apply inside our app.

Resources and Home work -

***Resources -***

* Styled Components documentation: <https://styled-components.com/docs/basics>
* Tailwind documentation: <https://v2.tailwindcss.com/docs>
* VS Code Extension for Tailwind: Tailwind CSS IntelliSense

***Homework: -***

Make your app look good using Tailwind classes.

**Answer:** I have already implemented this in my project. Please refer to the "NamasteReact\_Tailwind\_In\_Action" project on GitHub.

After using Tailwind in my project, the UI looks like this -

