Creating an Angular Component using Angular CLI

In this guide, we will walk you through the process of creating an Angular component using the Angular CLI. The Angular CLI (Command Line Interface) is a powerful tool that helps you scaffold, generate, and manage your Angular application's code.

Prerequisites

Before you begin, make sure you have the following installed on your system:

- Node.js: You can download and install Node.js from the official website: https://nodejs.org
- Angular CLI: Install the Angular CLI globally by running the following command in your command-line interface:

  ```
  npm install -g @angular/cli
  ```

Once you have the prerequisites installed, you can proceed with creating your Angular component.

Step 1: Generate the Component

1. Open your command-line interface and navigate to the directory where you want to create your Angular component.
2. Run the following command to generate the component using the Angular CLI:

   ```
   ng generate component component-name
   ```

   Replace `component-name` with the desired name for your component. Make sure to use kebab-case (all lowercase with hyphens) for the component name.

   The Angular CLI will create a new directory with the component files and add them to your project's file structure.
3. Once the command completes, you will see a success message indicating that the component has been generated.

Step 2: Explore the Generated Files

The Angular CLI will generate several files for your component:

- `component-name.component.ts`: This file contains the TypeScript code for your component's logic and functionality.
- `component-name.component.html`: This file contains the HTML template for your component's UI.
- `component-name.component.css` (optional): This file contains the component-specific CSS styles.
- `component-name.component.spec.ts` (optional): This file contains the unit tests for your component.

Step 3: Customize the Component

1. Open the generated `component-name.component.ts` file in your preferred code editor.
2. Modify the component's TypeScript code to add your desired functionality. You can define properties, methods, event handlers, and more within the component class.
3. Open the `component-name.component.html` file and update the HTML template to define the structure and content of your component's UI.
4. If needed, you can also modify the `component-name.component.css` file to add custom CSS styles specific to your component.