Building a Common Navbar Component in Angular: A Step-by-Step Guide

The navigation bar, commonly referred to as the navbar, is an essential element in most web applications. It provides users with easy access to different sections of your app. In this blog post, we will walk you through the process of creating a common navbar component in Angular. This component can be reused throughout your application to maintain a consistent look and feel. Let's get started!

Table of Contents

1. Introduction to Navbar Component

A common navbar component simplifies the process of maintaining a consistent user interface across your Angular application. It typically includes elements like a logo, navigation links, and user-related actions such as login or logout buttons.

2. Setting Up Your Angular Project

Before we dive into creating the navbar component, make sure you have Angular CLI installed. If not, you can install it globally using npm:

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

Create a new Angular project with the following command:

```
ng new my-app
```

Navigate to your project directory:

```
cd my-app
```

3. Creating the Navbar Component

To create the navbar component, use the Angular CLI:

```
ng generate component navbar
```

This command generates the necessary files for your navbar component.

4. Designing the Navbar

Open the `navbar.component.html` file and design your navbar. Here's a simple example:

```
<nav class="navbar">
 <div class="navbar-logo">
   <img src="assets/logo.png" alt="My App Logo" />
 </div>
 <ul class="navbar-links">
   <li><a routerLink="/">Home</a></li>
   <li><a routerLink="/about">About</a></li>
<li><a routerLink="/about">Services</a></li>
<li><a routerLink="/about">About</a></li>
   <li><a routerLink="/contact">Blog</a></li>
 </ul>
 <div class="navbar-actions">
   <!-- Add login/logout buttons or user-related actions
here -->
 </div>
</nav>
```

In this example, we've added a logo, navigation links, and a placeholder for user-relatr application's needs.

5. Adding Navigation Links

To make the navigation links functional, you'll need to configure routing in your Angular application. Open the `app-routing.module.ts` file and define your routes:

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';

const routes: Routes = [
 { path: '', redirectTo: '/home', pathMatch: 'full' },
 { path: 'home', component: HomeComponent },
 { path: 'about', component: AboutComponent },
 { path: 'contact', component: ContactComponent },
];

@NgModule({
 imports: [RouterModule.forRoot(routes)],
 exports: [RouterModule]
})


export class AppRoutingModule { }
```

ed actions. You can customize this HTML structure and styles according to you

Make sure to import and configure the necessary components for the respective routes.

6. Using the Navbar in the App Component

To use the navbar component in your `app.component.html`, you can include the `<app-navbar></app-navbar>` tag at the top of your template:

```
<app-navbar></app-navbar>

<router-outlet></router-outlet>
```

7. Conclusion

By following these steps, you've created a common navbar component in Angular that can be used throughout your application. This approach promotes code reusability and consistency in your user interface.

Remember that this is just a starting point, and you can enhance your navbar component by adding features like responsive design, authentication, and dynamic navigation links. Angular provides a powerful framework for building complex web applications, and your common navbar component is just one piece of the puzzle. Happy coding!