Setting Up an Angular Project

Prerequisites

1. Node.js and npm (Node Package Manager)

Angular requires Node.js and npm to function. If you haven't installed them yet, download and install the latest version of Node.js from `https://nodejs.org`.

You can verify the installation by running the following commands in your terminal:

```
node -v
npm -v
```

2. Angular CLI

Angular CLI (Command Line Interface) is a tool that simplifies the process of creating, managing, and testing Angular applications.

To install Angular CLI globally, run the following command:

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

Verify the installation of Angular CLI:

```
ng version
```

Step 1: Create a New Angular Project

1. Creating the Project

In the terminal, navigate to the directory where you want to create your Angular project. For example:

```
cd /path/to/folder
```

Use the `ng new` command to create a new Angular project:

```
ng new your-project-name
```

This command will prompt you with a few questions:

- Do you want to use Angular Routing? If you need routing, answer `Yes`.

- Choose a stylesheet format: CSS, SCSS, Sass, Less, or Stylus. Select your preferred format (CSS by default).

After answering these questions, Angular CLI will create the project in a folder named after the specified project name.

2. Navigate to the Project Folder

Once the project is created, navigate to the project directory:

```
cd your-project-name
```

3. Start the Development Server

To ensure everything is working correctly, start the development server with the following command:

```
ng serve
```

This command will:

- Compile the project.
- Start a development server at `http://localhost:4200`.

You can now open your browser and access the Angular application by navigating to `http://localhost:4200`. If everything is set up correctly, you will see the default Angular homepage.

Step 2: Project Structure

An Angular project generated by Angular CLI has the following structure:

```
your-project-name/
 e2e/                          # End-to-end tests
 node_modules/                 # Project dependencies
 src/                          # Application source code
    app/                      # Components, services, etc.
    assets/                   # Static files (images, etc.)
    environments/             # Configuration files for different environments
    index.html                # Application entry point
    main.ts                   # Application entry file (TypeScript)
    styles.css                # Global stylesheet
 angular.json                  # Angular project configuration
 package.json                  # npm dependencies and scripts
 tsconfig.json                 # TypeScript configuration
```

Step 3: Add a Component

1. To create a new component, use the following command:

`ng generate component component-name`

Or the shorthand:

`ng g c component-name`

This command will generate the necessary files in the `src/app/component-name` folder:

- `component-name.component.ts` (the component's TypeScript file)
- `component-name.component.html` (the component's HTML template)
- `component-name.component.css` (the component's styles)
- `component-name.component.spec.ts` (the component's unit test file)

Step 4: Add a Service

Services are used to provide data and functionality in an Angular application.

1. To generate a service, use the following command:

`ng generate service service-name`

Or the shorthand:

`ng g s service-name`

This will create a service file in `src/app/service-name.service.ts`.

Step 5: Add Routes

To configure routes in Angular: 1. Create an `app-routing.module.ts` file if it doesn't already exist. 2. Define the routes and import the `RouterModule`.

Example:

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';

const routes: Routes = [
  { path: '', component: HomeComponent },
  { path: 'about', component: AboutComponent }
];

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

```
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Listing 1: Defining Routes in Angular

Step 6: Deploy the Application

Once your application is ready, you can build it for production by running the following command:

```
ng build --prod
```

This command will create an optimized version of the application in the `dist/` folder.

You can then deploy the files from this folder to your production server or a cloud platform such as `Firebase`, `AWS`, or `Netlify`.