What is Angular?

Angular is a popular open-source front-end web application framework developed by Google. It is built using **TypeScript**, a superset of JavaScript, and provides a platform for building single-page applications (SPAs) with dynamic user interfaces. Angular offers a comprehensive framework that includes tools and libraries for routing, state management, form handling, HTTP client communication, and more, making it a powerful choice for developing modern web applications.

Angular is a complete rewrite of its predecessor, AngularJS, and is often referred to as **Angular 2+** to distinguish it from AngularJS. Angular uses a component-based architecture, which promotes modularity and reusability, allowing developers to build complex applications more efficiently.

Angular Environment and IDEs

To develop applications using Angular, you need a development environment that includes a code editor, Node.js, and the Angular CLI (Command Line Interface).

1. Development Environment Setup:

- **Node.js and npm**: Angular development requires Node.js (Node) and the Node Package Manager **(npm),** which are used to install and manage dependencies, including Angular itself. You can download Node.js from the official website. When you install Node.js, npm is included by default.
- **Angular CLI**: The Angular CLI is a command-line tool that helps initialize, develop, scaffold, and maintain Angular applications. It

provides commands to generate components, services, modules, etc., and to build, test, and deploy applications.

2. Integrated Development Environments (IDEs):

Several IDEs and code editors are well-suited for Angular development:

- **Visual Studio Code (VS Code)**: A lightweight, open-source code editor from Microsoft with robust Angular and TypeScript support. It has numerous extensions that enhance Angular development, such as Angular Language Service, Angular Snippets, and Debugger for Chrome.
- **WebStorm**: A powerful IDE by JetBrains specifically designed for JavaScript development. It has built-in support for Angular, TypeScript, and various other web technologies, along with features like intelligent code completion, refactoring, and debugging.
- **Atom**: A hackable text editor developed by GitHub, which supports Angular development with plugins for TypeScript and Angular snippets.
- **Sublime Text**: A popular text editor with plugins available for Angular development, including AngularJS and TypeScript plugins.
- **Angular Console**: A visual tool that integrates with Angular CLI, providing a graphical user interface (GUI) for executing Angular CLI commands.

Installing Angular CLI and Creating a Sample Project

The Angular CLI simplifies the process of creating, building, testing, and deploying Angular applications.

1. Installing Angular CLI:

To install Angular CLI, follow these steps:

1. **Install Node.js**: If you haven't installed Node.js yet, download and install it from the Node.js website.
2. **Install Angular CLI**: Open a terminal or command prompt and run the following command:

   npm install -g @angular/cli

   The -g flag installs the Angular CLI globally, making it available across your system.

3. **Verify Installation:** After installation, verify that the Angular CLI is installed correctly by running:

   ng version

   This command should display the Angular CLI version along with other environment details.

2. Creating a Sample Project:

1. **Create a New Angular Project**: Use the Angular CLI to create a new project by running:

   ng new my-angular-app

   Replace my-angular-app with your desired project name. The CLI will prompt you to select various configuration options, such as whether to include Angular routing and which stylesheet format to use (**CSS, SCSS**, etc.).

npx -p@angular/cli@15 ng new app3 --standalone=false

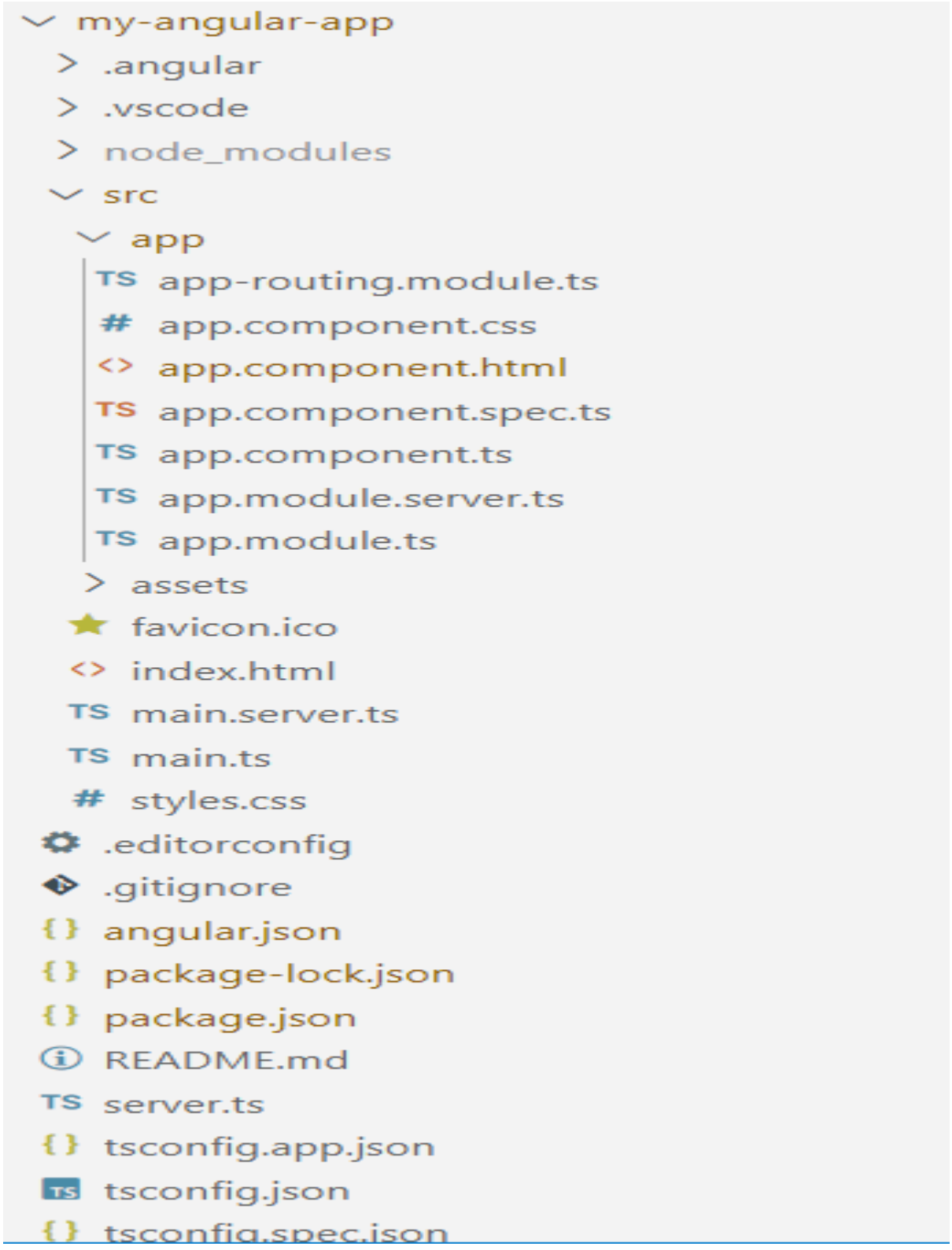
2. **Navigate to the Project Directory**: Once the project is created, navigate to the project directory:

   cd my-angular-app

3. **Serve the Application**: Start a development server to run your application locally:

   ng serve

   The Angular CLI will compile your project and start a development server. Open a web browser and go to **http://localhost:4200/** to see your application running.


my-angular-app
.angular
.vscode
node_modules
src
app
app-routing.module.ts
app.component.css
app.component.html
app.component.spec.ts
app.component.ts
app.module.server.ts
app.module.ts
assets
favicon.ico
index.html
main.server.ts
main.ts
styles.css
.editorconfig
.gitignore
angular.json
package-lock.json
package.json
README.md
server.ts
tsconfig.app.json
tsconfig.json
tsconfig.spec.json

Directory Structure and Various Configuration Files

When you create an Angular project using the CLI, it sets up a directory structure and configuration files necessary for development.

Directory Structure:

- **src/**: The source folder containing the application code.
  - **app/**: Contains the main application modules and components.
  - **assets/**: Stores static assets such as images, styles, and other files that need to be served.
  - **environments/**: Holds environment-specific configuration files (e.g., environment.prod.ts for production).
  - **main.ts**: The main entry point of the application, bootstrapping the root module.
  - **index.html**: The main HTML file that loads the Angular application.
- **angular.json**: The Angular workspace configuration file that defines the project settings, build, and serve options.
- **package.json**: Lists the dependencies and scripts for the Angular project. This file is used by npm to install packages.
- **tsconfig.json**: The TypeScript compiler configuration file, defining rules for how TypeScript code should be compiled.
- **karma.conf.js**: Configuration file for the Karma test runner, used to run unit tests.
- **tslint.json**: Linting configuration file that enforces code style and best practices in TypeScript.
- **node_modules/**: Contains all the npm packages installed for the application. This folder is auto-generated when dependencies are installed.

Configuration Files:

1. **angular.json**:
   - Configures the project, including build options, file paths, and default settings for generating components, modules, and more.

2. **package.json**:
    - Manages project dependencies and scripts. This file is crucial for defining the libraries required for your Angular application to run.
3. **tsconfig.json**:
    - Configures TypeScript compiler options such as target JavaScript version, module resolution, and inclusion/exclusion of files.
4. **tslint.json**:
    - Configures rules for TSLint, a static analysis tool that checks TypeScript code for readability, maintainability, and functionality errors.

Angular CLI Commands

The Angular CLI provides a variety of commands to create, develop, and maintain Angular applications. Here are some of the commonly used commands:

- **ng new <project-name>**: Creates a new Angular project.
- **ng serve**: Compiles the application and starts a local development server.
- **ng generate component <component-name>**: Generates a new component with the specified name.
- **ng generate service <service-name>**: Creates a new service.
- **ng generate module <module-name>**: Creates a new module.
- **ng build**: Compiles the application into an output directory (default: dist/). Can include flags for production builds.
- **ng test**: Runs unit tests using Karma.
- **ng lint**: Runs TSLint on the project.
- **ng e2e**: Runs end-to-end tests using Protractor.
- **ng add <library-name>**: Adds support for a library or tool to the project (*e.g., ng add @angular/material*).
- **ng update**: Updates Angular and its dependencies.

Installing Bootstrap and Styling Your Application

Bootstrap is a popular CSS framework that helps you design responsive web applications. You can easily integrate Bootstrap into your Angular application.

1. Installing Bootstrap:

Use **npm** to install Bootstrap:

```
npm install bootstrap@latest
```

2. Adding Bootstrap to Angular:

1. **Update angular.json**: Add Bootstrap's CSS file to the styles array in the **angular.json** file.

```
"styles": [
            "src/styles.css",
            "node_modules/bootstrap/dist/css/bootstrap
.min.css"
          ],
```

2. **Use Bootstrap Classes:** You can now use Bootstrap classes in your Angular components to style them.

For example, in your **app.component.html**:

```
<div class="container">
  <h1 class="text-center">Welcome to My Angular App</h1>
</div>
```

Decorators in Angular

Decorators are a special kind of declaration in TypeScript and Angular that can be attached to a class, property, method, or accessor to modify its behavior. Angular uses decorators extensively to define components, modules, services, and other elements.

Common Angular Decorators:

1. **@Component**: Defines a component in Angular. It includes metadata such as the component's selector, template URL, and styles.

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'myapp';
}
```

2. **@NgModule**: Defines an Angular module. It includes metadata such as the declarations, imports, providers, and bootstrap components.

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

@NgModule({
  declarations: [
    AppComponent,
    MyHelloWorldComponent
  ],
```

3. **@Injectable**: Marks a class as available to be injected as a dependency. This decorator is used for services.

```
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root',
})
export class DataService {
  constructor() { }
}
```

4. **@Directive:** Defines a directive, a class that allows you to attach behavior to elements in the DOM.

```
import { Directive, ElementRef, Renderer2 } from
'@angular/core';

@Directive({
  selector: '[appHighlight]'
})
export class HighlightDirective {
  constructor(el: ElementRef, renderer: Renderer2) {
    renderer.setStyle(el.nativeElement, 'backgroundColor',
'yellow');
  }
}
```

5. **@Pipe**: Defines a pipe, a class that allows you to transform input data for display in a template.

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({ name: 'capitalize' })
export class CapitalizePipe implements PipeTransform {
  transform(value: string): string {
    return value.charAt(0).toUpperCase() + value.slice(1);
  }
}
```

Decorators are a key feature of Angular that provide metadata for classes and help configure how they should behave in the Angular framework. They are essential for defining and linking the various parts of an Angular application.