Angular

Commands

npx @angular/cli@latest          # Install the latest Angular CLI

npx ng new DemoApp               # Create a new Angular application

npx ng version                   # Check Angular CLI version

npx ng g c home                  # Generate a new component named 'home'

npx ng g s homeService           # Generate a new service named 'homeService'

All files and logic should be written in **TypeScript**.

**Files Created in Angular Component**

When a component is generated, the following files are created:

* **home.component.ts** – Contains the component class logic.
* **home.component.html** – Template file for the component.
* **home.component.css/scss** – Styling for the component.
* **home.component.spec.ts** – Unit test file for the component.

**Custom Project Structure (Optional Enhancements)**

You can modularize your app with additional files:

* **app.config.ts** – Used to configure providers and global services.
* **app.routes.ts** – Centralized routing configuration for the application.

**Configuration Files**

* **.editorconfig** – Defines coding standards across the team.
* **angular.json** – Angular workspace configuration (build, serve, test, etc.).
* **package.json** – Lists project dependencies and scripts.
* **package-lock.json** – Locks the versions of installed packages for consistency.

**Bootstrapping Process**

Angular bootstraps the application in the following order:

index.html → main.ts → AppModule → AppComponent → Other Components

**Components**

* A component is like a class with:
  + **Properties** – Data
  + **Methods** – Logic
* Decorated with @Component decorator.

@Component({

  selector: 'app-home',

  templateUrl: './home.component.html',

  styleUrls: ['./home.component.css']

})

**Modules**

* Angular apps are modular.
* **NgModules** help organize an app into cohesive blocks.
* AppModule is the root module.

**Dependency Injection**

* Services are injected into components via **constructors**.
* Promotes modularity and testability.

**Data Binding**

**One-Way Binding**

* **Interpolation**: {{ value }}
* **Property Binding**: [property]="value"
* **Event Binding**: (event)="handler()"

**Two-Way Binding**

* Uses [(ngModel)] (requires FormsModule):

**Pipes**

* Used to transform data in templates.

**Built-in Pipes:**

* date, uppercase, lowercase, currency, percent, json

**Directives**

* Modify DOM behavior or appearance.

**Types:**

* **Structural**: \*ngIf, \*ngFor
* **Attribute**: [ngClass], [ngStyle]
* **Custom Directives**: You can create your own.

**Routing**

* Enables navigation between views/components.
* Defined in app-routing.module.ts
* Routes = []
* Routes.navigate
* routeLink

Lazy Loading = Loading parts of application only when they are needed

Guards: Restricting the access of the route based on the either the condition and other logic (Used on the dashboard where the token are required to access)

**Forms**

**Template-Driven Forms**

* Simpler, uses HTML directives
* Requires FormsModule

**Reactive Forms**

* More powerful and memory-efficient
* Requires ReactiveFormsModule
* Uses FormGroup, FormControl, and FormBuilder

Assignment

Create an application which has the template-driven and reactive forms in one single component, home page which adds the list of items. Use all the concepts from this document. Example: ToDo App shown in the session.

**Angular Project**

* **Creating new Project: ng new Assignmen1**

✔ Which stylesheet format would you like to use? **CSS**

✔ Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? **No**

✔ Do you want to create a 'zoneless' application without zone.js? **No**

✔ Which AI tools do you want to configure with Angular best practices? <https://angular.dev/ai/develop-with-ai>  **None**

* **To Run the Project: ng serve**
* **To Create Component (any part of the Frontend):**
  + Create Folder Name and Open in the terminal
  + Enter this Cmd **ng g c header**

CREATE src/app/components/header/header.spec.ts (551 bytes)

CREATE src/app/components/header/header.ts (196 bytes)

CREATE src/app/components/header/header.css (0 bytes)

CREATE src/app/components/header/header.html (22 bytes)

* + TypeScipt File Consist Code (Logic Part)
  + Html File Consist of Html Codes
  + Css file Consist of styling of the Component
  + These three files are connected to each other through the type script decorator
  + Decorators mentions:
    - **Selector** use to export and import component to another files
    - **Imports** to include another component inside a file insert the or import component in the TypeScript file
    - **TemplateUrl:** Specify the Location of the Html file which consist of the html code of the component
    - **StyleUrl:** Specifies the location of the Css file which is only applicable to that component.
* **Data Binding:** Data binding in Angular enables communication between the component's TypeScript logic (data and methods) and its HTML template (view). Angular supports two main types of data binding: **one-way binding (3)** and **two-way binding**.
  + **Interpolation –** Display the Data inside the variable in the html tags using {{}} brackets
  + **Property Binding –** Passing the value to the variable or the css class to the html tags dynamically using []
  + **Event Binding -** Binds user interactions (e.g., clicks, keypresses) to component methods using parentheses (()).
  + **Two-way Binding-** Two-way binding allows data to flow both ways: from the component to the template and from the template to the component. It is commonly used with form inputs using the [(ngModel)] directive, which requires the FormsModule.
* **Attaching function to button:**

To declare an function it should be handled in the following way

showAlertEvent() {

alert('Hey You have Click a button');

}

To assign an function to html it should be done in the following way

<button *class*="btn btn-outline-secondary me-2" *(click)*="*showAlertEvent*()">

            Show Toast

          </button>