* **What is Angular? What are Angular advantages? What the use of Angular?**

Angular is a component-based framework or JavaScript binding framework, which binds HTML UI with JavaScript model for building structured, scalable and single page application for client side with the help of routing.

Advantage:

Relatively simple to build SPA with help of component-based application.

To make flexible and structured applications.

Cross platform and open source.

Reusable code.

Testability with help of specs.ts

* **What is the difference between AngularJS and Angular?**

|  |  |
| --- | --- |
| **Angular JS** | **Angular** |
| 1. It only supports JavaScript. | It supports both JavaScript and TypeScript. |
| 2. This framework has a model-view-controller (MVC) architecture. | This framework has a component-based architecture. |
| 3. It does not have a CLI tool. | It has a CLI tool. |
| 4. It does not use Dependency Injection. | It uses Dependency Injection. |
| 5. It does not support mobile browsers. | It also supports mobile browsers. |
| 6. It is not so fast. | It is very fast. |

|  |  |  |
| --- | --- | --- |
| **Feature** | **AngularJS** | **Angular (2, 4, 5, 6, 7, 8, and 9 IVY)** |
| **Language** | JavaScript | TypeScript |
| **Architecture** | Controller | Component |
| **Mobile compliant** | No | Yes |
| **CLI** | No | Yes |
| **Lazy loading** | No | Yes |
| **SEO** | No | Yes |
| **Server side** | No | Yes |

* **Explain the importance of NPM and Node\_Modules folder?**

NPM is a package manager which makes installation of JavaScript library along with more free libraries.

Node\_modules folder where all the packages are installed.

* **What is CLI tool?**

Is a command line interface by which we can create Initial angular Project.

npm install @angilar/cli

This CLI will give whole component structure format

* **Explain the importance of Package.json file in Angular?**

It has all the reference of javascript references needed for the project.

* **Explain the importance of Component and Modules?**

Components are the most UI building block of a angular app.

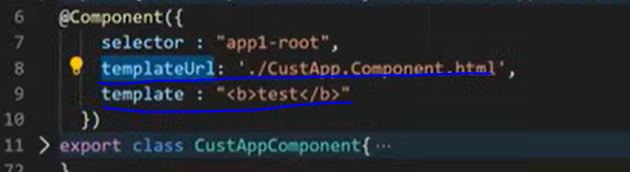
* **What is selector?**

A selector is used to identify each component uniquely into the component tree.

* **What is a template?**

Template is an HTML view of Angular in which we can write directives.

Two types (inline and separate physical HTML file.)



* **What is a module & what is app.module.ts?**

Module is a place where you can group the components, directives, pipes and services which are related to the application.

* **How an Angular App gets Loaded and Started? What are index.html, app-root, selector and main.ts?**

Angular is used to create Single Page Applications. index.html file is that single page. Index.html will invoke main.js file which is the JavaScript version of main.ts file.

main.ts file is like the entry point of the web app. It compiles the web app and bootstraps the AppModule to run in the browser.

App module file will then bootstrap the AppComponent.

AppComponent or app-root component is the HTML which you will see finally.

* **What is a Bootstrapped Module & Bootstrapped Component?**

When the Angular web application starts, the first module launched is the bootstrapped module, and the same is true for the bootstrapped component also.

Bootstrap component is an entry component that Angular loads into the DOM during the bootstrap process or application launch time.

* **What is Data Binding in Angular?**

Databinding is the communication between typescript view and the component of HTML view.

* **What is String Interpolation in Angular?**
* **What is Property Binding in Angular?**

**INPUT DATA**

**Interpolation** - one-way data-binding technique that is used to transfer the data from a **TypeScript code (component)** to an **HTML template (view).**

**It will work only with string type.**

**It is represented only inside bouble braces.** {{data}}

**Property Binding** - One-way binding from to transfer the data from a **TypeScript code (component)** to an **HTML template (view).**

**It** **is a** **superset of interpolation**.

**it can set an element property to a non-string data.** [property] =”data”

* **What is Event Binding in Angular?**

**OUTPUT DATA**

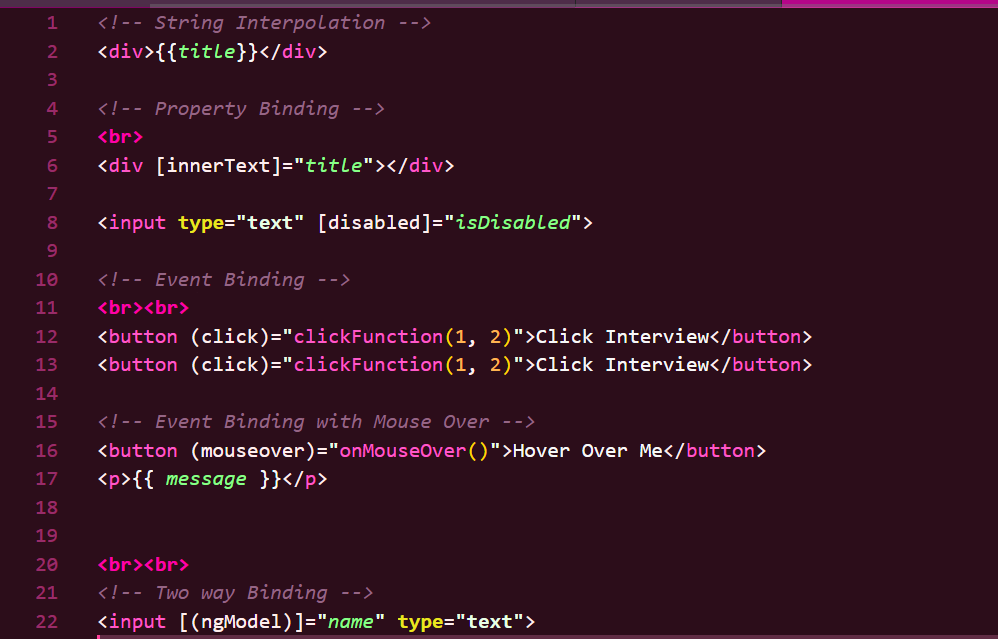
**Event Binding** - (event) =”expression”→ One-way binding from UI to the component

* **What is Two way Binding in Angular?**

**TWO WAY DATABINDING**

**Two-Way Binding** - Two-way data binding in Angular will help users to exchange data from the view to component and then from component to the view at the same time

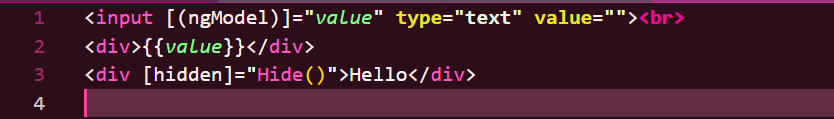
[(ngModel)] =”data”→



* **What are directives in Angular? and different types of Angular directives?**

Directives are classes that add additional behaviour to elements.

Angular syntax which can write inside HTML, so that we can change the behaviour of HTML DOM.



Angular has three types of directives:

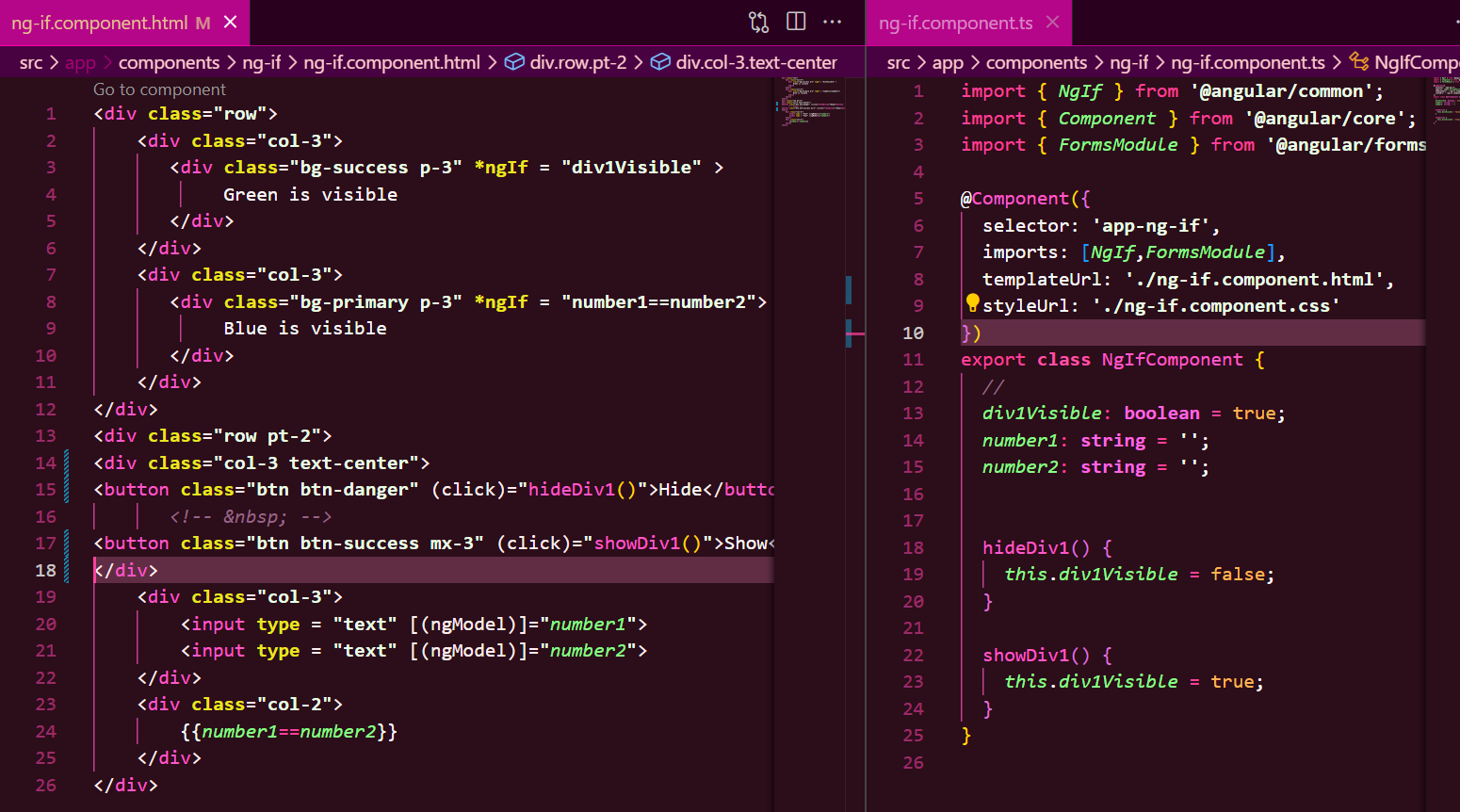
1. Component Directives (Most Common) → Customized user control. Directives with template (@Component)
2. Structural Directives → Change the DOM layout by adding or removing elements. (\*ngIf, \*ngFor, \*ngSwitch)
3. Attribute Directives → Modify the appearance or behaviour of an HTML element ([ngClass], [ngStyle], Custom directives)

* **What is \*ngIf Structural directive?**

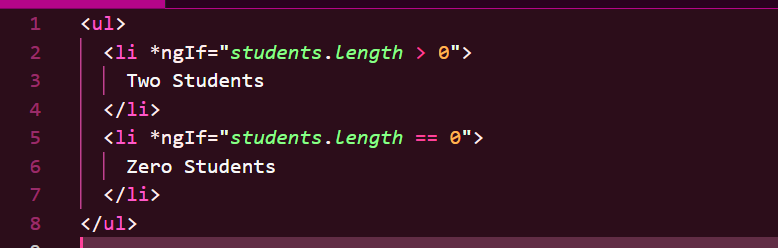
To display elements conditionally

example:

On click of hide button it should hide and vice versa.

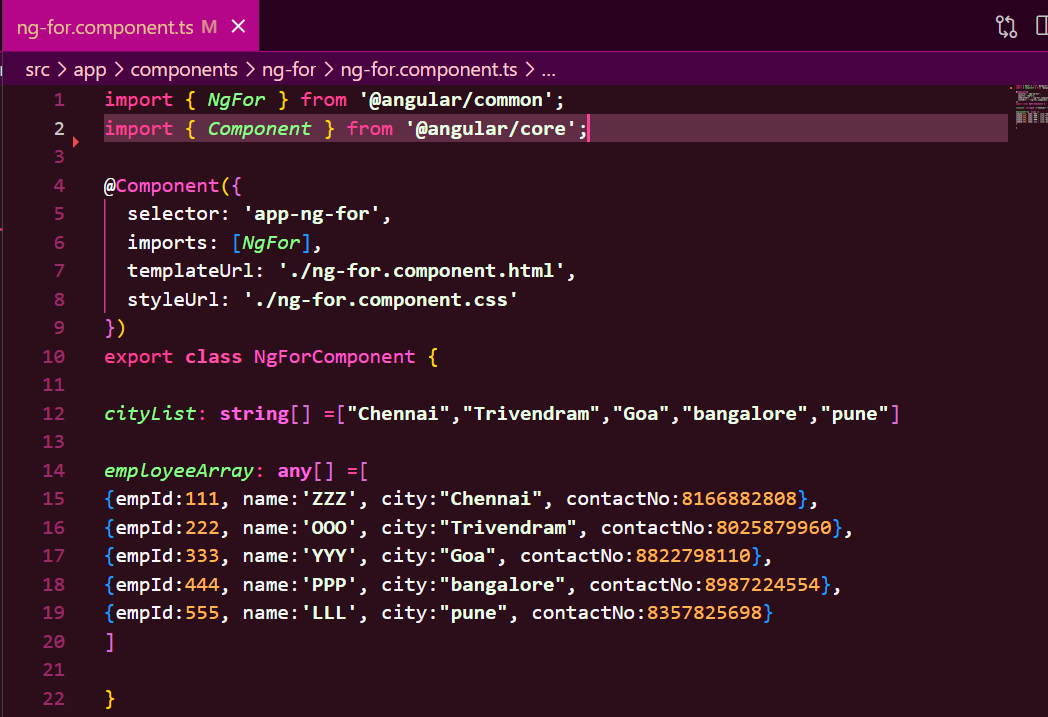


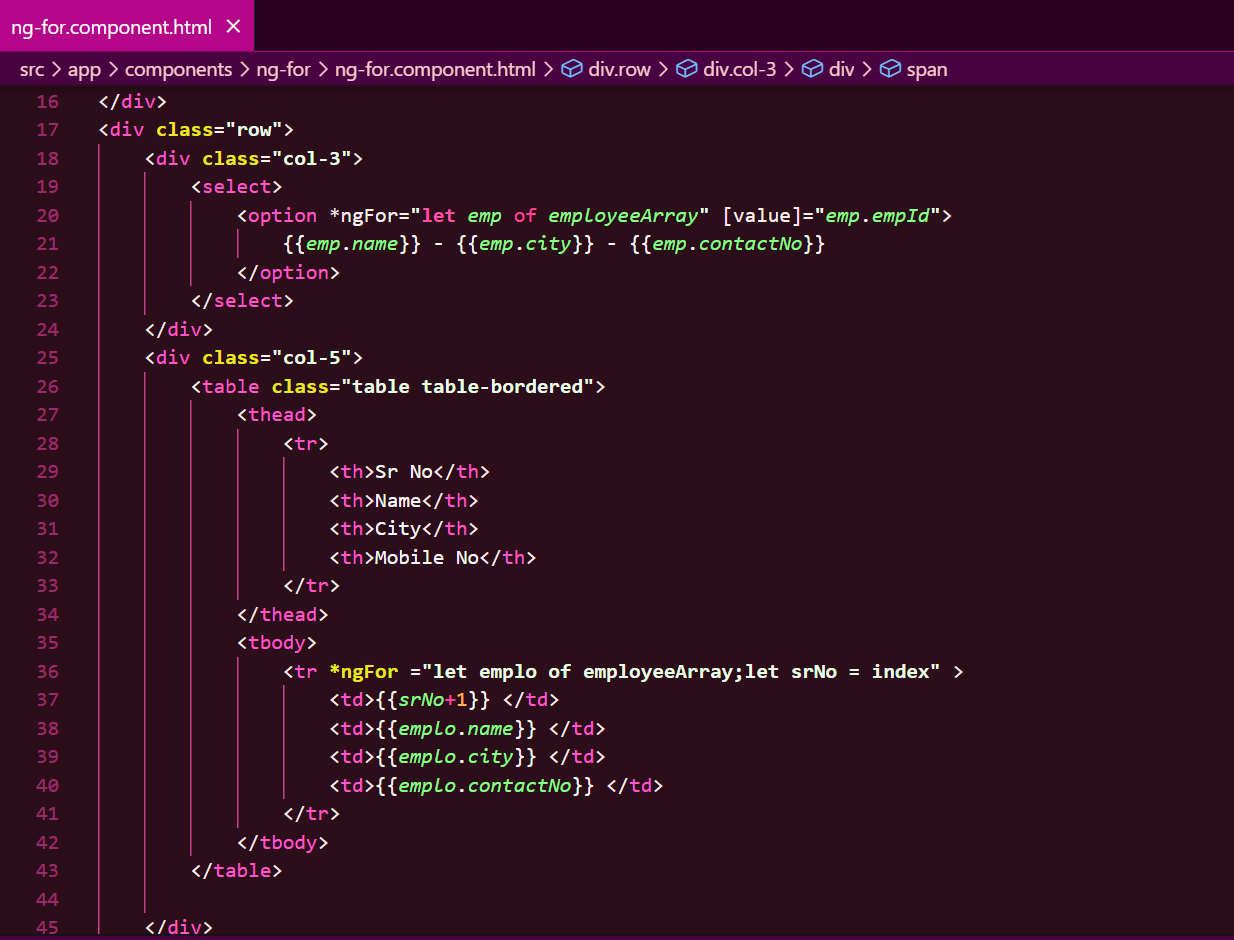
Another example



* **What is \*ngFor Structural directive?**

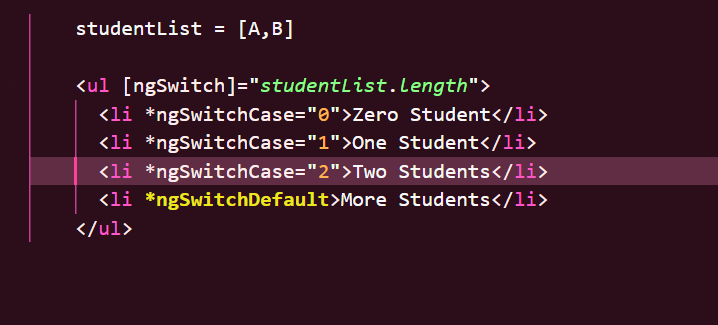
To iterate a list of items.





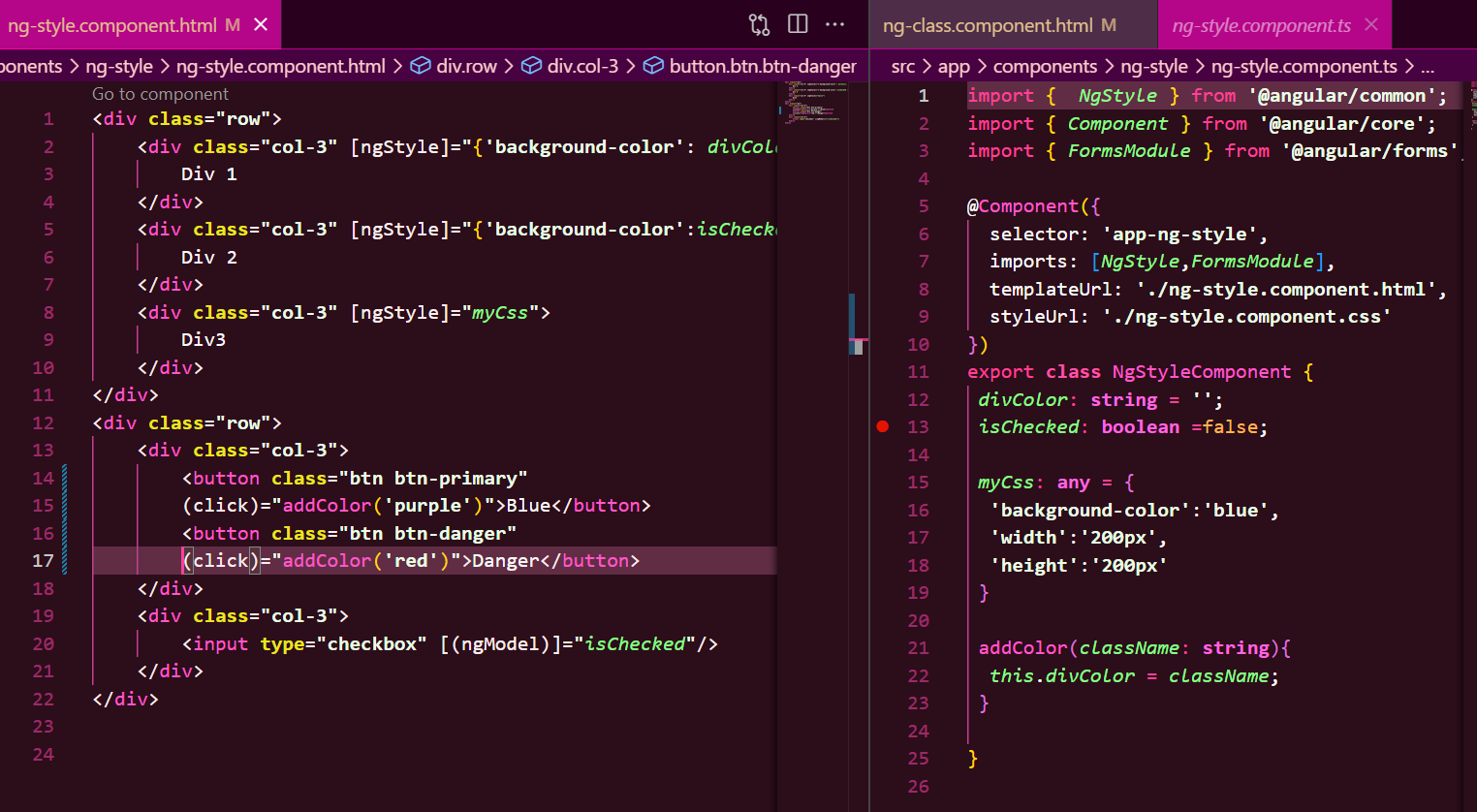
* **What is \*ngSwitch Structural directive?**

Used in combination with \*ngSwitchCase and \*ngSwitchDefault

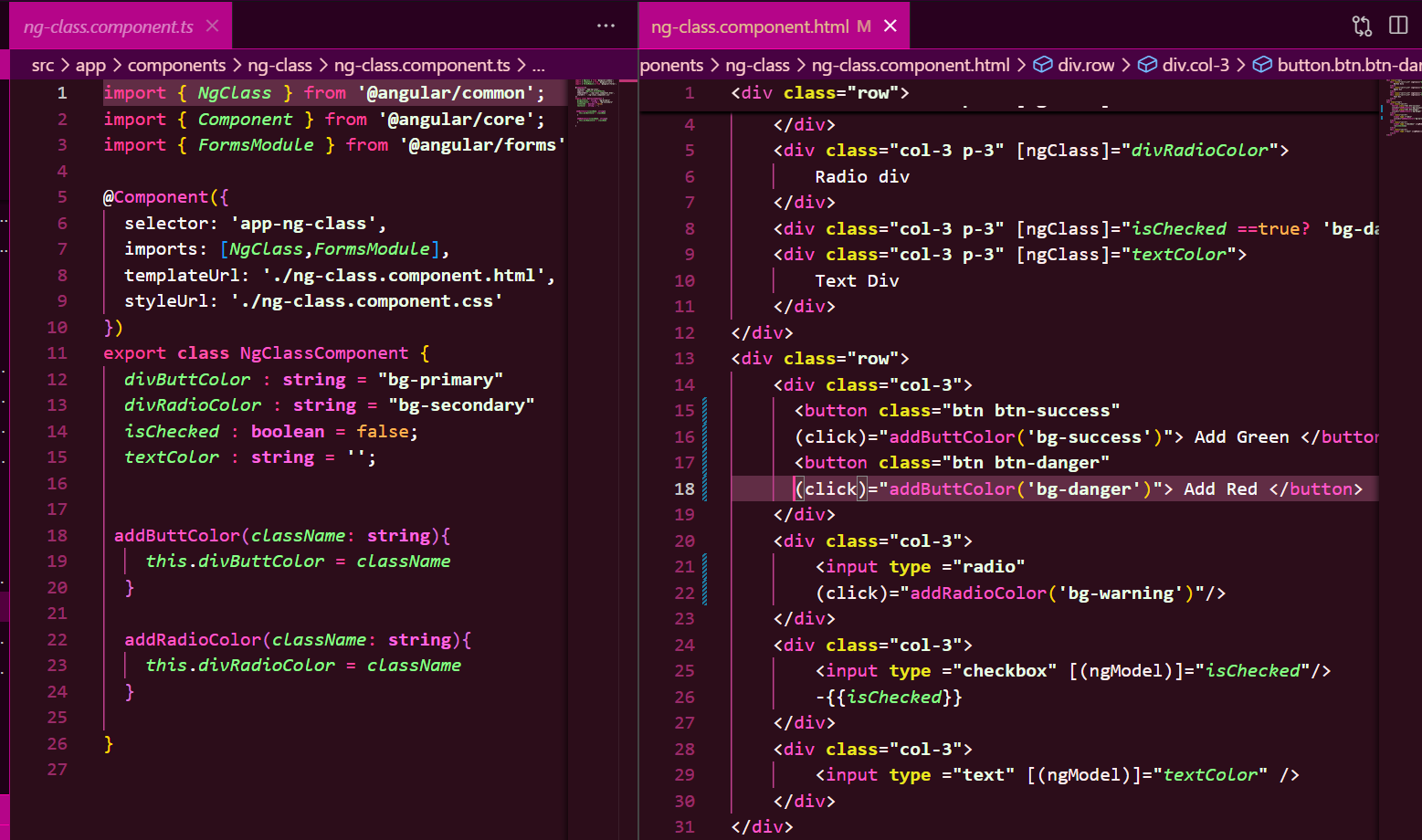


* **What is [ngStyle] Attribute directive?**

It updates the style for the HTML element.



* **What is [ngClass] Attribute directive**



* **What is the difference between Component, Attribute and Structural Directives?**

A Component is a special type of directive that includes a template.

**For example**, a UserProfileComponent might render a complete section of a page.

An **Attribute Directive** modifies the **appearance or behavior** of an existing element

**For instance,** ngClass and ngStyle are commonly used to dynamically apply styles or classes based on conditions.

A **Structural Directive**, on the other hand, **modifies the structure of the DOM** — it can add or remove elements from the view. Examples include \*ngIf and \*ngFor, which render elements conditionally or based on collections.

* **What is Decorator?**

A decorator is a special function

@Component() tells Angular that the class is a component and provides metadata like the selector, template, and styles.

* **What are the types of Decorator?**

@Component() – defines a class as a component.

@Directive() – marks a class as a directive.

@Pipe() – marks a class as a pipe.

@Injectable() – makes a class available for dependency injection.

@NgModule() – defines a module.

* **What is a decorator in Angular? Or What are Annotation or Metadata?**

Defines what king of angular class it is.

@component denotes Angular component.

@NgModule denotes Angular Module

* **What are Pipes? What are the types of Pipes & Parameterized Pipes?**

In Angular, **pipes** are used to **transform data** before displaying it in the view (template).  
They are **pure functions** that take input data and return a formatted output.

You use pipes with the pipe (|) symbol in templates.

* **What is Chaining Pipes?**

Chaining pipes in Angular means applying multiple pipes one after the other to a single value in a template.  
The output of one pipe becomes the input for the next pipe.

<p>{{ 'angular PIPE Example' | lowercase | titlecase }}</p>

* **Explain architecture of Angular?**

**⬇️ User Clicks a Button**

**┌──────────────────────────────────────────┐**

**│ COMPONENTS (UI + Logic) │**

**│ - app.component.ts (Controller) │**

**│ - app.component.html (View) │**

**└──────────────────────────────────────────┘**

**⬇️ Calls Service**

**┌──────────────────────────────────────────┐**

**│ SERVICES (Business Logic) │**

**│ - Fetch data from APIs │**

**│ - Perform calculations │**

**└──────────────────────────────────────────┘**

**⬇️ Uses DI to Fetch Data**

**┌──────────────────────────────────────────┐**

**│ DEPENDENCY INJECTION (DI) │**

**│ - Inject services into components │**

**└──────────────────────────────────────────┘**

**⬇️ Updates the UI**

**┌──────────────────────────────────────────┐**

**│ DIRECTIVES & PIPES │**

**│ - \*ngIf, \*ngFor (Structure) │**

**│ - ngClass, ngStyle (Styling) │**

**│ - Pipes (Data Transformation) │**

**└──────────────────────────────────────────┘**

* **What is SPA in Angular?**

SPA is single page Application where the main UI gets loaded ay once and the needed UI gets loaded on need.

* **How to implement SPA in Angular?**

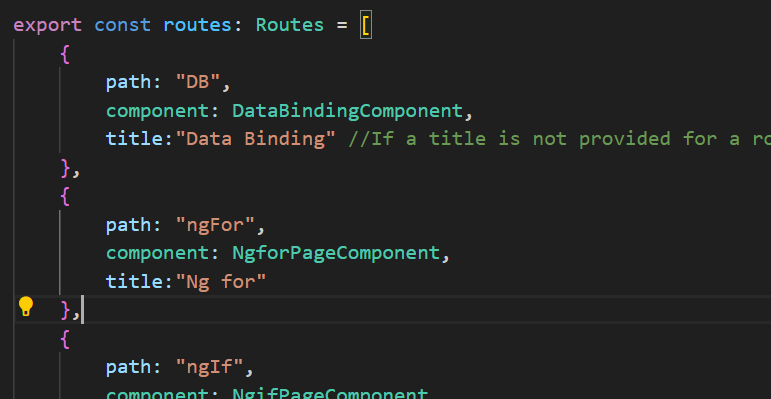
We have to use Angular routing, Routing is a simple collection in Angular.

* **How to implement routing in Angular?**

Routing is implemented using the @angular/router.

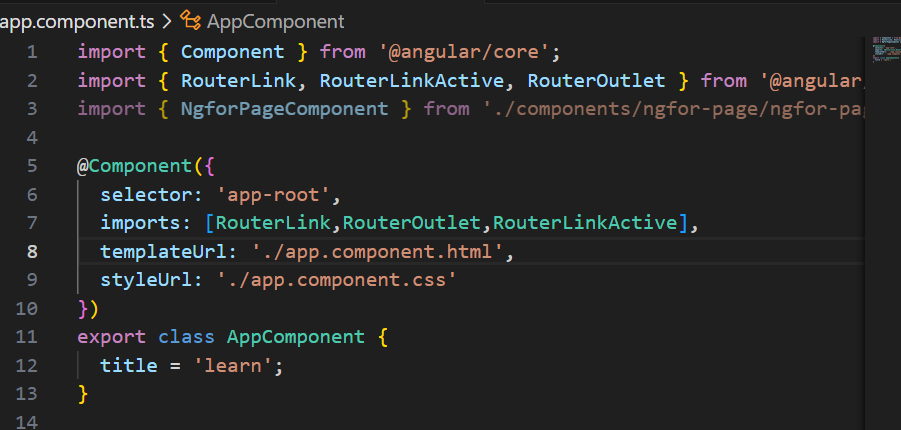
It enables navigation between different **components or views** without reloading the entire page so it supports SPA.

App.route.ts

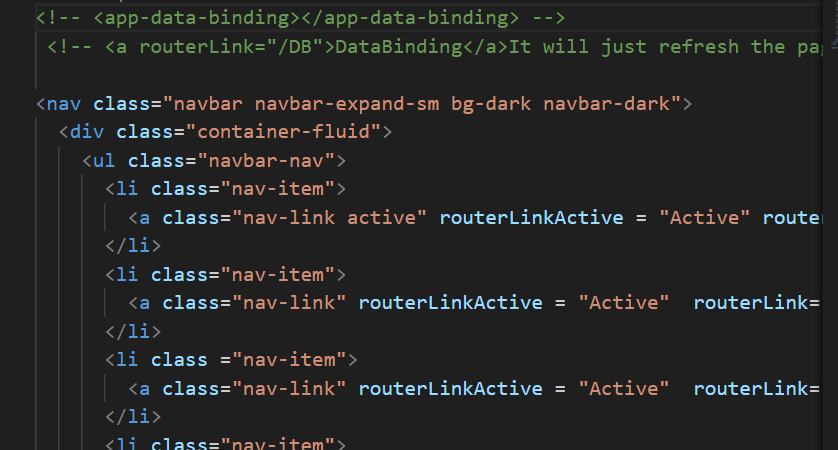


Add <router-outlet> in AppComponent

Appcomponent.ts



Setup the navigation link in app component.html



* **Explain Lazy Loading ?**

Instead of loading all modules at the application startup a technique used to load feature modules only when they are needed.

Imagine you have a dashboard only used by admins. There's no need to load admin features unless an admin logs in.  
With lazy loading, you only load that module **when the route /admin is visited**.

* **How to implement Lazy Loading in Angular?**

lazy loading is done using loadComponent for standalone components or loadChildren for feature modules/routes, improving performance by loading code only when needed.

* **Explain Services with Example?**

A service is a typescript class and a reusable code which can be used in multiple components.

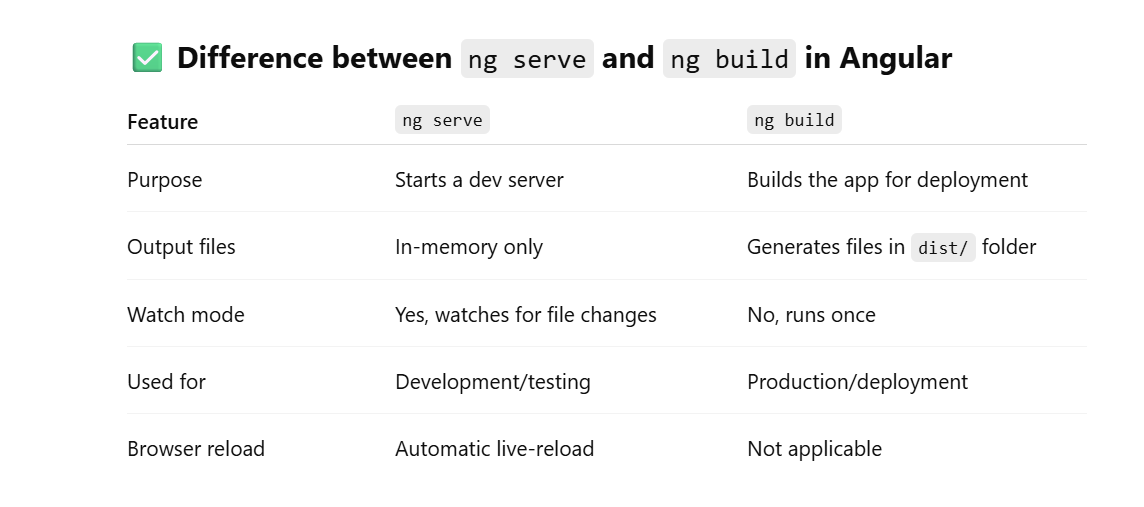
Service can be implemented with the help of dependency injection.

We can create a centralized or common data service.

* **How to create Servicein Angular?**

Using the command ng generate service logging.

* **Differentiate between ng Serve and ng build?**



* **Explain the --prod parameter in ng build ?**

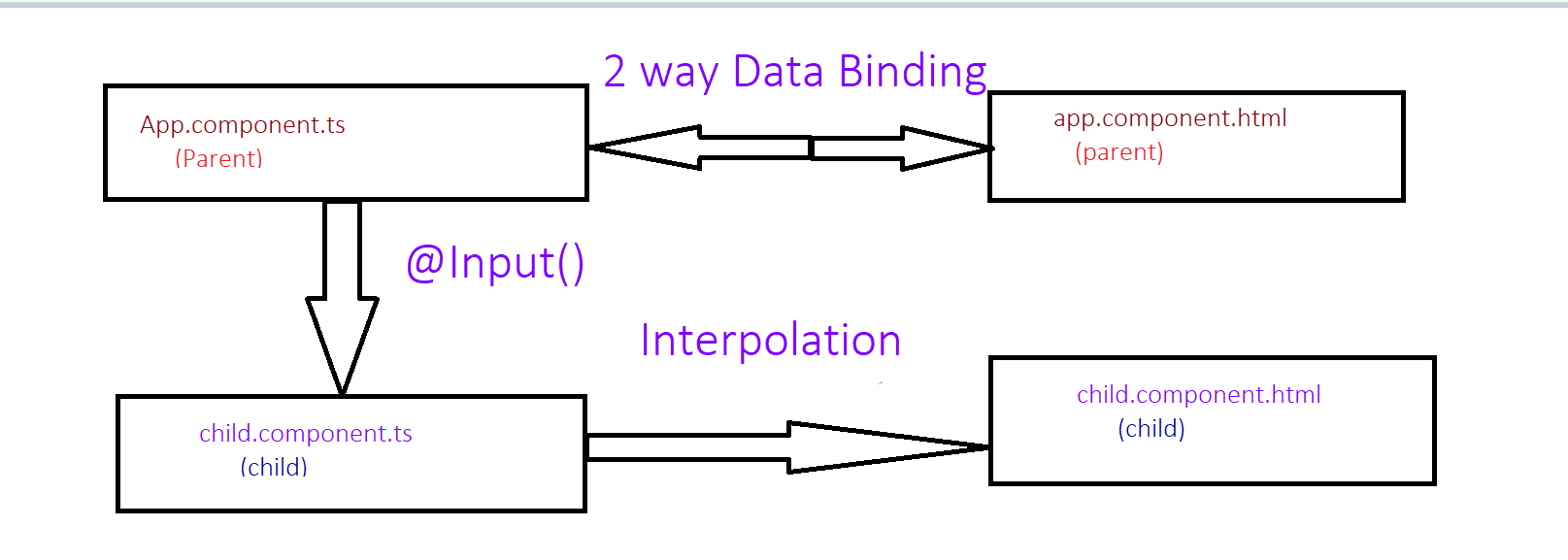
ng build --prod compiles the app with optimizations like AOT, minification, and tree-shaking to generate a smaller, faster production-ready bundle.

* **What is @Input decorator ? How to transfer data from parent to child component.**

@input decorator is used to transfer data from parent to child component.

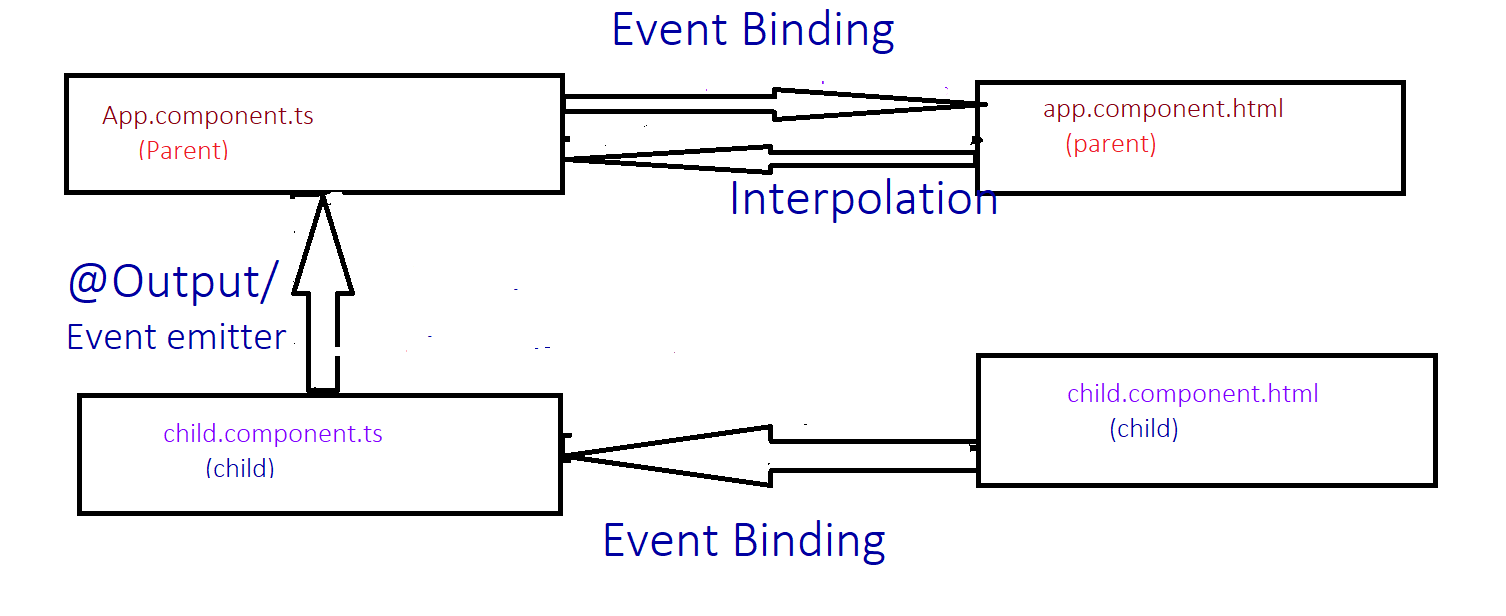
@input(): ability to receive data from outside component

1. **Two-Way Binding (Parent Component)**
   * Between app.component.ts and app.component.html
   * Uses [(ngModel)] for form elements
2. **Data Transfer from Parent to Child (@Input)**
   * Parent (app.component.ts) sends data to child using @Input()
   * Example: <app-child [data]="parentData"></app-child>
3. **Data Rendering in Child (Interpolation)**
   * Child (child.component.html) receives and displays data using {{ data }}



* **What is @Output decorator? How to transfer data from child to parent component.**

@Output decorator and event emitter together are used to pass data from child to parent component.



* **What is Depedency Injection?**

To achieve loose coupling, modularity, and testability by injecting required services or dependencies into a class instead of creating them manually.

* **How to implement Depedency Injection?**

To implement DI, Create a service ng generate service user

with @Injectable, provide it via providedIn: 'root', and inject it into components or services via the constructor.

* **What’s the benefit of Depedency Injection?**

Dependency Injection promotes modular, testable, and maintainable code by letting Angular manage service creation and lifecycle, rather than components doing it themselves.

* **How to use Dependency Injector with Services in Angular?**

To use Angular’s Dependency Injector with services, create the service with @Injectable, provide it (commonly with providedIn: 'root'), and inject it into components or other services using the constructor.

* **What is Hierarchical Dependency Injection?**

In angular if we inject any service in parent with help of provide it is available to all the child’s under it.

If we inject to any of the child component, that will be accessible to the belonging child but not to the siblings or parent.



* **What is Provider in Angular?**

A provider in Angular defines how a dependency (like a service) is created and injected by the DI system.

* **What is the role of @Injectable Decorator in a Service?**

The @Injectable() decorator tells Angular that a class can participate in Dependency Injection (DI).

It allows Angular to:

* Inject dependencies into the service itself, if needed.
* Register the service as a provider, when providedIn is used.
* **What are Parent-Child Components?**

Parent-child components are components where:

**The parent component** contains or hosts another component.

**The child component** is nested inside the parent and can communicate with it.

* **What are Lifecycle Hooks in Angular?**

A component from creation to destruction goes through several stages and these stages are the life cycle hooks.

The stages will cover activities like:

Component Instantiating.

Rendering the component HTML view.

Creating the child components if required.

Destroying the components.

|  |
| --- |
|  |

* **What is ngOnInit life cycle hook in Angular?**
* **What is the difference between constructor and ngOnInit?**

|  |  |
| --- | --- |
| **ngOnInit** | **Constructor** |
| 1. NgOnInit is an Angular lifecycle hook, which signals the activation of the created component. | The constructor is a method in a TypeScript class, that automatically gets called when the class is being instantiated. |
| 2. ngOnInit is called after ngOnChanges lifecycle-hook. | Constructor is called before any lifecycle-hook. |
| 3. When ngOnInit is called, everything about the component is already ready, so it’s used to perform most of the business logic on the component. | When the constructor is called, everything in the component is not ready, so it's mostly used for injecting dependencies only. |

* **What are Asynchronous operations?**

Asynchronous operations allow a program to perform tasks in the background (like API calls, timers, or file loading) without blocking the main execution thread.

* **What is the difference between Promise and Observable?**

Both are used to transfer the data in javascript asynchronously.

| **Observables** | **Promises** |
| --- | --- |

|  |  |
| --- | --- |
| **1.** Emit **multiple** values over a period of time. Also called **streaming of data**. | Emit a **single** value at a time. |

|  |  |
| --- | --- |
| **2.** Are **lazy**: they’re not executed until we **subscribe** to them using the subscribe() method. | Are **not lazy**: execute **immediately** after creation. |

|  |  |
| --- | --- |
| **3.** Have subscriptions that are **cancellable** using the unsubscribe() method. | Are **not cancellable**. |

* **What are Angular forms and its types?**

Angular forms used to handle users input.

Types of forms:

1.Template Form: most of the code will be written in .html file

2.Reactive Forms: most of the code will be written in component.ts file.

| **Template Driven Forms** | **Reactive Forms** |
| --- | --- |

|  |  |
| --- | --- |
| **1.** Most code and validation logic is written in **HTML template**. | Most code and validation logic is written in **component TypeScript class file**. |

|  |  |
| --- | --- |
| **2.** Requires adding **FormsModule** in AppModule to activate it. | Requires adding **ReactiveFormsModule** in AppModule to activate it. |

|  |  |
| --- | --- |
| **3.** Used when the application is **simple** and has **fewer controls**. | Used when the application is **complex** and has **more controls**. |

* **What is RxJS?**

**RxJS (Reactive Extensions for JavaScript)** is a **library for reactive programming** using **Observables**, allowing you to **work with asynchronous data streams** (like events, HTTP responses, or user input).

It’s the core of Angular’s reactive features—especially HttpClient, Forms, and routing.

* **What is Observable? How to implement Observable**

An **Observable** is a core concept from RxJS that represents a **stream of asynchronous data** that can emit multiple values over time (like events, HTTP data, etc.).

Angular uses Observables for:

HTTP calls (HttpClient)

Forms and value changes

Event handling

Reactive programming

* **What is the role of HttpClient in Angular?**

HttpClient is Angular’s core service for making HTTP requests and handling responses asynchronously using Observables.

* **What is routing and how to set it in Angular?**

Routing helps in navigating from one view to another view with the help of URL.

How to setup Routing in Angular?

1. Create Angular project with Routing option
2. Create components for routing
3. Configure Routes
4. Configure links

In Angular 19:

In app.routes.ts need to add the pat and its component name

{

        path:**'admin'**,

        component:*AdminComponent*

},

In app.components.html needs to add the navigation link.

<li **class**=**"nav-item"** >

**<a class="nav-link active" routerLink="admin">**Admin**</a>**

 </li>

Router link is used to navigate to different route.

* **What is the role of HTTP client in Angular?**

**HttpClient** is a **built-in service** class available in Angular.  
 @angular/common/http package.  
Performs **HTTP requests**.s

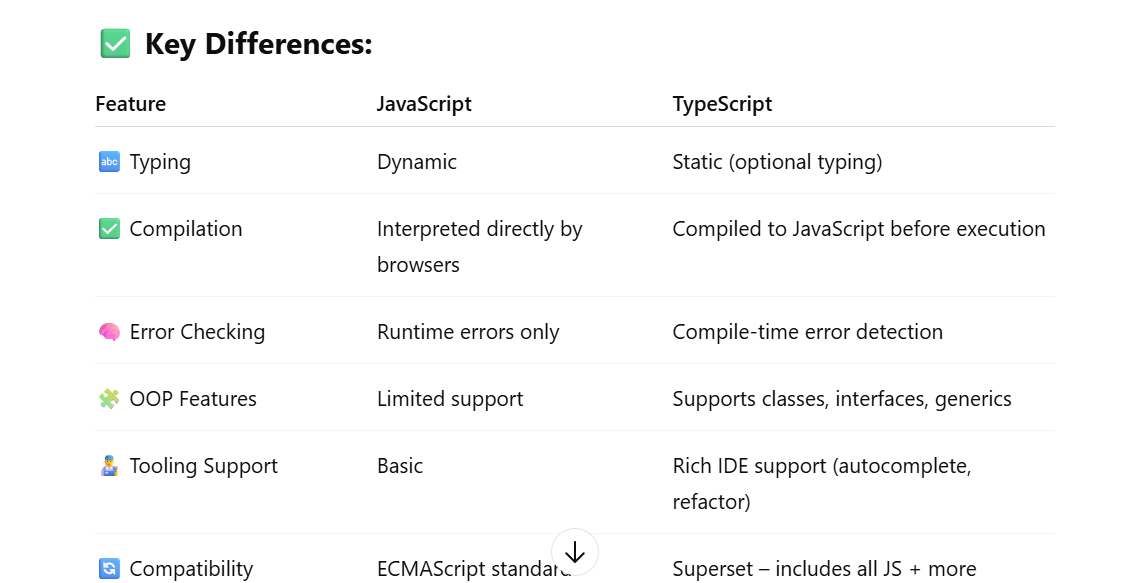
* **What is the HTTP Interceptor? When and where to use Angular?**

HTTP Interceptor are special services only which are used to intercept HTTP requests’

* **What is typescript and why do we need it?**

It adds types to JavaScript or superset of JavaScript.

* **What is Typescript? Or What is the difference between Typescript and Javascript?**



* **What is the difference between let and var keyword?**

var is function-scoped and hoisted, while let is block-scoped and safer to use in modern JavaScript.

* Hoisting
* Scope/Memory
* Redeclaration/ Updating

Hoisting means it is a mechanism where it moves the variable/function declaration to the top of its nearest scope

Example

|  |  |  |
| --- | --- | --- |
| **Var** | **Let** | **Const** |
| console.log(a); | HOISTING will not support | |
| var a = 10; |
| Internally var a; console.log(a);//nearest scope a = 10 ; |
| SCOPE/Memory var is a function level scope | let/const are block level scope { } | |
| var supports redeclaration and updation | let supports only updation | with const no redeclaration / updation |

* **What is Type annotation?**

Type annotation is the practice of explicitly specifying types in TypeScript to ensure safer, more maintainable code.

* **What are Built in/ Primitive and User-Defined/ Non-primitive Types in Typescript?**

Primitive types are built-in and hold single values, while user-defined types are custom structures used to model complex data in TypeScript.

* **What is “any” type in Typescript?**

The any type disables type checking for a variable, offering flexibility but reducing type safety.

* **What is Enum type in Typescript?**

enum in TypeScript defines a set of named constants (numeric or string), improving code readability and type safety.

* **What is Type Assertion in Typescript?**

Type assertion tells TypeScript what type a variable is, helping the compiler with better type checking without affecting runtime.

* **What are Arrow Functions in Typescript?**

Arrow functions are a shorthand way to write functions and preserve the this context, making them ideal for callbacks in TypeScript.