**Angular**

Angular is a web framework that empowers developers to build fast, reliable applications.

Web Framework to Develop Single Page Application (SPA)

Built by GoogleYear October 2010

Angular provides a broad suite of tools, APIs, and libraries to simplify and streamline your development workflow.

Angular gives you a solid platform on which to build fast, reliable applications.

that scale with both the size of your team and the size of your codebase.

Why I should Go for Angular?

* It is a One Stop where you Can
* Build
* Develop
* Run
* Test
* Deploy Application
* Angular (also referred to as "Angular 2+") is a TypeScript-based free and open-source single-page web application framework.
* It is developed by Google and by a community of individuals and corporations.
* Angular is a complete rewrite from the same team that built AngularJS.
* The Angular ecosystem consists of a diverse group of over 1.7 million developers, library authors, and content creators. According to the Stack Overflow Developer Survey, Angular is one of the most commonly used web frameworks

**Features**

Component-based architecture

* Angular uses a component-based architecture, which allows developers to build encapsulated, reusable user interface elements. Each component encapsulates its own HTML, CSS, and TypeScript, making it easier to manage and test individual pieces of an application.[9]

Data binding

* Angular supports two-way data binding, which synchronizes data between the model and the view. This ensures that any changes in the view are automatically reflected in the model and vice versa.

Routing

* Angular includes a router that allows developers to define and manage application states and navigation paths, making it easier to build single-page applications with complex routing.[13]

Angular CLI

* The Angular CLI (Command Line Interface) provides a set of tools for creating, building, testing, and deploying Angular applications. It enables rapid application setup and simplifies ongoing development tasks.

To be Noted.....

* Reactive framework
* Component based architecture
* Module based design
* Efficient and performance template engine
* Lot of Inbuilt directives to enhance the template engine
* Easy to create custom directives to extend the features of template engine
* Pipes to add helper functions into the template engine
* Automatic data binding
* Efficient and performance routing engine
* Easy to use dependency injection framework
* Support modern HTTP and CSS functionality
* Support CSS preprocessor
* Inbuilt support for HTTP client
* Supports Single page application development
* Supports progressive web app development (PWA app)
* Supports accessibility
* Supports internationalization
* Supports Server Side Rendering (SSR) through Angular Universal
* Inbuilt End-to-End testing support
* Support code generation through Angular CLI tools
* Efficient and performant code building through in-house build tools

Advantages Angular:

**Advantages of the Angular Framework**

**TypeScript Language**

TypeScript offers type safety, which helps in identifying potential issues early in the development process. This reduces bugs and ensures the creation of high-quality applications by providing a more structured coding environment.

**Full-Stack Framework**

Angular is a comprehensive full-stack framework that supports both client-side and server-side rendering. It also enables the development of Progressive Web Apps (PWAs). With built-in tools like a testing framework, dependency injection, reactive programming, a template engine, and data binding, Angular optimizes performance by efficiently bundling code for faster browser rendering.

**Easy to Learn**

Angular's all-inclusive functionality means developers don't need to search for, analyze, or learn multiple libraries. Additionally, Angular offers extensive documentation, complete with code examples for every version of the framework, making it easier for developers to get started.

**Quick to Get Started**

With Angular's Command Line Interface (CLI), developers can quickly set up a new project with default settings. The CLI also supports routing and configuration for CSS preprocessors, making the setup process streamlined and efficient.

**Simplified Development Process**

Angular's CLI tools help developers create new components and directives with ease. The framework's component-based architecture, modular design, reactive programming, data binding, and routing system all contribute to faster application development.

**Simplified Deployment**

Angular offers integrated tools to compile and deploy applications. The framework's build tools are regularly updated and improved by the Angular team, enhancing the overall developer experience when it comes to deployment.

**Efficient Testing**

Angular comes with an integrated end-to-end testing framework that covers all aspects of the framework's features. Developers can write specifications for each component and conduct thorough testing to ensure reliability and quality.

**Continuous Improvement**

The Angular team consistently updates the framework every six months, introducing new features, improving existing ones, and ensuring the framework stays aligned with the latest advancements in web development.

**Disadvantages of the Angular Framework**

**Limited Support for JavaScript**

While TypeScript is a superior language, the lack of robust support for JavaScript means developers must first learn TypeScript before diving into Angular development. This requirement can delay the adoption of Angular for those already familiar with JavaScript, making the learning curve steeper.

**Steep Learning Curve**

Angular's extensive set of built-in features and concepts can be overwhelming, requiring significant time and effort to fully understand and master all aspects of the framework.

**Difficulty in Specialization**

Mastering Angular to the level required to develop high-performance applications demands a substantial investment of time and energy. This makes it challenging for developers to specialize and become experts in Angular quickly.

**Limited Flexibility**

As a full-stack framework, Angular offers fewer options when it comes to selecting external libraries for specific tasks. Developers must rely on the features provided by the Angular team and may have to wait for the framework to introduce the functionality they need.

**Fewer Developer Tools**

Compared to other frameworks like React, Angular has a smaller selection of developer tools for tasks like debugging, profiling, and testing. This can make certain aspects of development more difficult or less efficient.

**Version In Angular:**

* Angular 2 14 September 2016 Initial release
* Angular 4 23 March 2017Added ngIf and ngFor. Backward compatible with Angular 2.
* Angular 4.3 18 July 2017HttpClient for making HTTP requests, conditionally disabling animations, new router life cycle events for Guards and Resolvers.
* Minor release, meaning that it contains no breaking changes and that it is a drop-in replacement for Angular 4.x.x.
* Angular 5 1 November 2017 Support for progressive web apps, a build optimizer and improvements related to Material Design.
* Angular 6 4 May 2018 Experimental custom element support, added ng update command
* Angular 7 18 October 2018 Updates regarding Application Performance, Angular Material & CDK, Virtual Scrolling, Improved Accessibility of Selects. Support for Content Projection using web standard for custom elements, and dependency updates regarding Typescript 3.1, RxJS 6.3 and Node.js 10.
* Angular 9 6 February 2020 Improved build times, enabling AOT on by default
* Angular 10 24 June 2020 New Date Range Picker (Material UI library).
* Angular 11 11 November 2020 Experimental Webpack 5 support
* Angular 12 12 May 2021 Deprecated support for Internet Explorer 11.
* Angular 13 4 November 2021 Removed deprecated View Engine renderer
* Angular 14 2 June 2022 Typed forms, standalone components, and new primitives in the Angular CDK (component dev kit).
* Angular 15 November 18, 2022 Standalone APIs, directive composition API.
* Angular 163 May 2023 Partial hydration for Angular Universal server-side rendering, experimental Jest support, and esbuild-based build system for development servers.
* Angular 17 November 8, 2023 Application builder, a new syntax for control flow, and a re-worked learning and documentation website.
* Angular 18 May 22, 2024 Experimental zoneless change detection support and server-side rendering improvements.
* Angular 19 November 19, 2024 Angular directives, components and pipes are now standalone by default.

**Installation**

* **Node.js - v**[**^18.19.1 or newer**](https://angular.dev/reference/versions)
* **Text editor - We recommend**[**Visual Studio Code**](https://code.visualstudio.com/)
* **Terminal - Required for running Angular CLI commands**

**Instructions**

[**Install Angular CLI**](https://angular.dev/installation#install-angular-cli)

**Open a terminal (if you're using**[**Visual Studio Code**](https://code.visualstudio.com/)**, you can open an**[**integrated terminal**](https://code.visualstudio.com/docs/editor/integrated-terminal)**) and run the following command:**

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

Create a new project:

In your terminal, run the CLI command **ng new** with the desired project name. In the following examples, we'll be using the example project name of my-first-angular-app.

ng new <project-name>

✔ Packages installed successfully.

Successfully initialized git.

[Running your new project locally](https://angular.dev/installation#running-your-new-project-locally)

In your terminal, switch to your new Angular project.

cd my-first-angular-app

All of your dependencies should be installed at this point (which you can verify by checking for the existent for a node\_modules folder in your project), so you can start your project by running the command:

npm start

If everything is successful, you should see a similar confirmation message in your terminal:

Watch mode enabled. Watching for file changes...

NOTE: Raw file sizes do not reflect development server per-request transformations.

➜ Local: http://localhost:4200/

➜ press h + enter to show help

**Components**

The fundamental building block for creating applications in Angular.

Components are the main building blocks of Angular applications.

Each component represents a part of a larger web page.

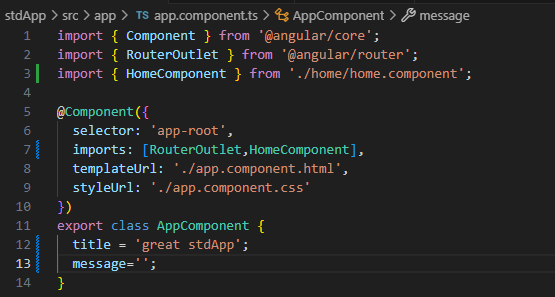
Organizing an application into components helps provide structure to your project, clearly separating code into specific parts that are easy to maintain and grow over time.

[Defining a component](https://angular.dev/essentials/components#defining-a-component)

Every component has a few main parts:

1. A @Component[decorator](https://www.typescriptlang.org/docs/handbook/decorators.html) that contains some configuration used by Angular.
2. An HTML template that controls what renders into the DOM.
3. A [CSS selector](https://developer.mozilla.org/docs/Learn/CSS/Building_blocks/Selectors) that defines how the component is used in HTML.
4. A TypeScript class with behaviors, such as handling user input or making requests to a server.

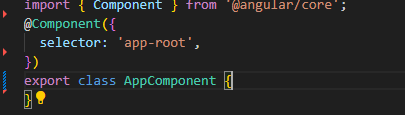
Here is a simplified example.



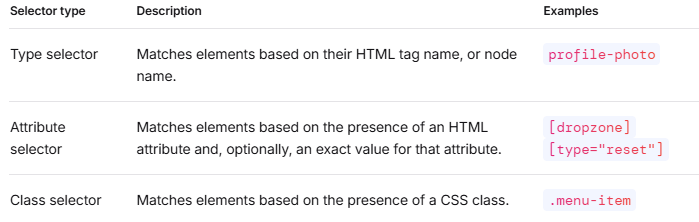
By default, Angular components are ***standalone***, meaning that you can directly add them to the imports array of other components. Components created with an earlier version of Angular may instead specify standalone: false in their @Component decorator. For these components, you instead import the NgModule in which the component is defined

**Selectors in Angular:**

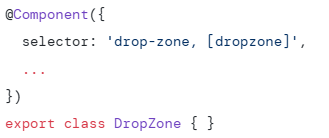
Every component defines a CSS selector that determines how thecomponent is used:



Types of Selectors:



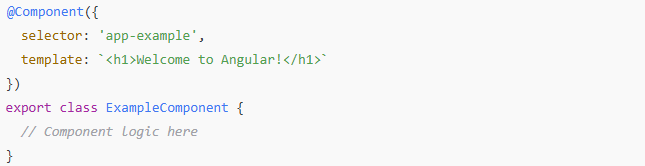
Combining Selector:



**Template vs Template URL**

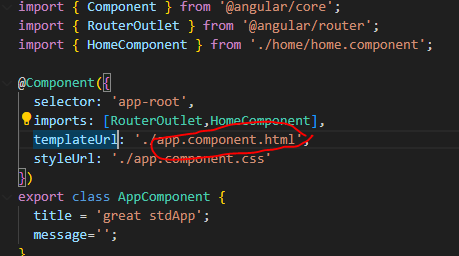
template

* The template property is used to directly define the HTML markup inside the component’s decorator, as a string.
* This is useful for small components or when you want to keep the template code within the same file for simplicity.
* The HTML content is written directly in the component's TypeScript file.



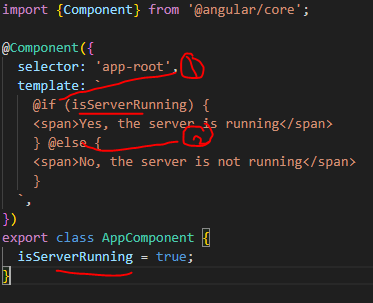
templateUrl

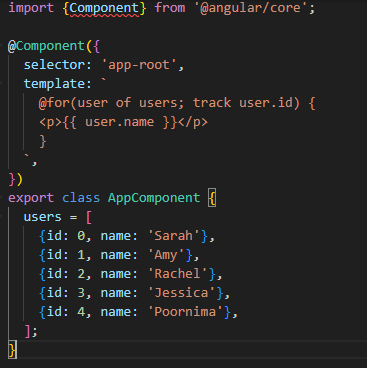
* The templateUrl property refers to an external HTML file that contains the template. This is useful for larger components, where separating the HTML from the TypeScript logic improves readability and maintainability.
* It points to the file path of the HTML file relative to the component.



**Control Flow in Components - @if**

Deciding what to display on the screen for a user is a common task in application development. Many times, the decision is made programmatically using conditions.





**Binding**

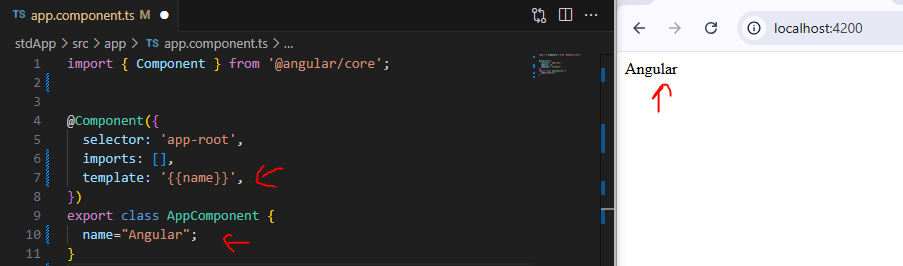
Binding is the process of linking the data from the component (model) to the view (template). When the data in the component changes, the view updates automatically to reflect those changes.

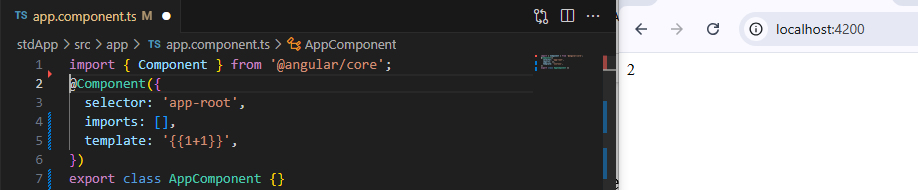
Type of Binding

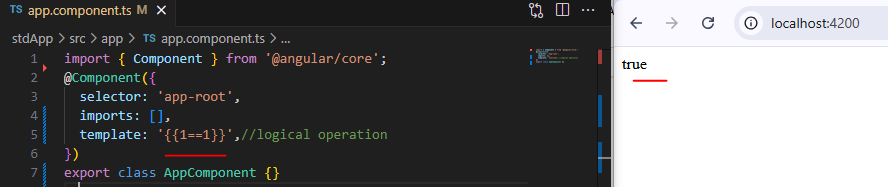
* Text interpolation
* Attribute binding
* Class binding
* Style binding
* Property binding
* Event binding
* Two way data binding

**Text interpolation**: It isthe simplest form of binding we use brackets

{{}} -Represent Interpolation bracket which evaluate text, logical, numerical operation

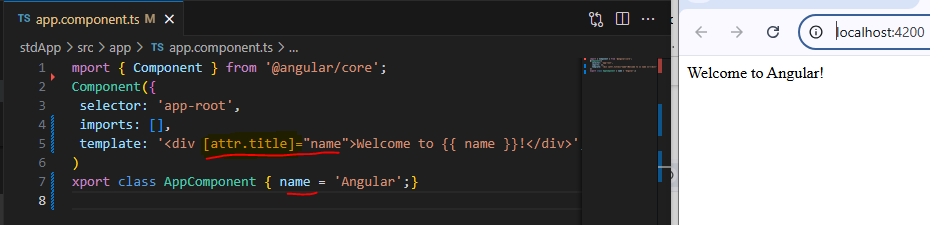






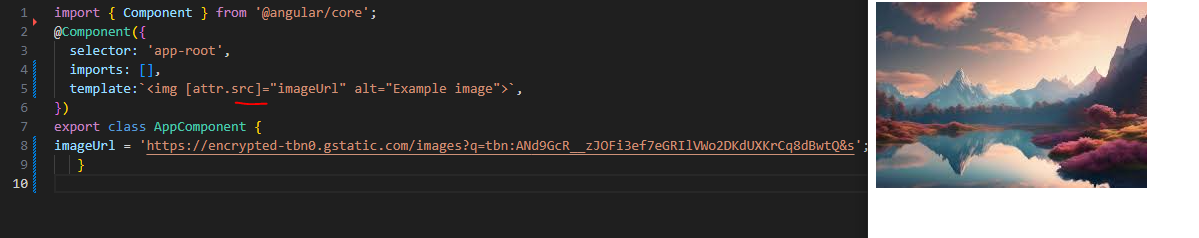
**Attribute binding**: is used to connect a property from the component (like name) to an HTML element's attribute in the template.

For example, if you have a name property in your component, you can bind it to an attribute of an HTML element, such as setting the title attribute.

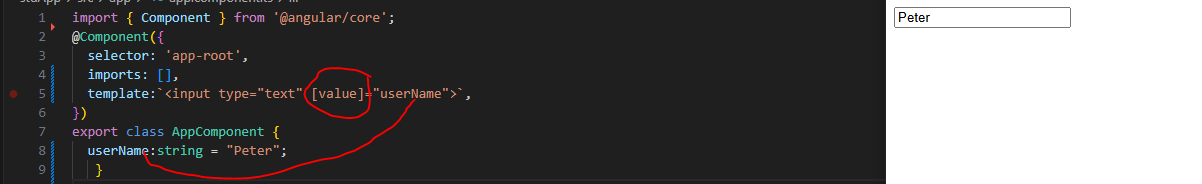


It helps to set the value for the attribute of the HTML element.

Angular exposes attributes of the HTML element with a matching property with the attribute name converted into camelCase



**Property binding:** is used to bind the data from property of a component to DOM elements. It is denoted by **[]**.



If you want to dynamically bind any property add square brackets to that property like [value] we did value=>[Value]

**Class binding** is used to bind the data from component to HTML class property

<p [class]=”first”>Hi am Red</p>

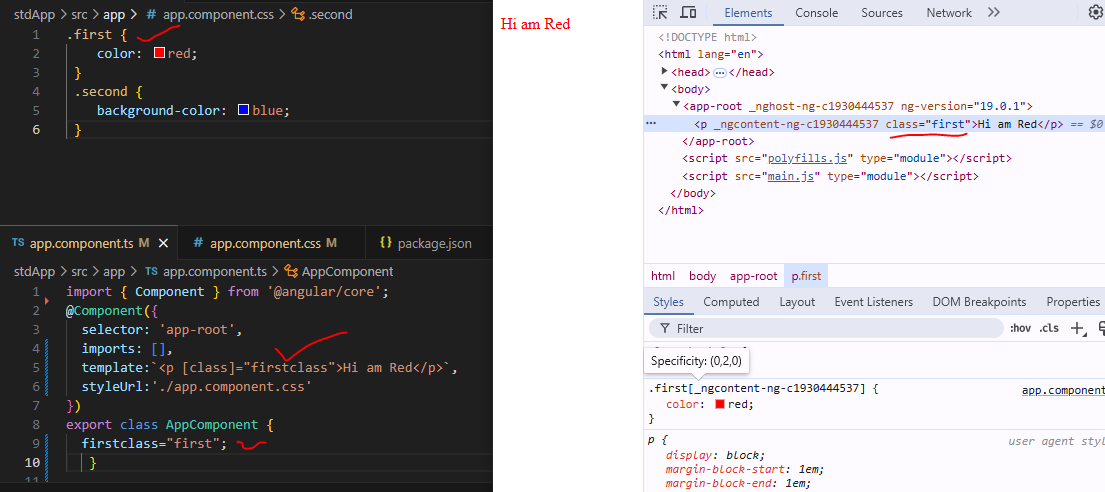
Classes can be added:

1)string

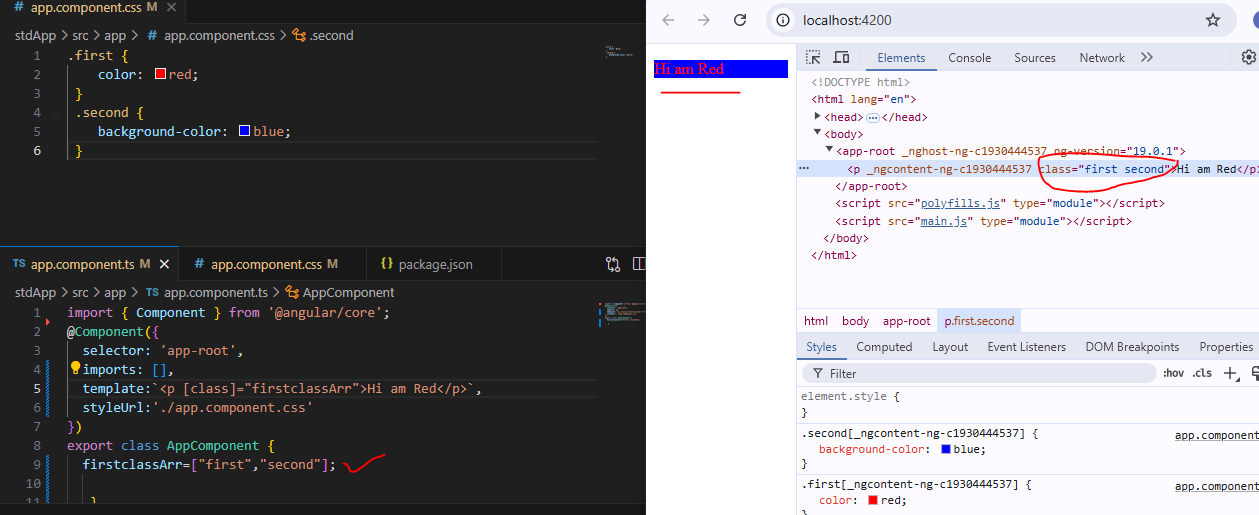
2)array

3)object

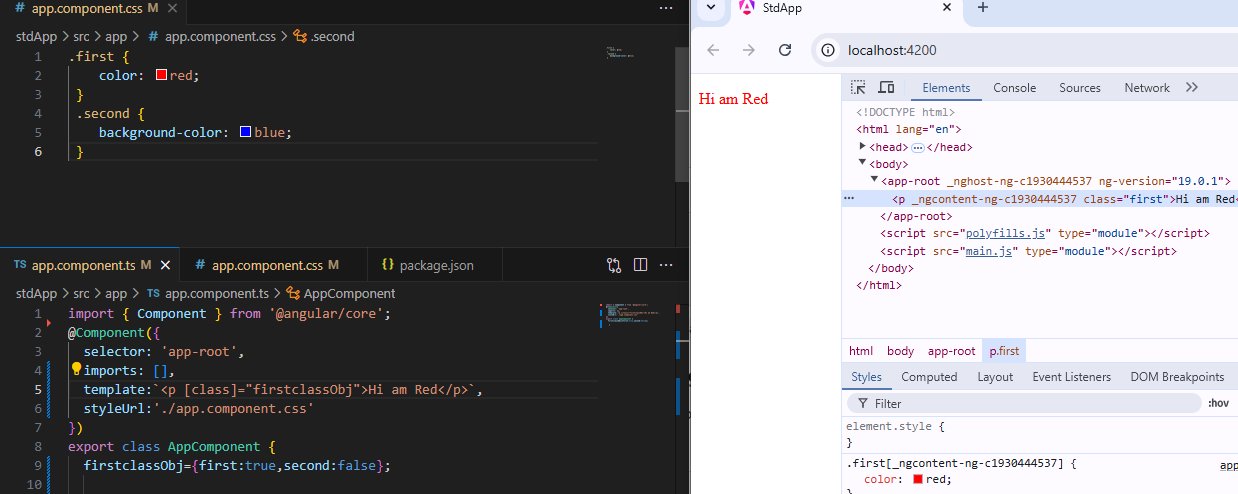
String Example



Array Example:



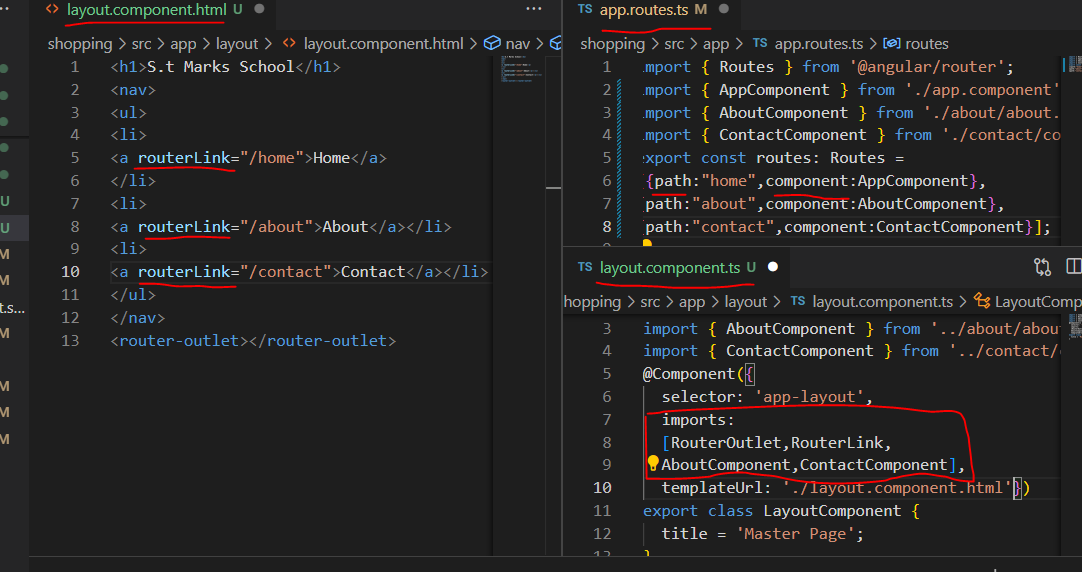
Object Example:



Routing in Angular :

If you want to Navigate form One Component to Another Component you can use Routing

**Step1:**Create 4 Components -Home,About,Contact,Layout

**Step2:**

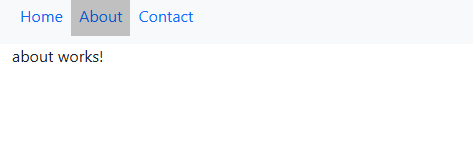
Here we need to import in Layout Component

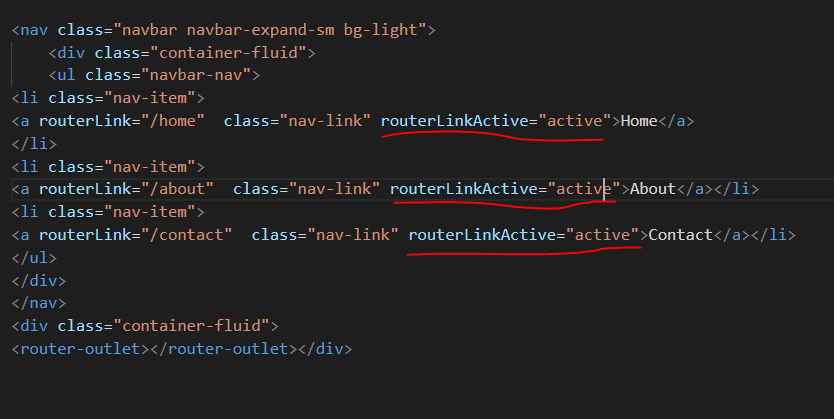
**RouterOutlet**: is used display Our called component in master layout

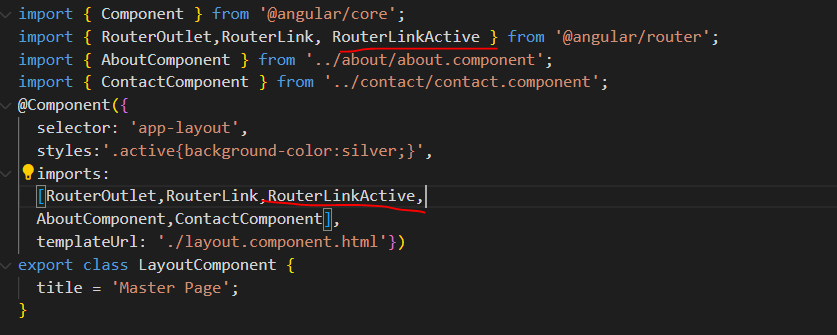
**RouterLink:** is used with anchor tag to navigate to pages without reloading page

**All Configuration regarding routes written in app.routes.ts**

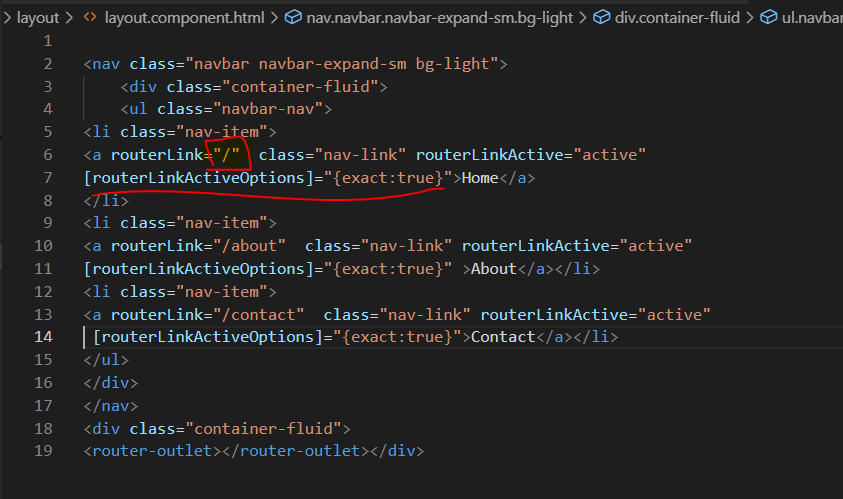
**How to make Menu Highlight which page we click**





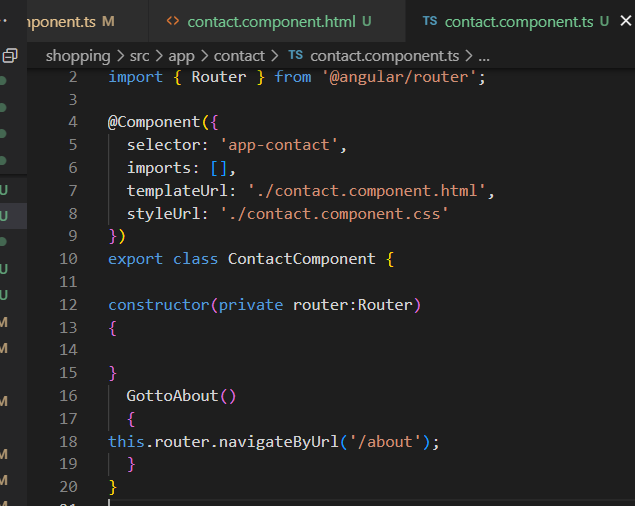


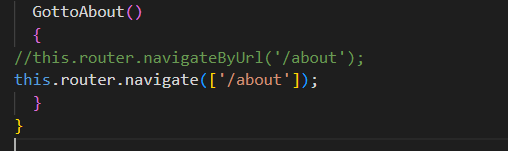
**RouterLinkActiveOptions :**

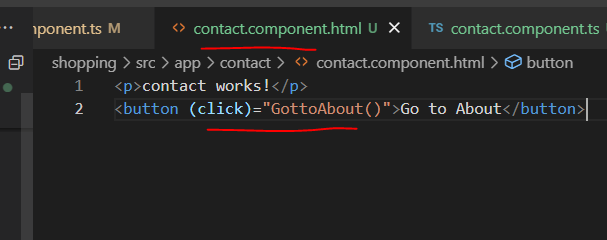


**If you want to Navigate Programiticallyfrom one page to Another**

**Use ‘Router’ add it in Constructor**

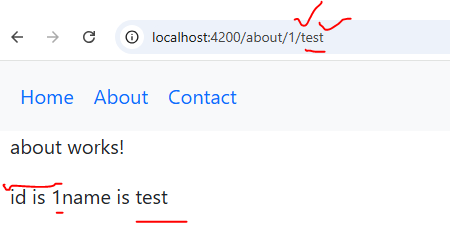


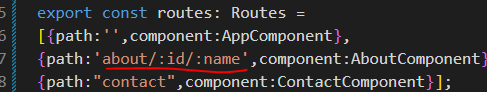


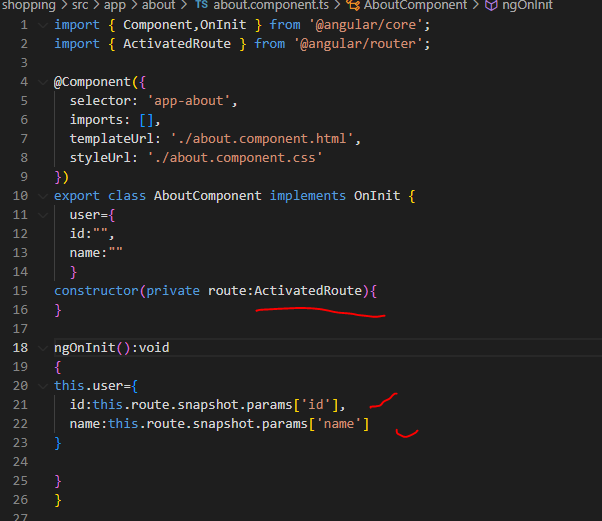


**Either by NavigateURl or Navigate method you can Navigate from one page to Another page**

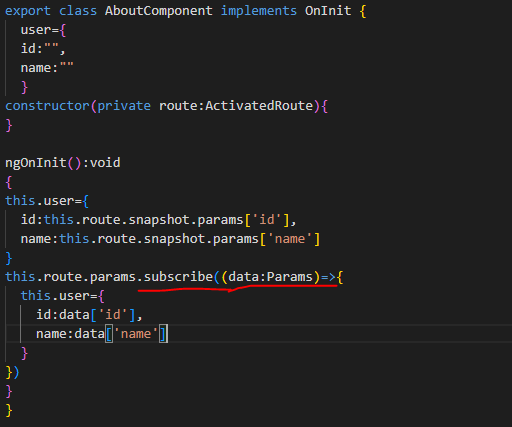
**How to Receive Value basedon uri**





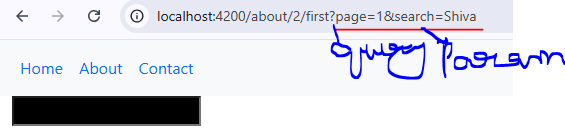


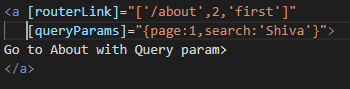




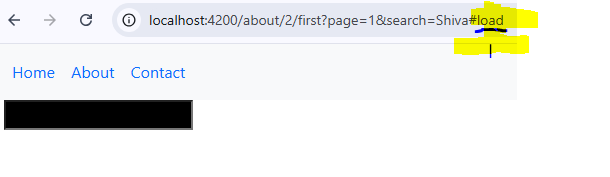
**How to Send Data as a Queryparam and Fragment?**

**Queryparam:**

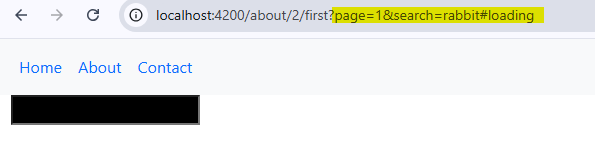


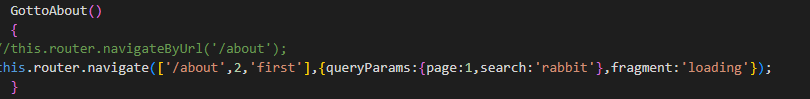


**Fragment**

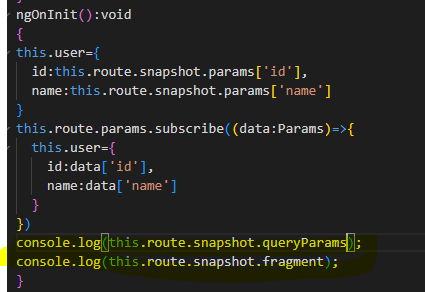


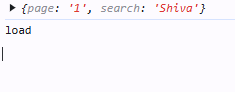
**Using Programmatically:**

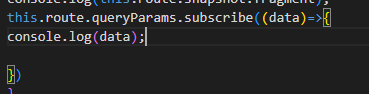


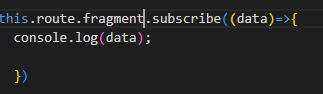


**Reading Query Param Data**

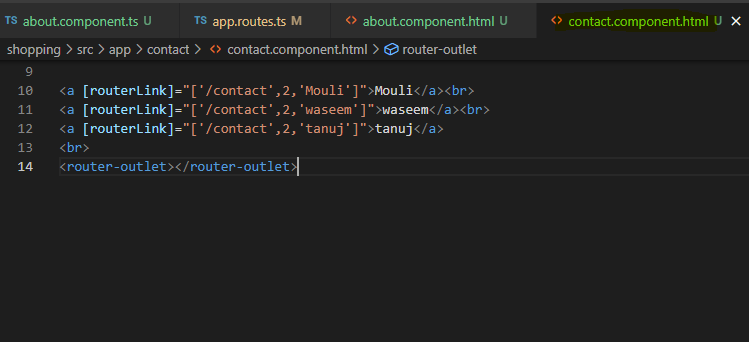




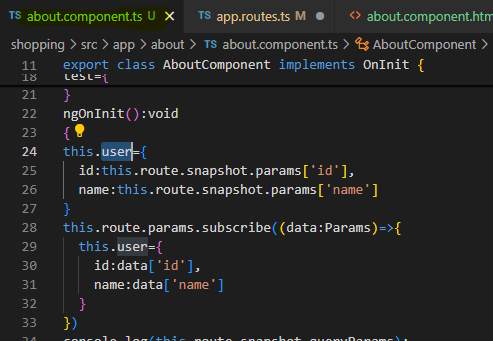


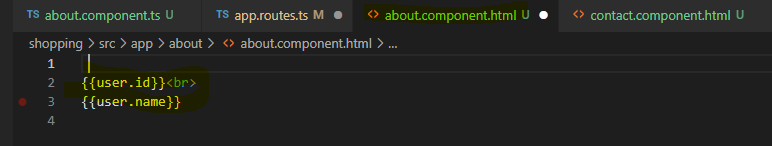


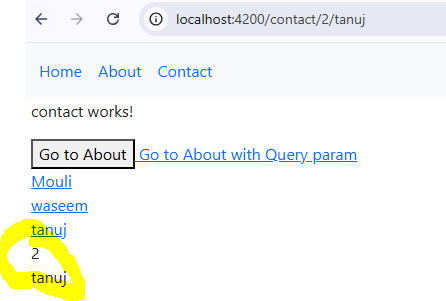
**Nested Routing in Angular:**





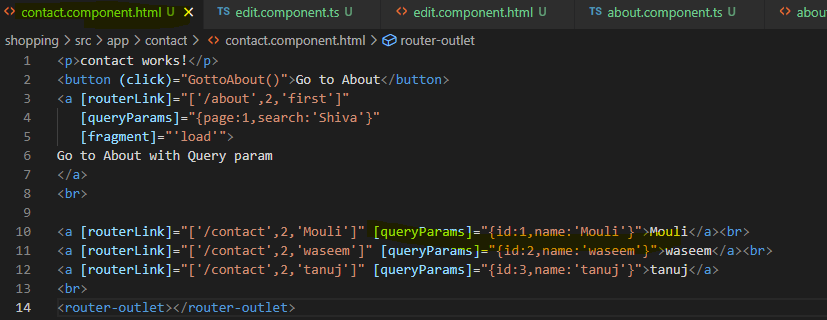




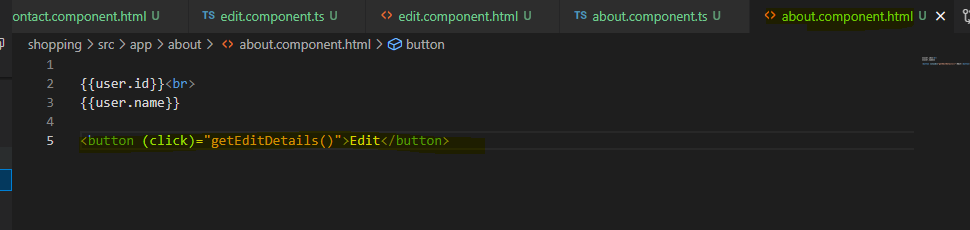


**QueryParameterHandling**:will preserve the queryparameter next page

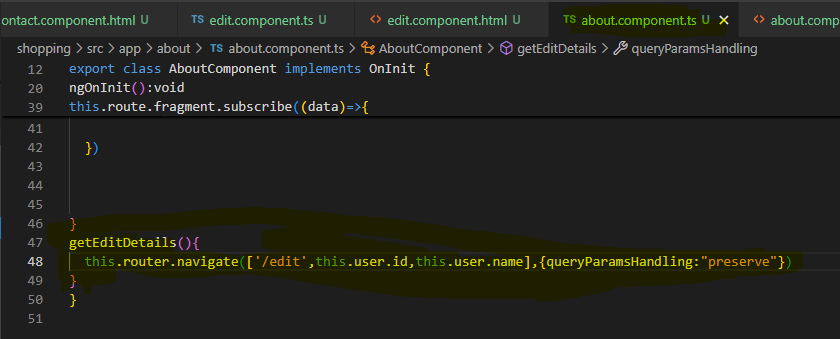
add in Component.html QueryParams



Add in About.component.html



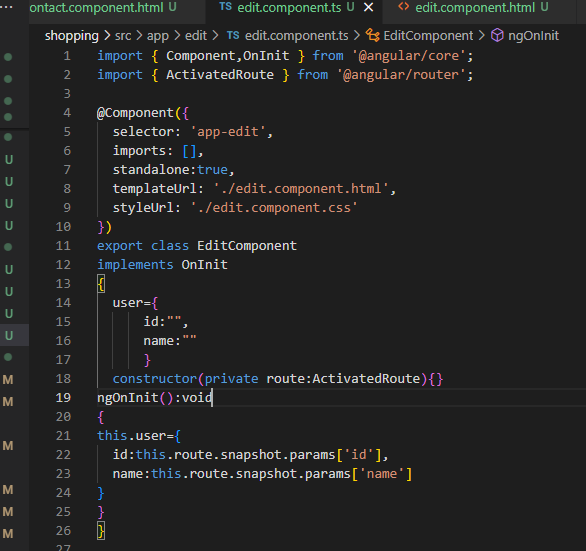
Add GetEditDetailMethod

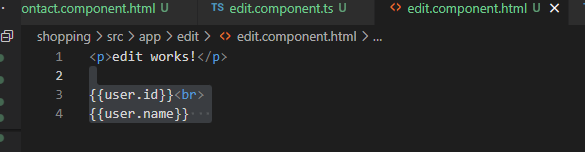


Add in Route Edit



Create Edit Component





How to implement Guard Service?

If you want to provide your based on crediantial redirect into routes you can take Guard Service

CanActivate:used for your current route

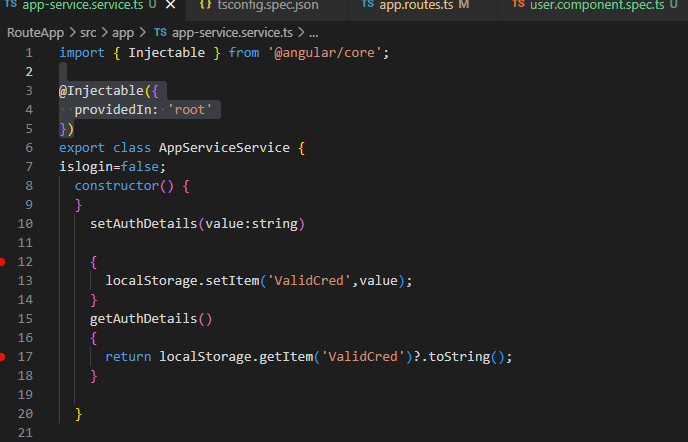
CanActivateChild:used for Child Route

CanDeactivate :called when we leave one Route to another Route

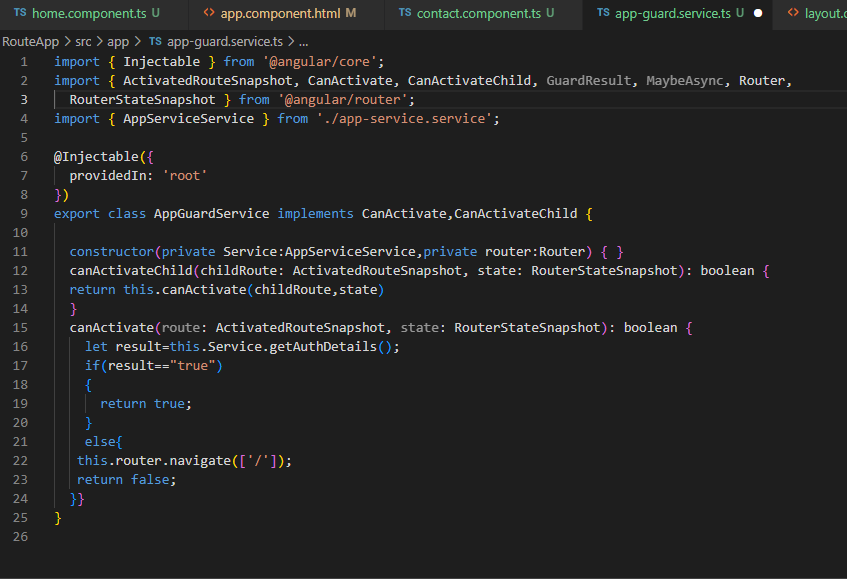
Example on CanActivate:

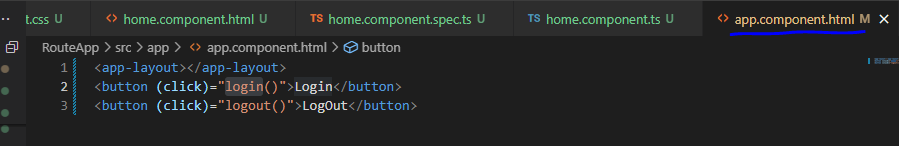
Create Service:

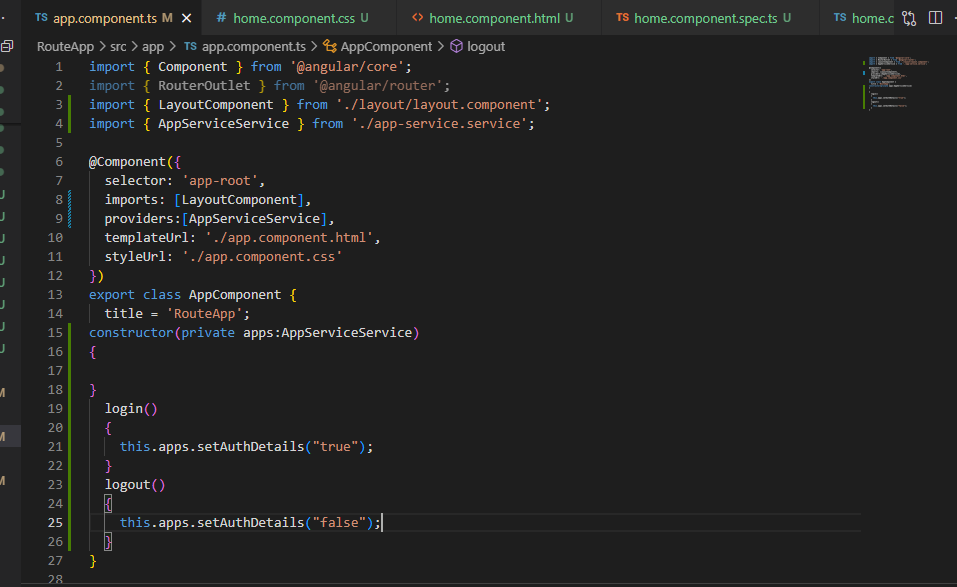
App-service using ng g s App-service

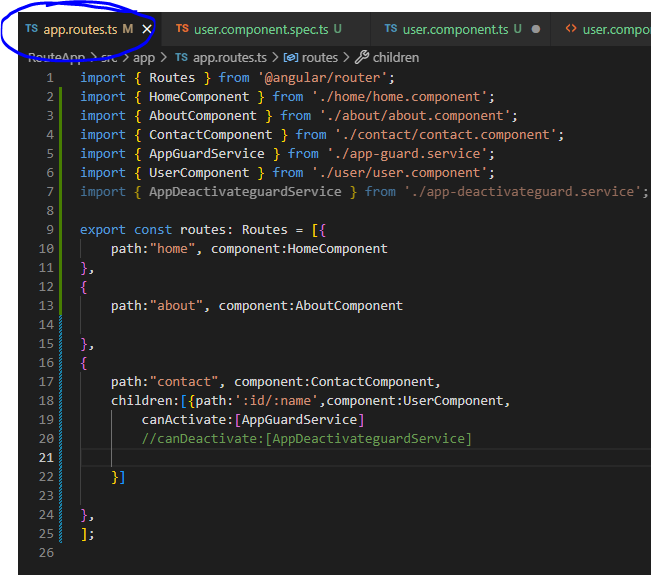


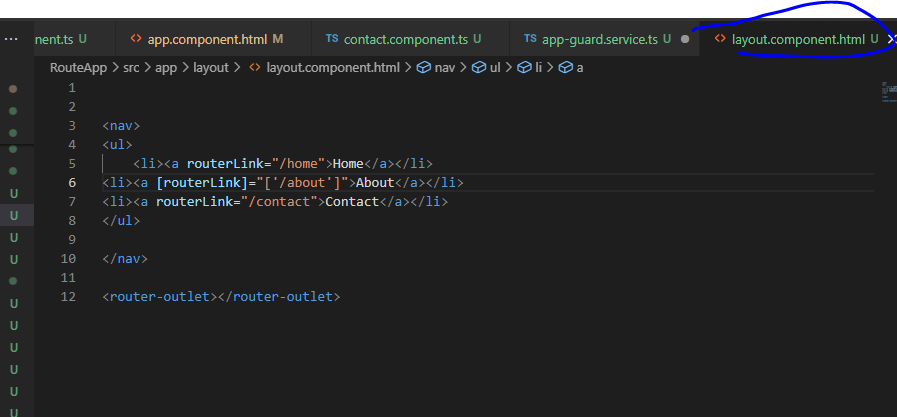
Create App-guard Service:Using ng g s App-guard

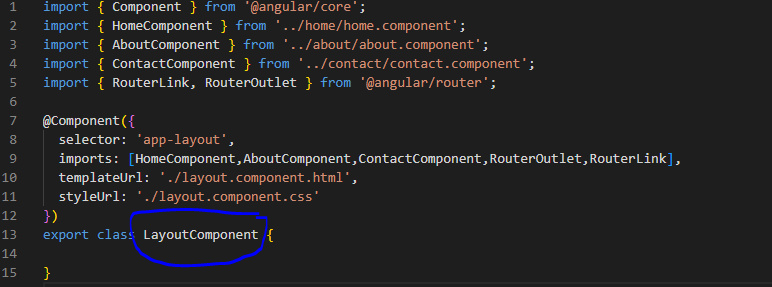


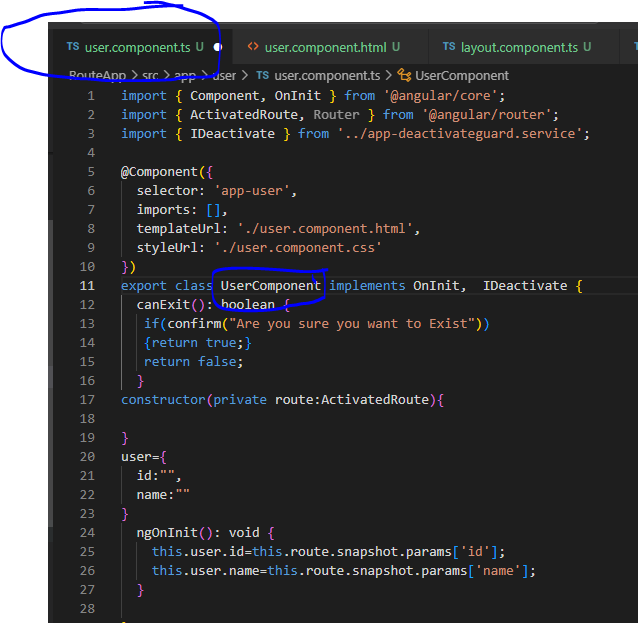


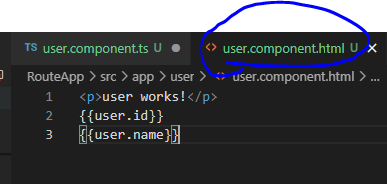






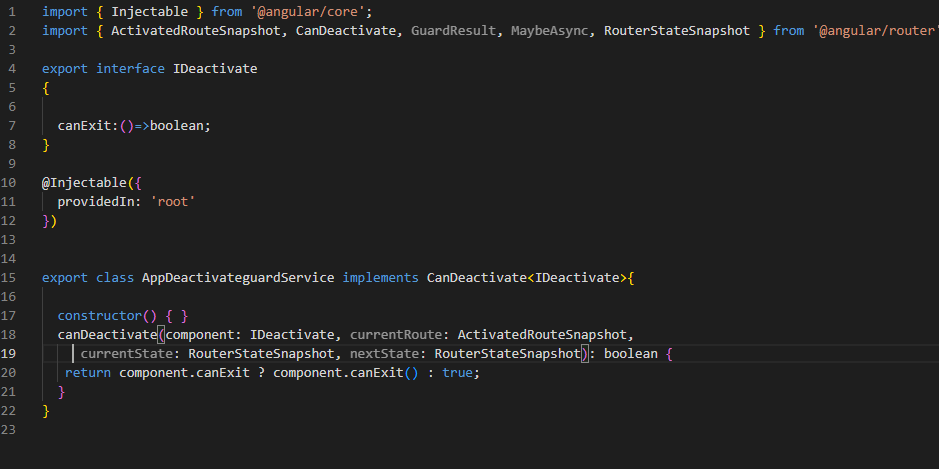


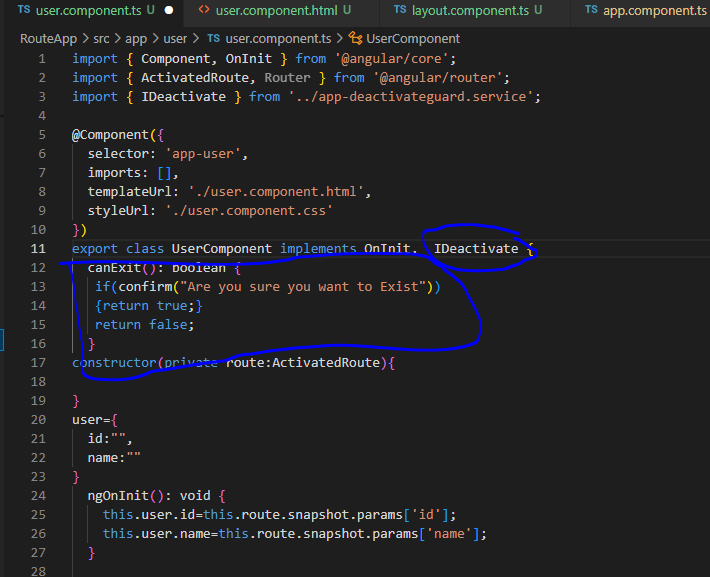


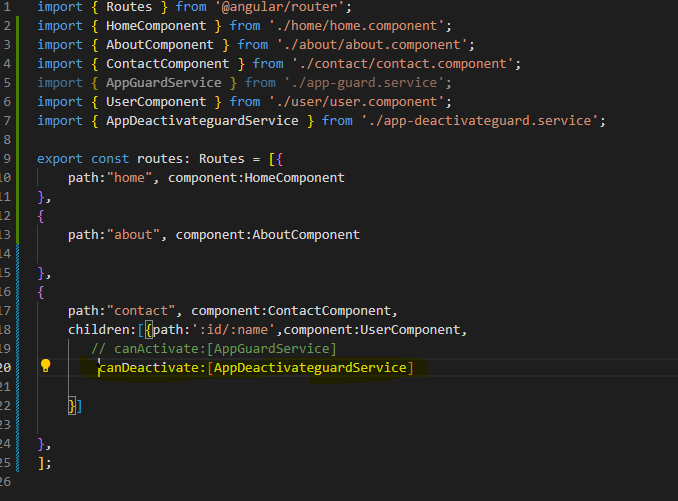


To Implement Deactivate Guard Service :

Create app-deactivateGuard.ts using ng g s app-deactivateGuard



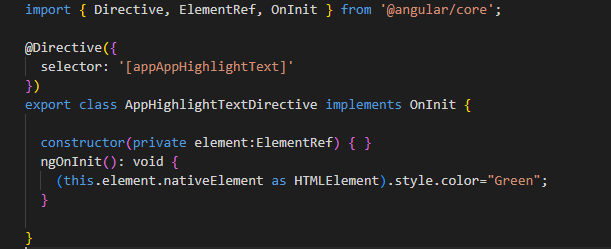




**Creating Custom Directive:**

type ng g d **AppHighlightTextDirective**

**will create Directive AppHighlightTextDirective**

****

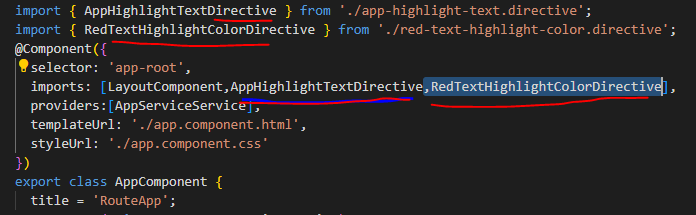
**Here with the help of ElementRef you can access native element**

**Next convert to HtmlElement**

**Then specify Style.color=red**

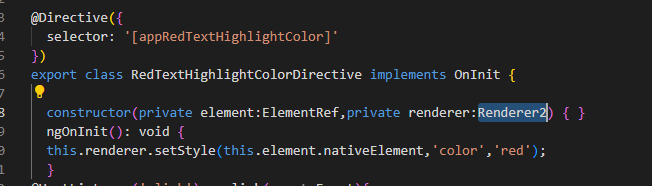
**How to Import Directive ?**

**Ans:Same as Component**

****

****

**Another way to Specify the Styles for it using** Renderer2 property



How to Here events for it? Using HostListener

