A framework is a condensed set of libraries. E.g. Angular, ExpressJs etc.

Libraries generally perform a specific operation. E.g. React, MomentJs, Lodash

**Features of Angular**

* Template Engine
* Data Binding
* Forms
* Routing
* Observables
* PWA

**Typescript** – a strongly typed Programming Language created and maintained by Microsoft. It’s a superset of Javascript.

It compiles to Javascript.

**Type Safety**

* Keeps your application free from Type Errors.
* Languages like C#, Java are examples of Type Safe Languages.
* Keeps your Js code free from undefined and null values.
* In Typescript types are stripped when your code is converted to JS.

**Installing and Creating First Typescript program**

* Install Node
* Use latest version of Nodejs
* **‘**npm init’ to create a package.json
* Install Typescript ‘npm i typescript’ Note : use npm install typescript@latest -g to install Typescript globally such that the tsc command can be recognized
* Initialize Typescript
* **‘**tsc –init’

**NOTE: Typescript is very essential when your codebase is huge and you’re likely to lose track of the different types of your functions / variables.**

**SPA – Single Page Applications**

Angular, React, or Vue, to create SPA.

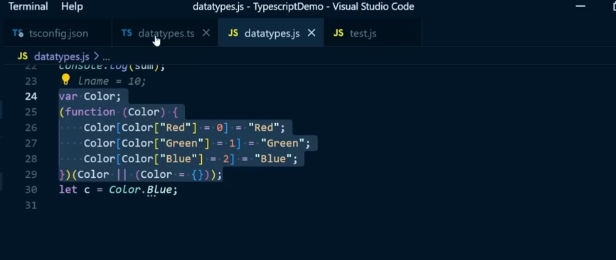
All resources needed to render the static pages are sent to your client’s machine. SPA does not make requests to server to every URL request, basically a full reload is not needed.

Angular has Routing functionality to create SPA

Angu lar also offers SSR (Server-Side Rendering) which supports SPA.

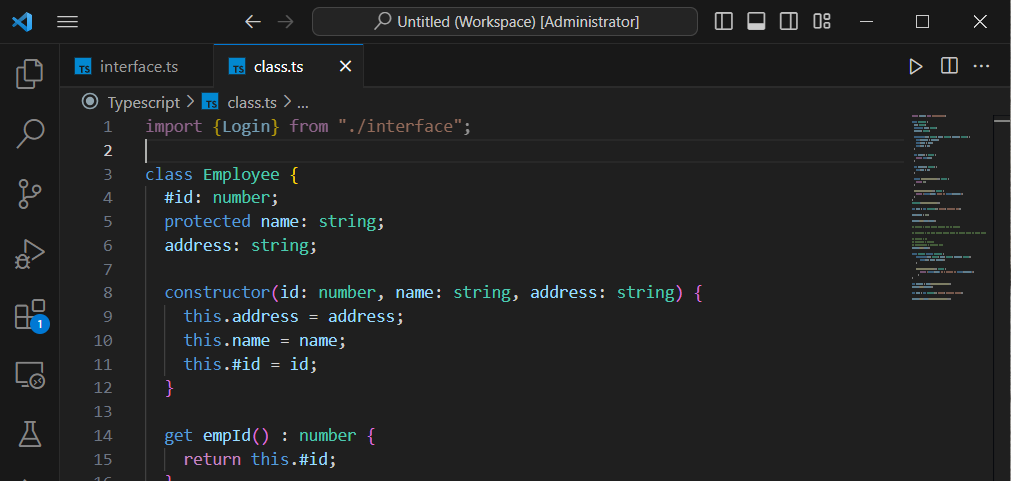
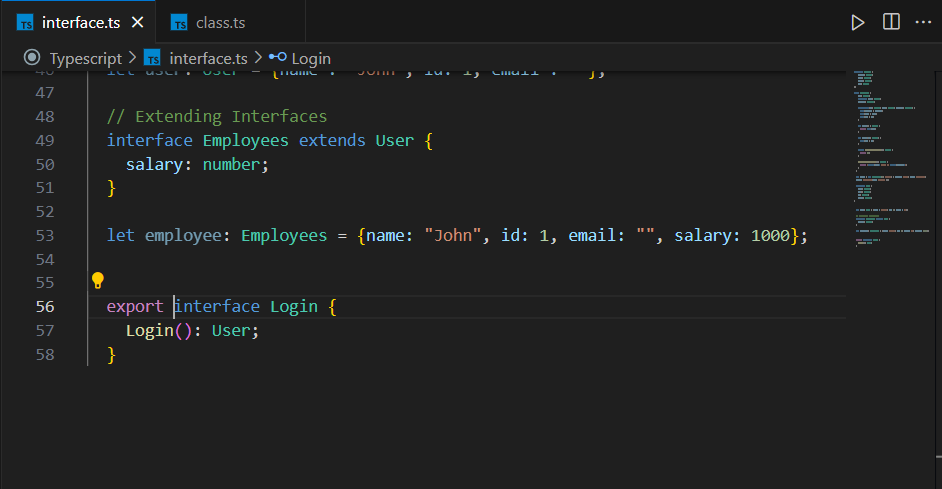
Note: SSR is better for SEO than SPA.

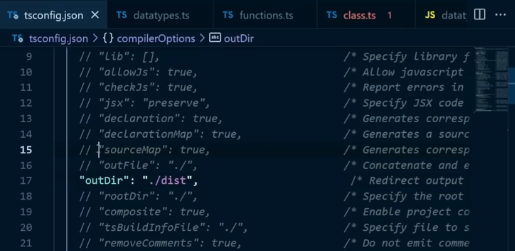
**Typescript Datatypes**



In typescript declaring an enum as const will trim down the value from what’s compiled above to a single line of code

Exporting an interface and using it in a different module



Setting your outDir specifies a specific directory where all the compiled JS files are placed.

By setting “sourceMap” to true you generate a file that let’s you know where exactly the compiled JS file came from.

By setting “noEmit” to “false”, when you compile your typescript file you won’t get a JS file.

**ANGULAR**

**Installation**

npm install -g @angular/cli

or Use npx to avoid installing the global version

Creating Angular workspace

ng new hotelinventoryapp –createApplication=false **OR** ng new hotelinventoryapp

Note:

* package-lock.json help you most especially in big projects to make sure that all developers are working on the same version of different dependencies.
* browsersList shows the browsers that support angular applications
* To install a **specific version** of angular, first download the cli tool locally eg **npm i @angular/cli@14.1.3** then run **npx ng new your-project-name**.
* Run the server using ng server or ng serve -open / ng serve -o to open your browser as well

**Mono-repo**

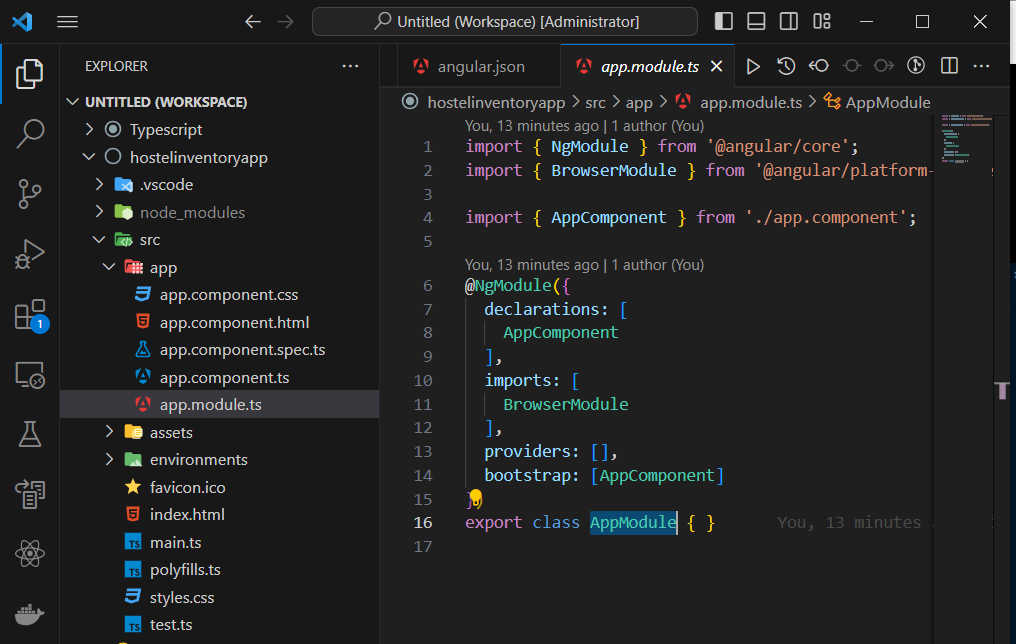
Involves creating and maintaining multiple apps in the same repo

Use libraries within the project

Deploy multiple apps/libs from the same repo

Easy to share code within the project

Under angular.json the styles attribute is use to set where your styles are located

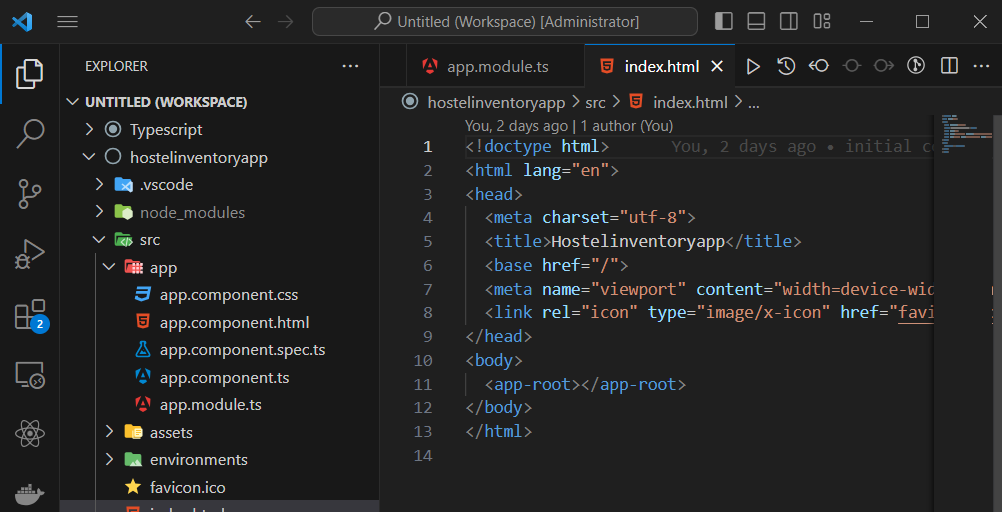
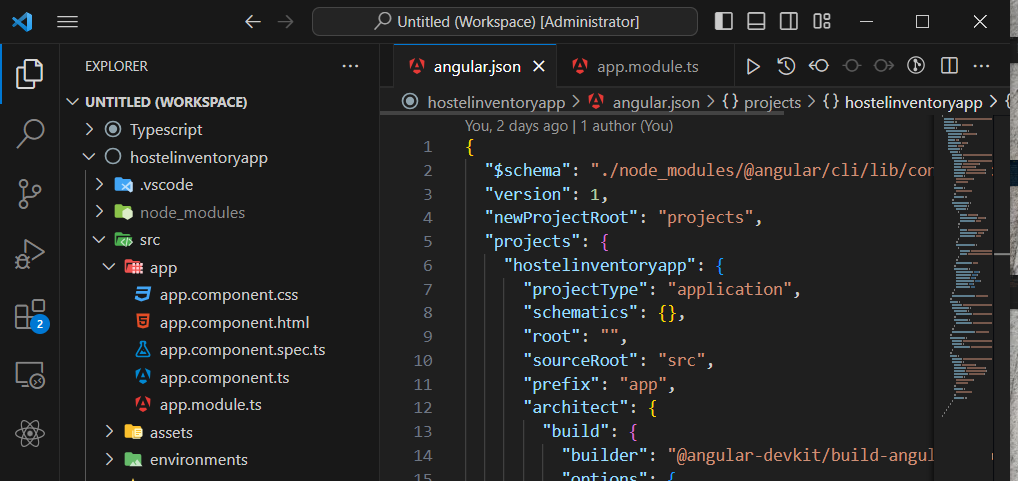


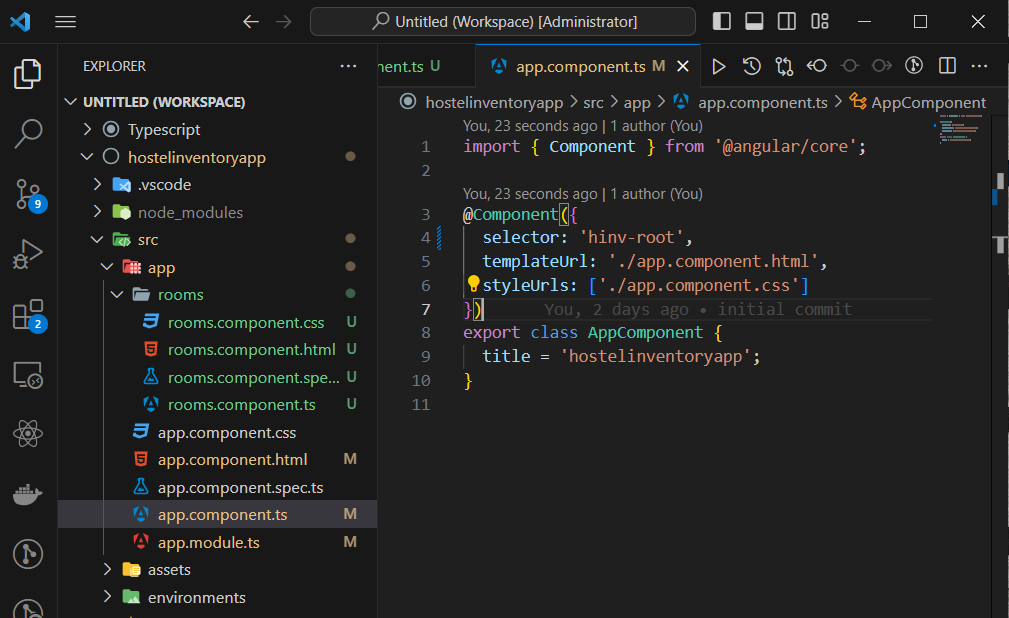
This is the root module. You can create any number of modules but you need at least one root module. In Angular 17 it’s not available.

Any components, directives and pipes need to be registered by the declarations array in the app.module.ts.

Components are views that are rendered to your user. You need one root component i.e. the AppComponent

Whenever you’re creating a new component, have a prefix. By default, it’s app. But in the angular.json file under the prefix property, you could change the **prefix.** And make sure to update the selectors in the component.ts files.

****

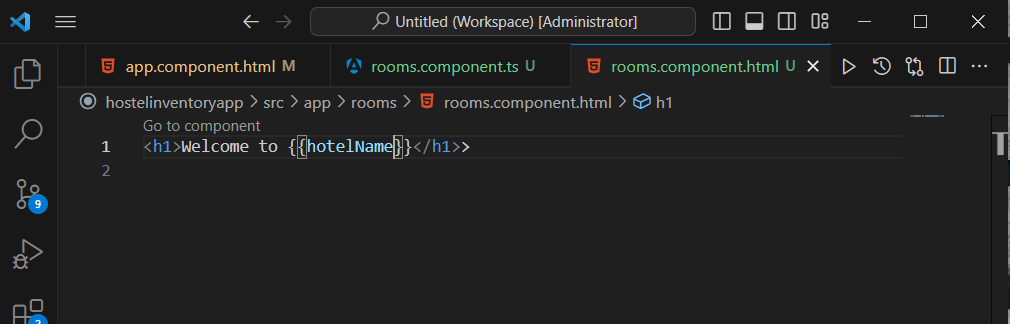
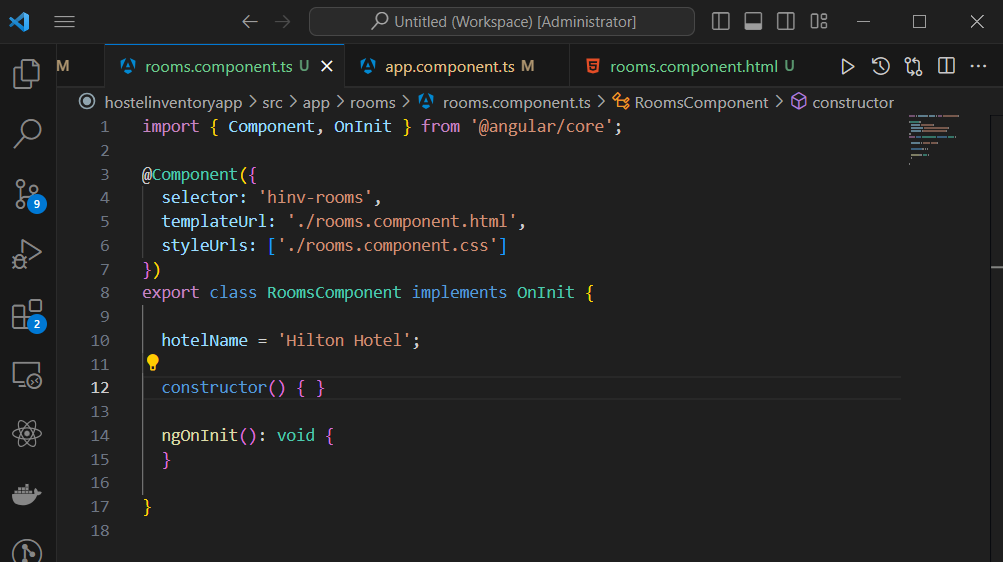
****

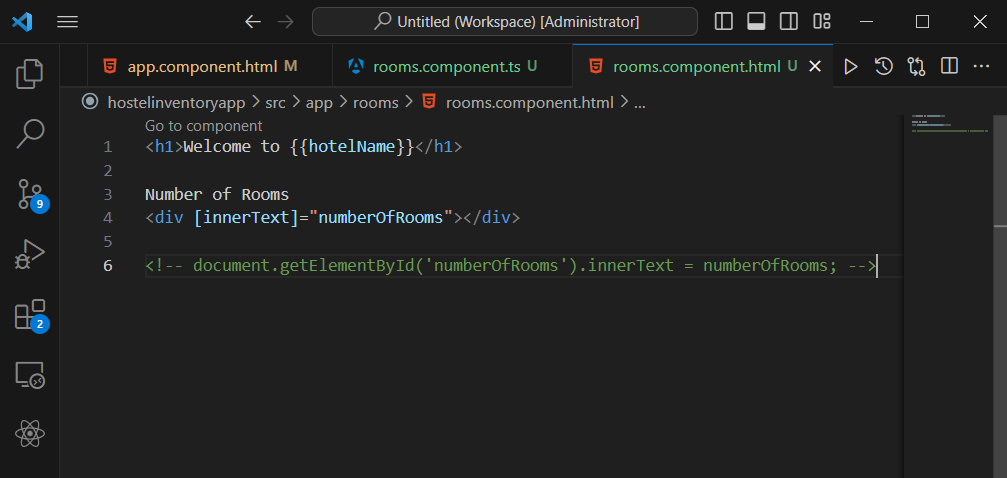
**Note:** angular.json files are not watched hence you’ll have to restart your application if you make changes to them.

* To **create a component**, in the command line type:

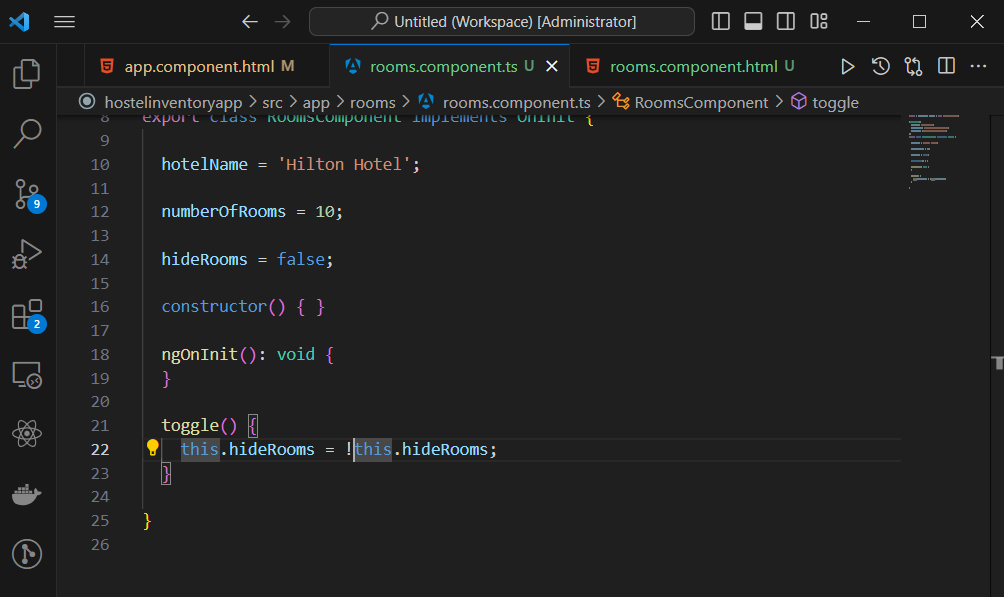
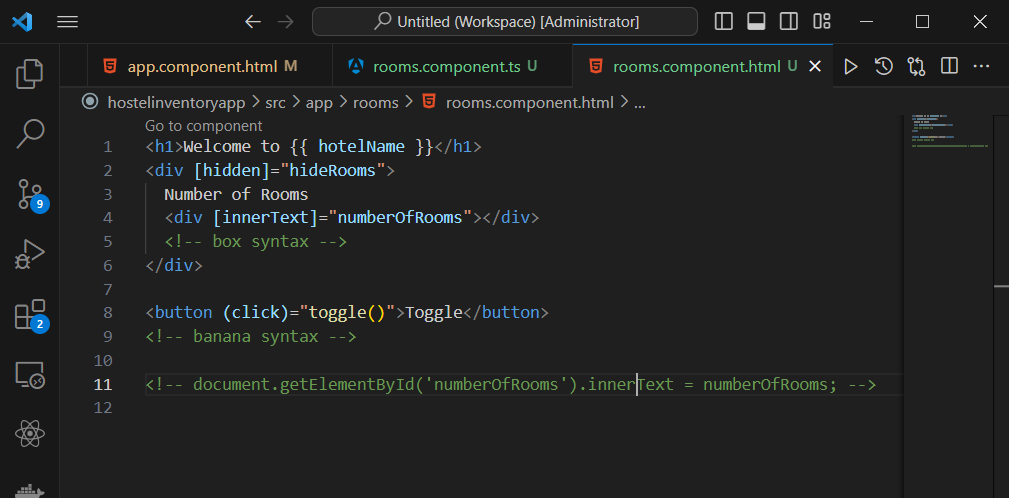
**ng g c <<component\_name>>**  also ng generate component <<component\_name>>

The binding syntax in Angular Includes:

**Interpolation **

**Property Binding** – used with any valid html property

**Event Binding**



You can debug your application by going to **Sources panel in the Browser Inspection, press CTRL + P, search for the filename and add breakpoints accordingly.**

**Directives**

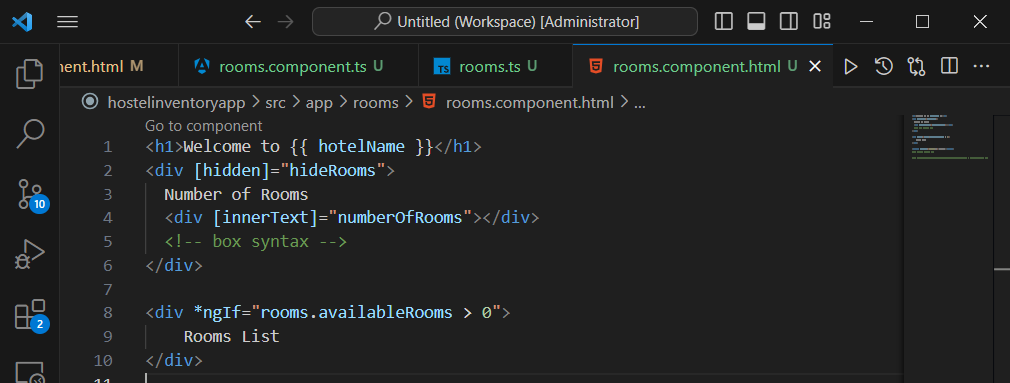
* They are used to change the behavior and appearance of DOM element.
* Directives can implement all lifecycle hooks.
* They cannot have template.

Types include

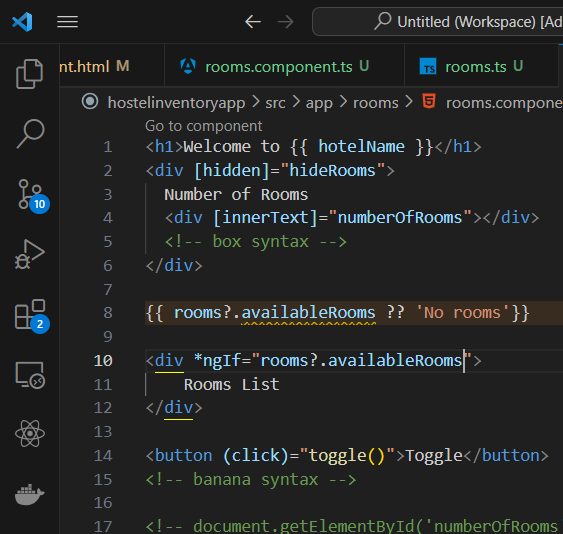
* Structural directives – costly because they modify the behavior of your DOM in terms of adding or removing elements from the DOM
* Attribute directives – mostly used to just merely modify the DOM without adding or removing elements from the DOM.

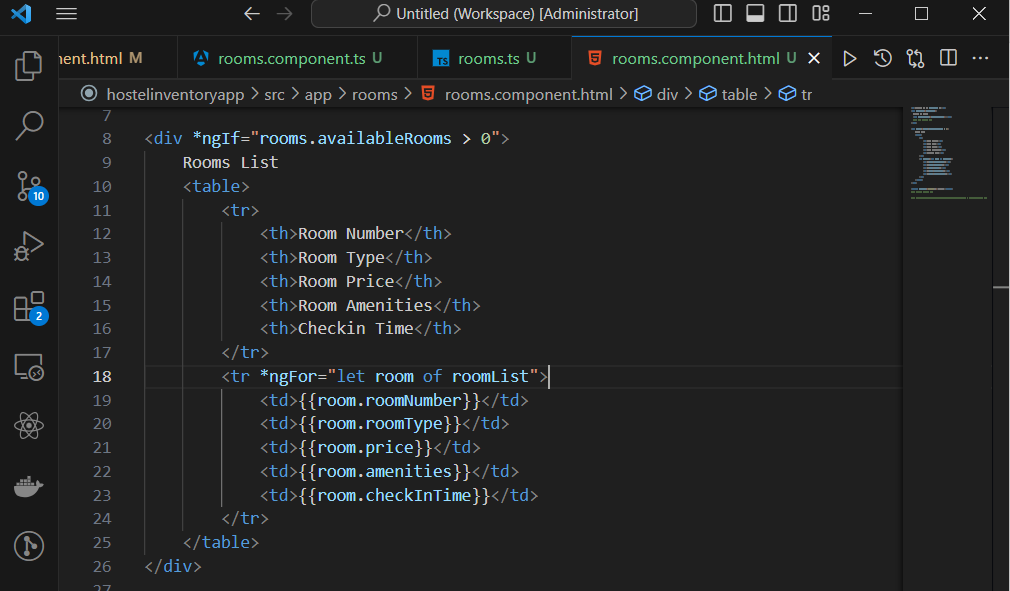
**Built-in directives** – the structural directives have \* ahead of them

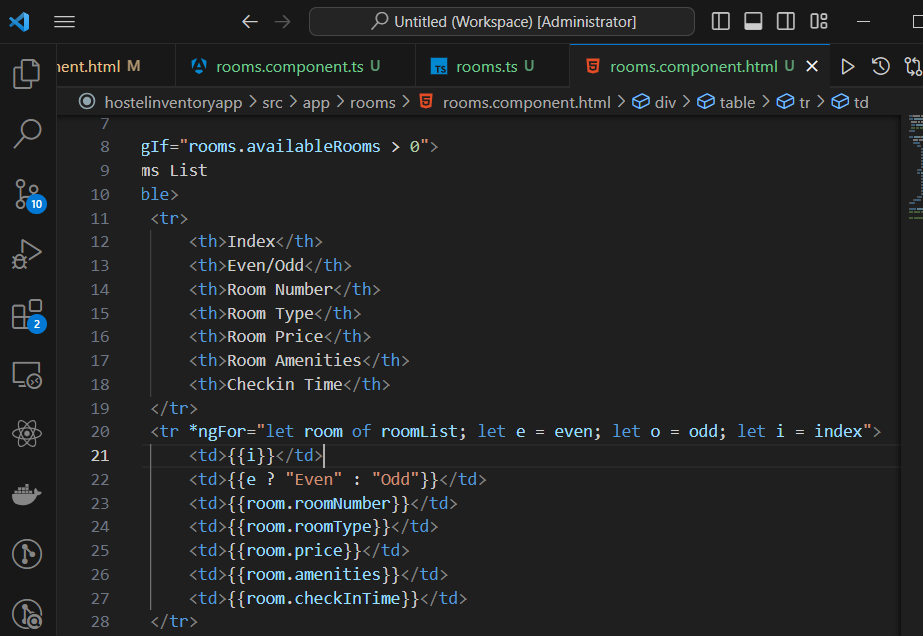
**\*ngIf**

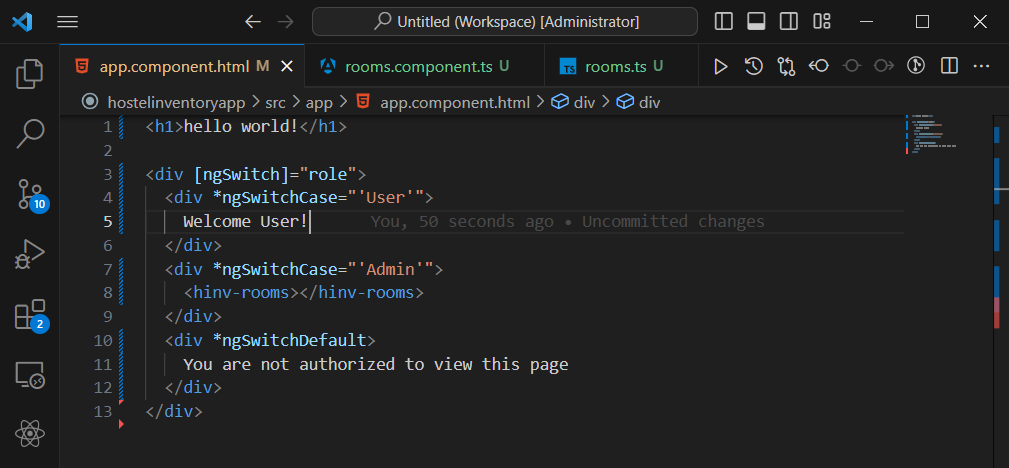
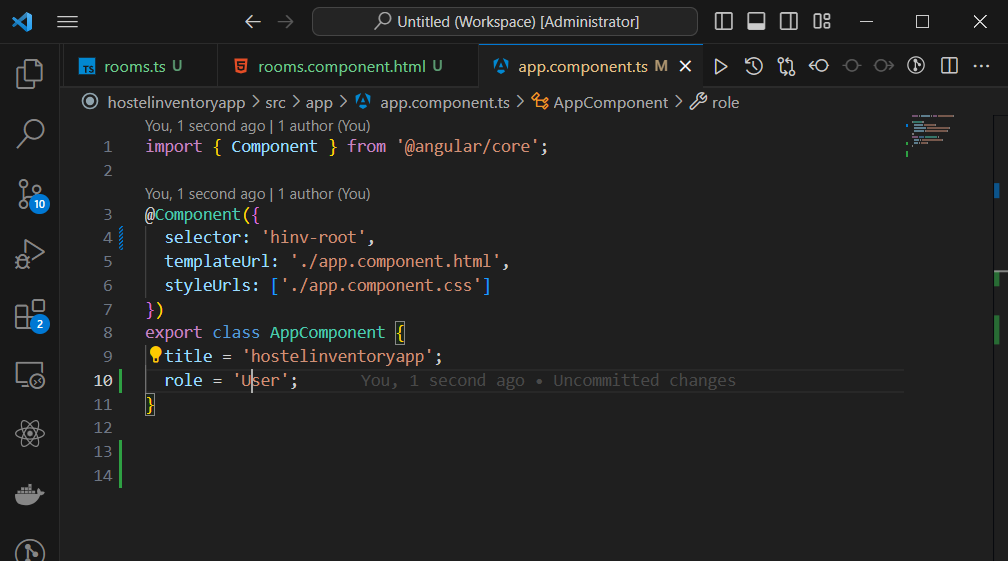
****

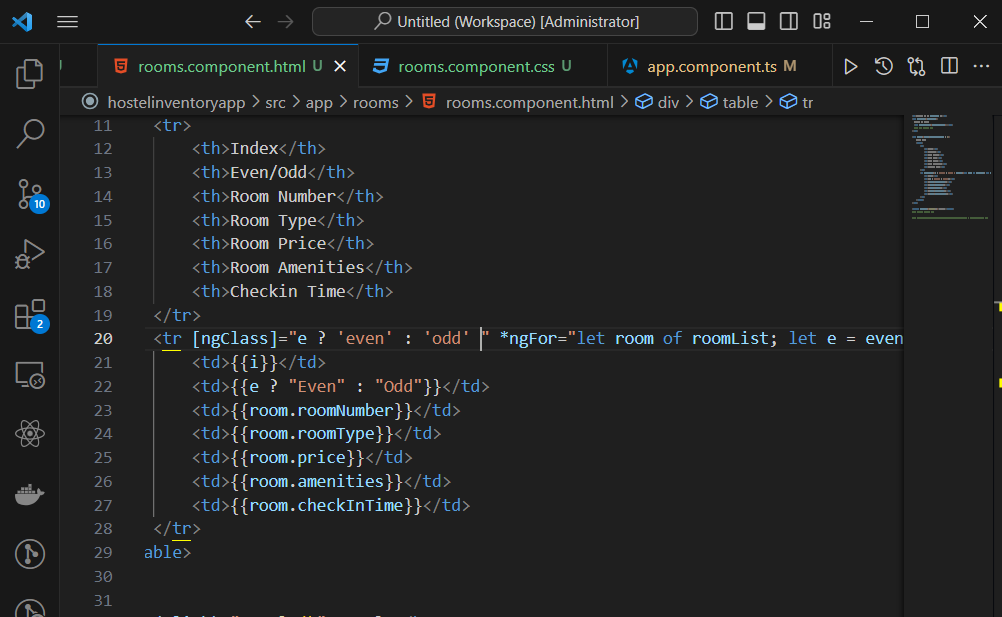
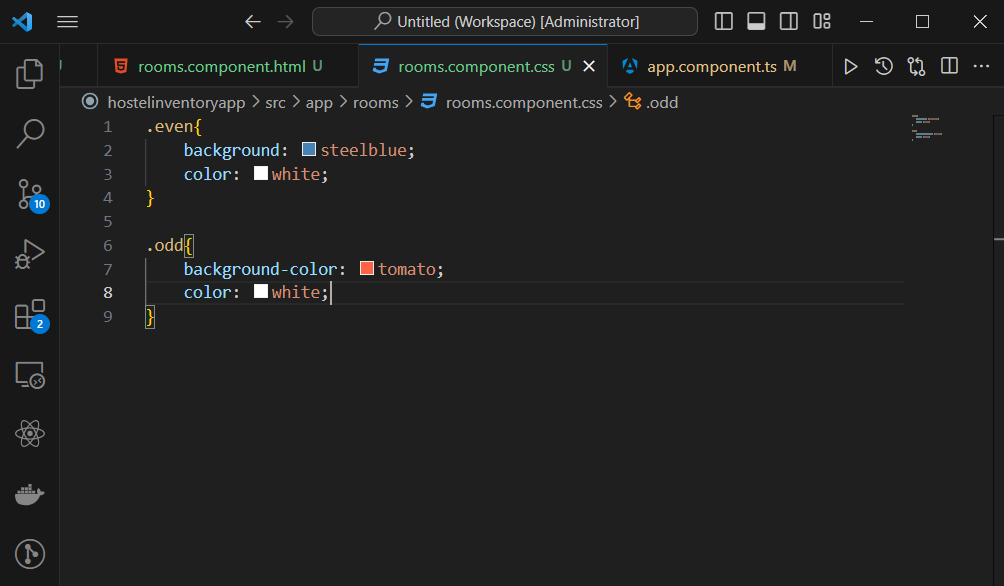
Optional chaining



**\*ngfor**



**\*ngSwitchCase**

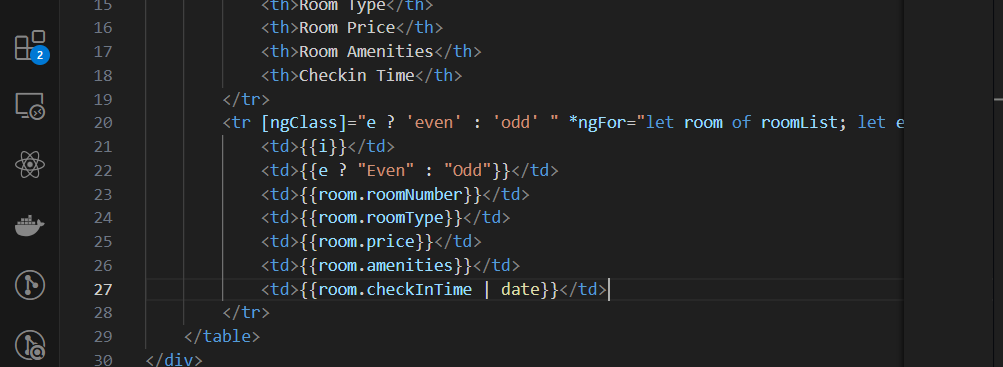
**ngClass**

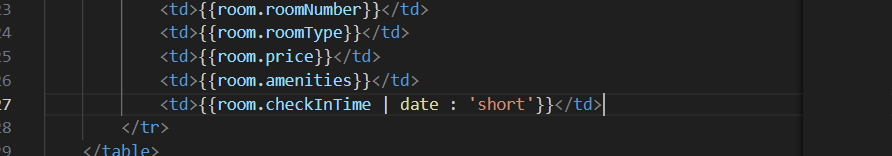
**Pipes**

* Pipes are used for data transformation
* Pipes don’t change the actual object

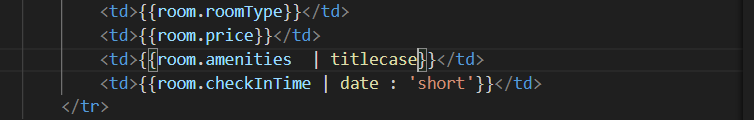
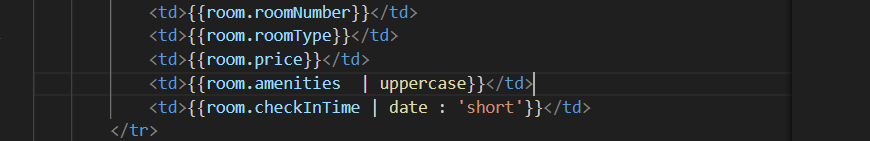
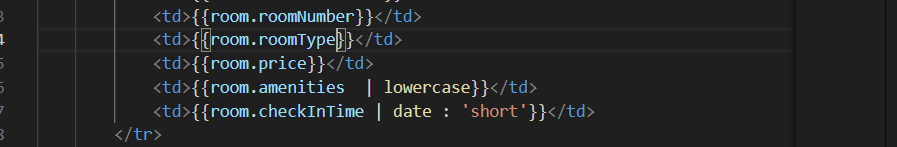
**Built-in pipes**

1. Date pipes

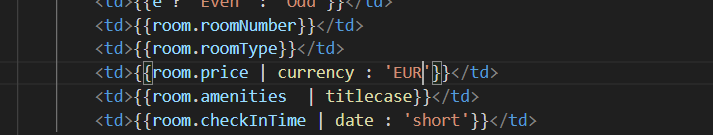
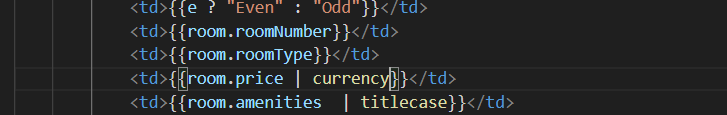




1. Uppercase pipe, Lowercase pipe and Titlecase pipe



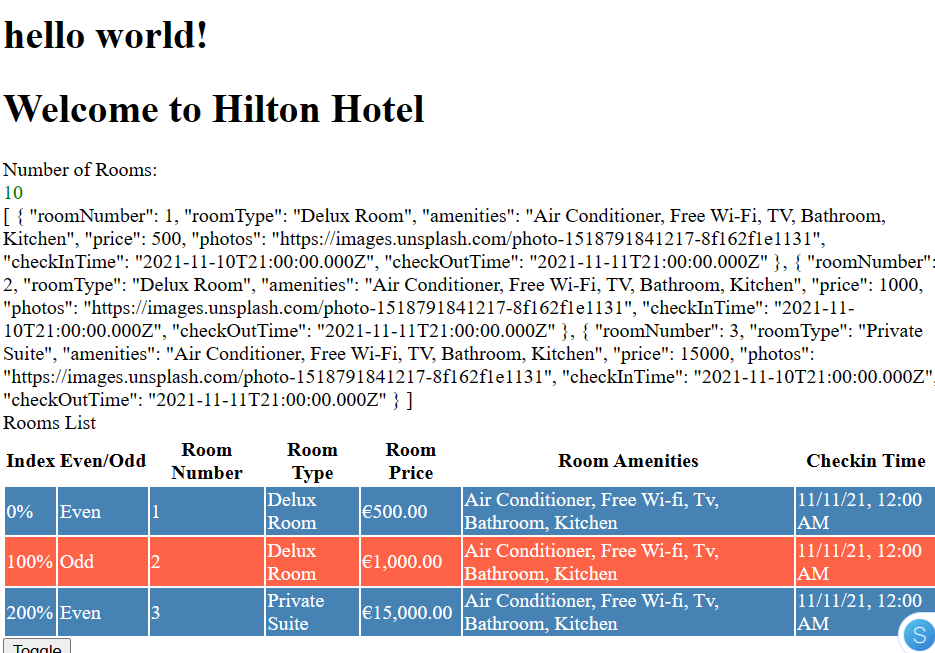
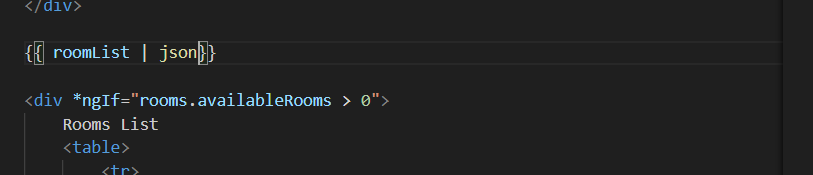
1. Currency pipe

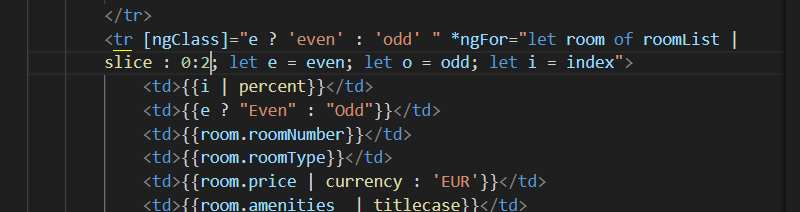


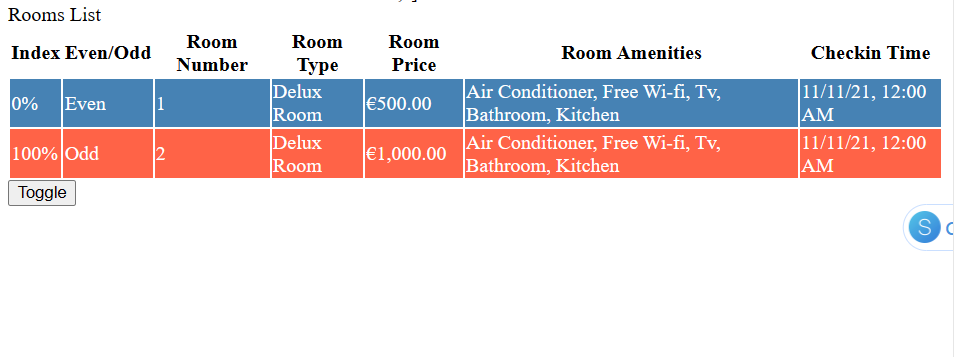
1. Percent pipe



1. Json pipe

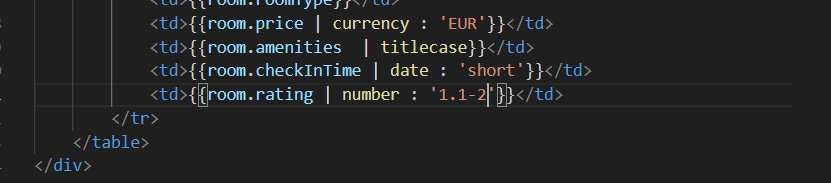


1. Slice pipe 



Used to cut down the number of records to show excluding the last one. i.e. start from index 0 and exclude the last one.

1. Decimal pipe



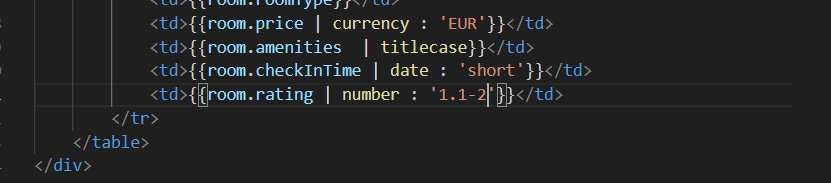
Here we want a maximum of 2 decimal point value

**NOTE:**  To add bootstrap to your angular application, go to **ngx-bootstrap**  and follow the instructions.

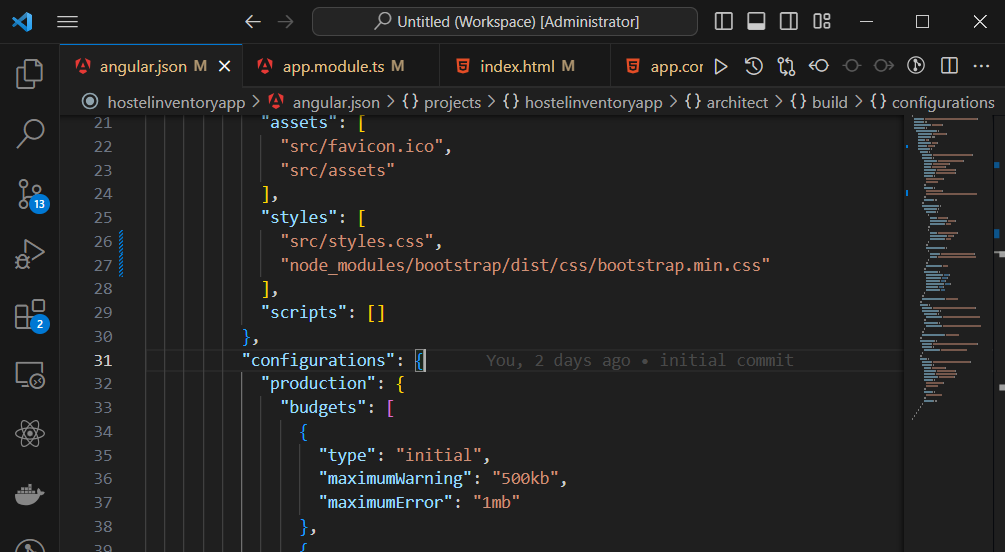
ng add ngx-bootstrap

or use more stable **npm i bootstrap**

Adding styles to angular

In your global css file add

Or in the angular.json file



LifeCycle Hooks

ngOnInit will be good for adding code that handles data from an api.

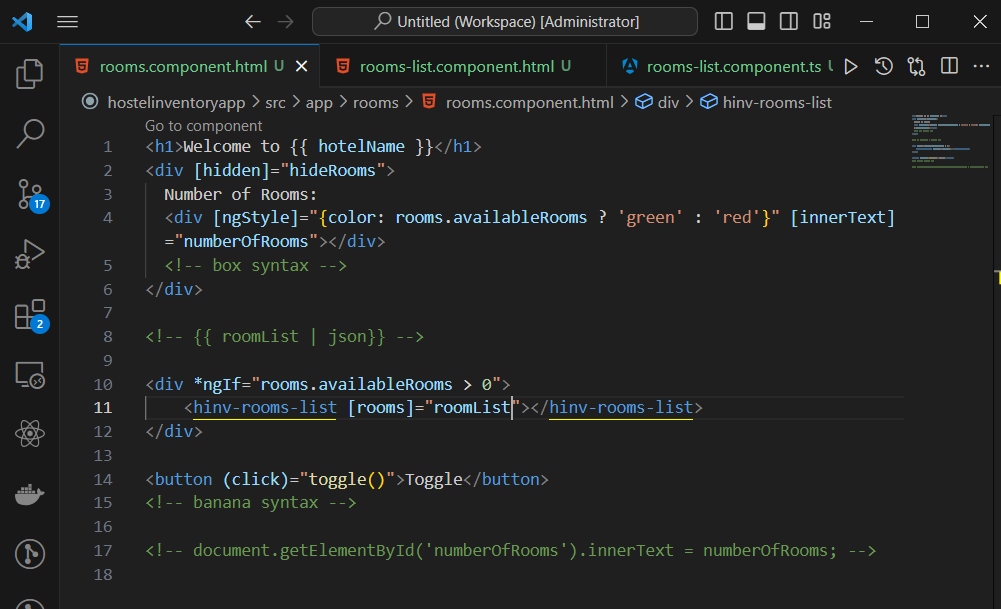
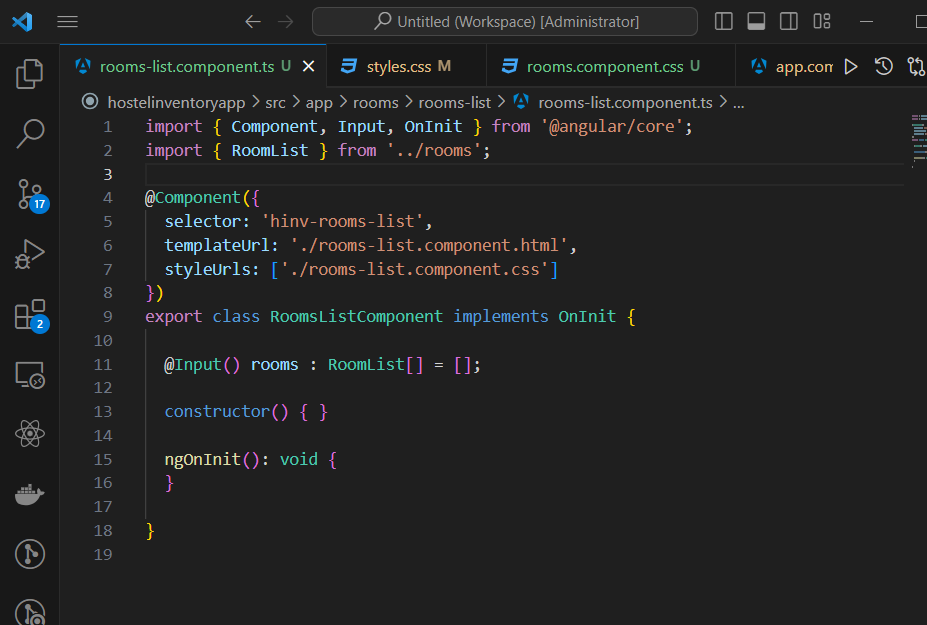
**Component communication –** the scenario where two or more components interact with each other.

Note: get angular dev tools

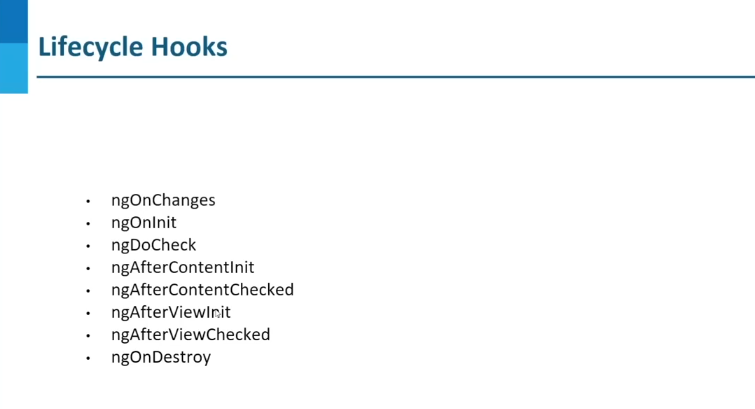
Ways they communicate.

* Using @Input and @Output
* Using @ViewChild and @ContentChild
* Using Services

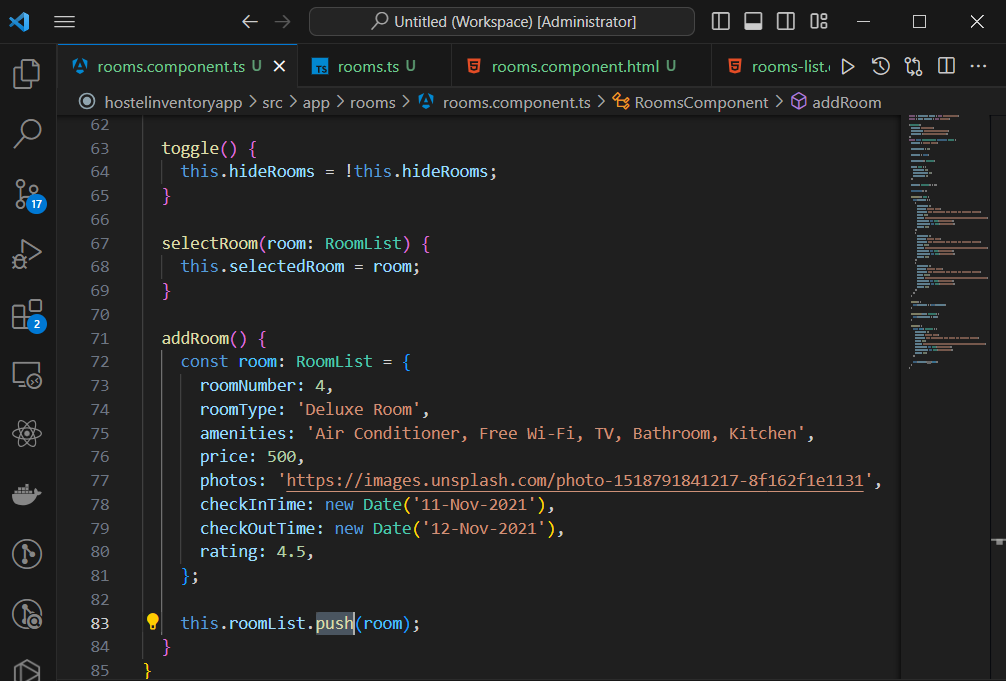
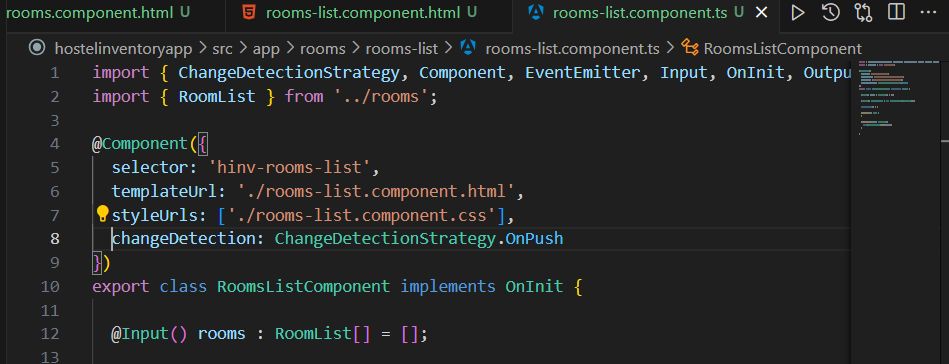
@Input



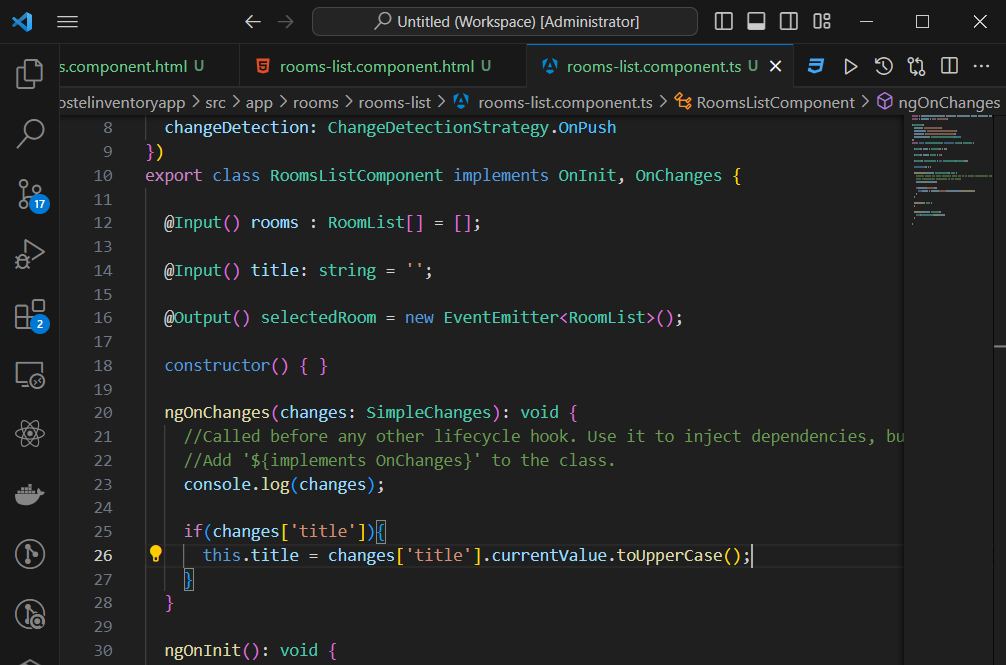
@Output is used to pass data from the child class to the parent class. Eg when you select a button in the child class and you catch that event from the parent class.



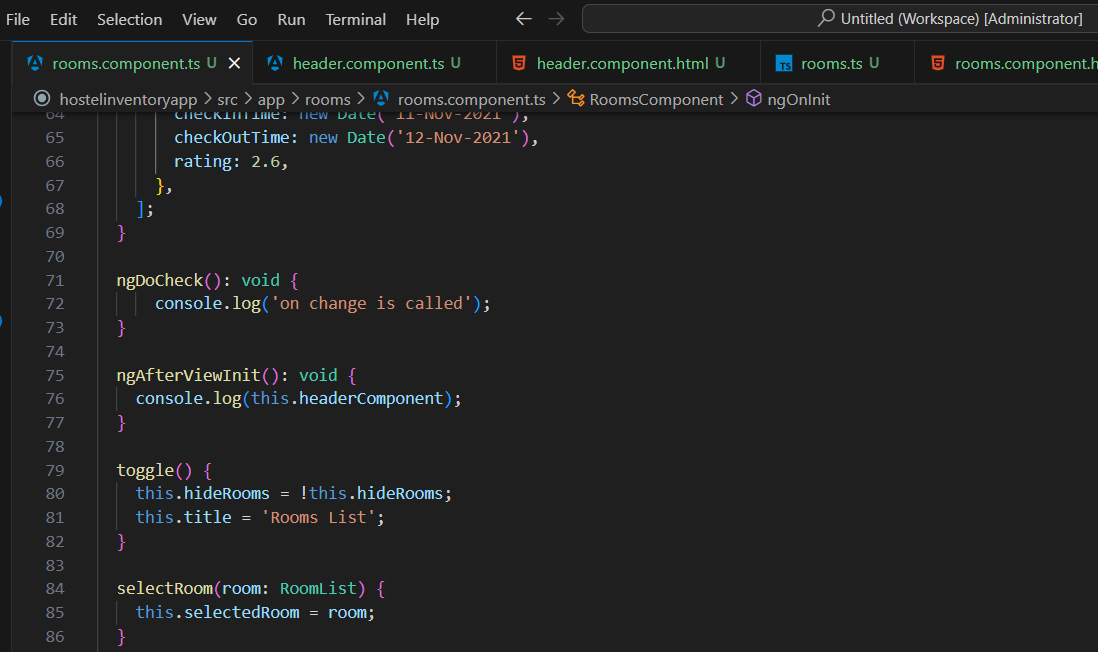
ChangeDetectionStrategy.OnPush change detection strategy is not used if you’re not making a change internally in a component. Always make sure to deal with immutable values hence don’t directly change the internal properties. In other words return a new instance of the existing object / value.





ngOnChanges

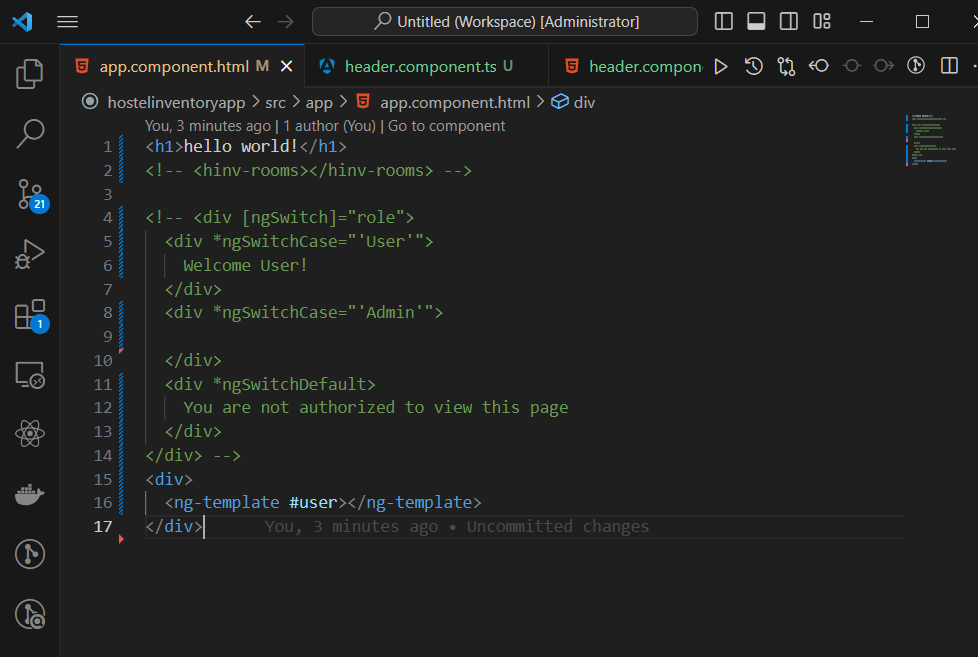
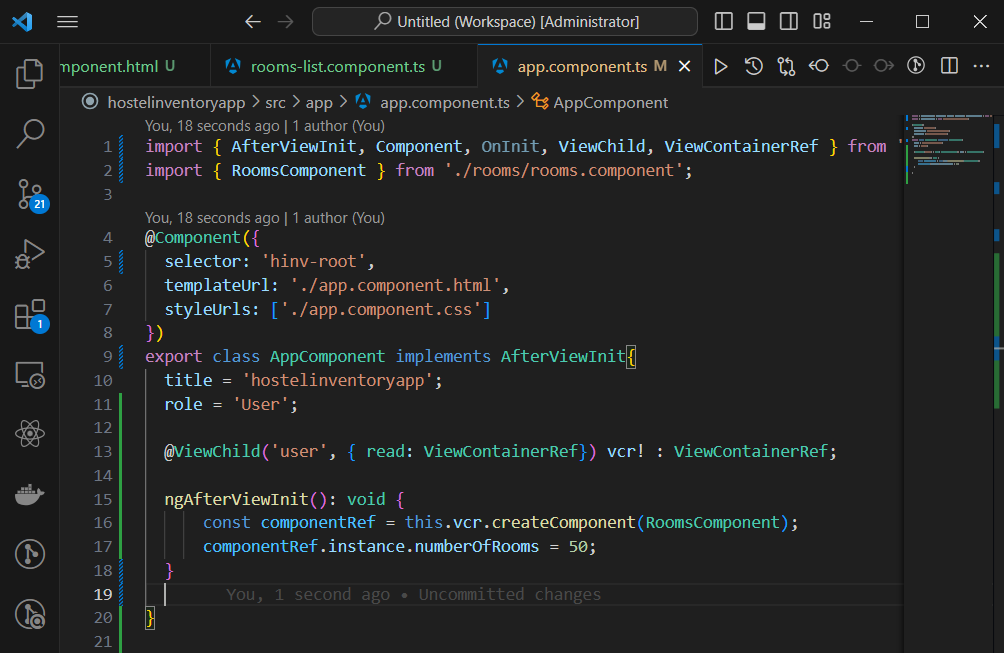
Don’t implement ngOnChanges and ngCheckChanges on the same component. doCheckchanges are very very rarely used.

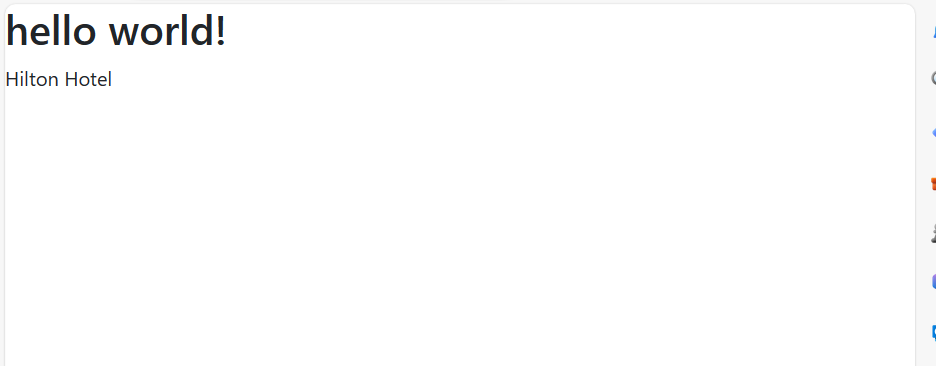
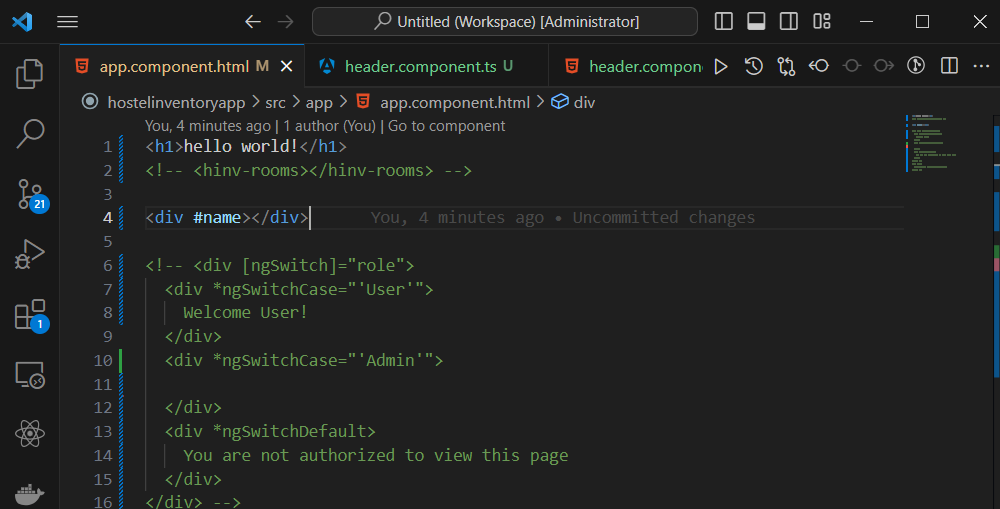
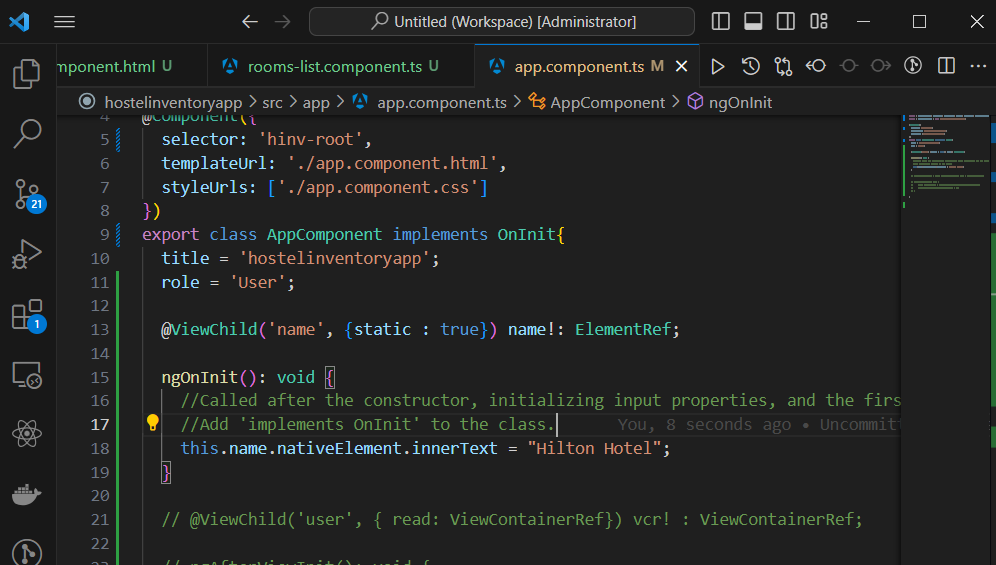


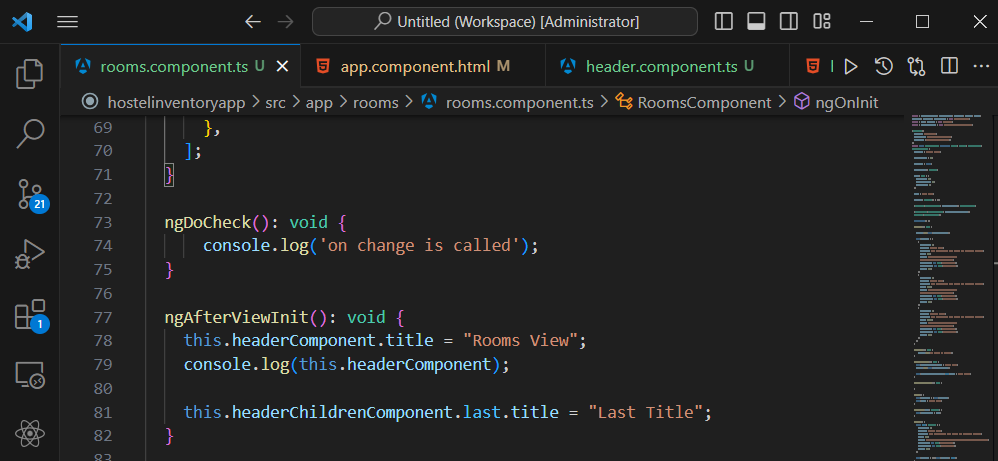
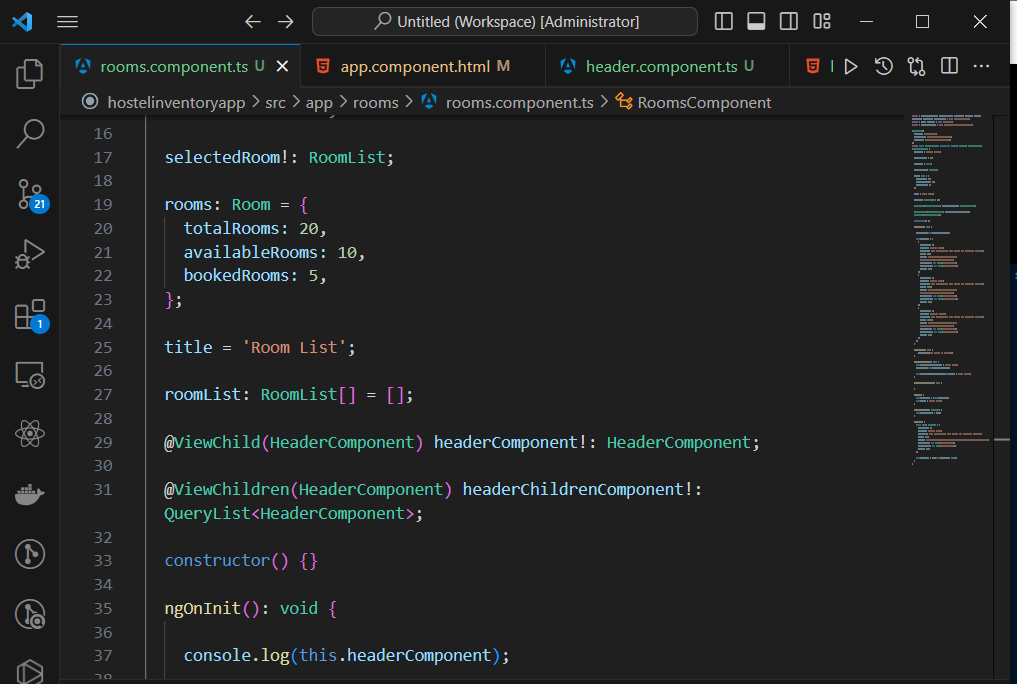
#user is a template ref

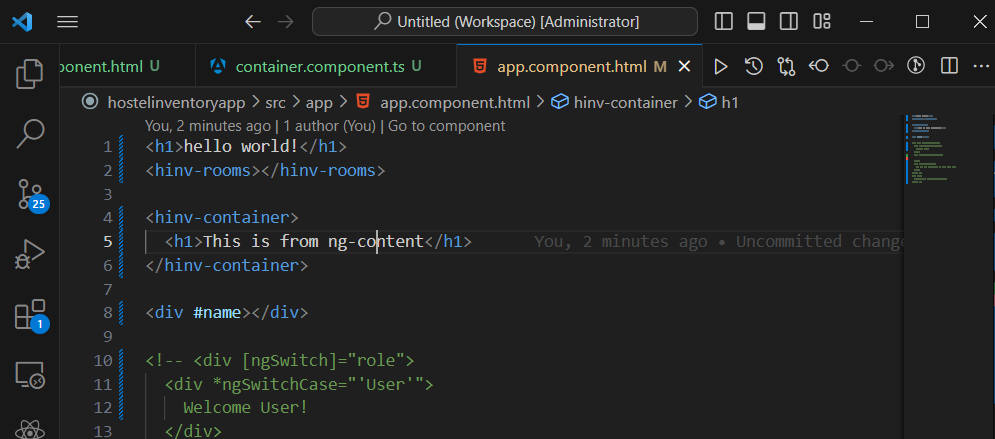
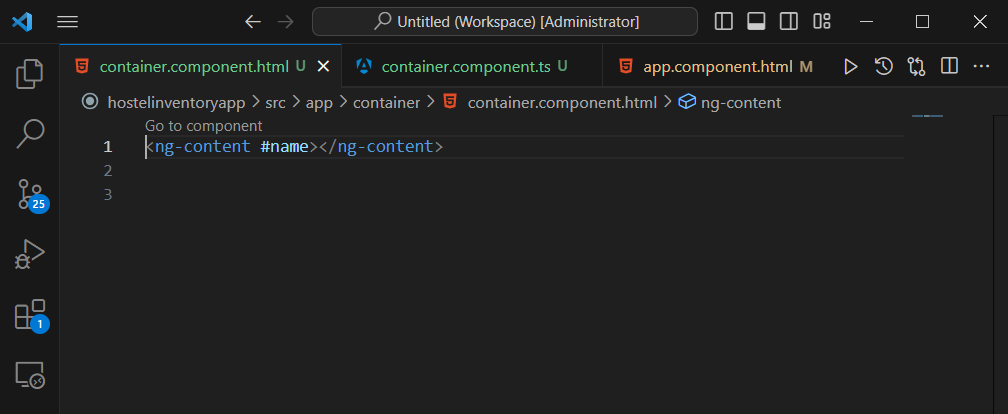
You can modify a component dynamically using @ViewChild and AfterViewInit. @ViewChild will access only the first instance of your component and @ViewChildren will access multiple instances.

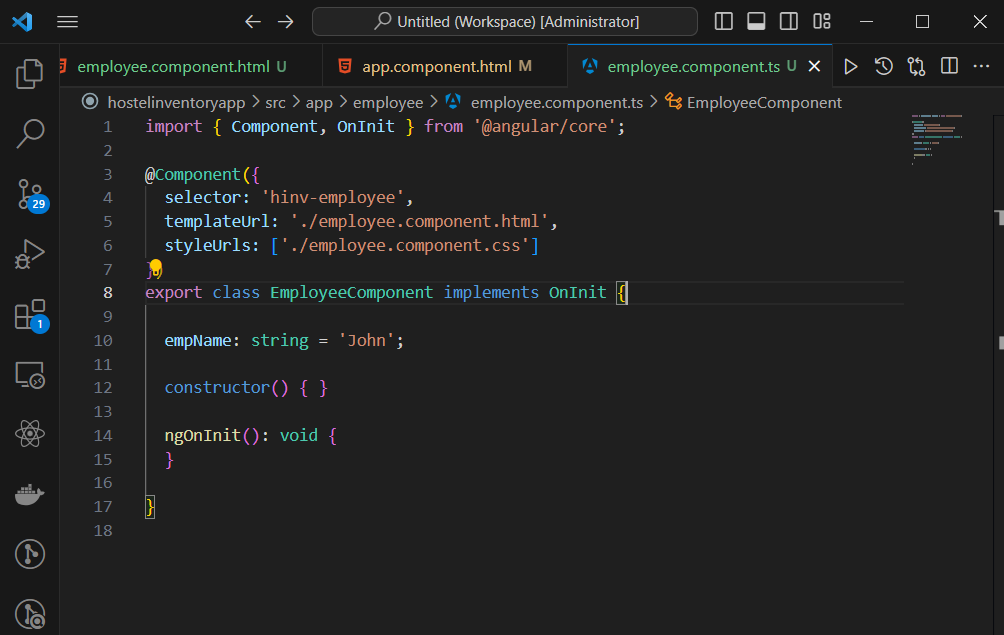
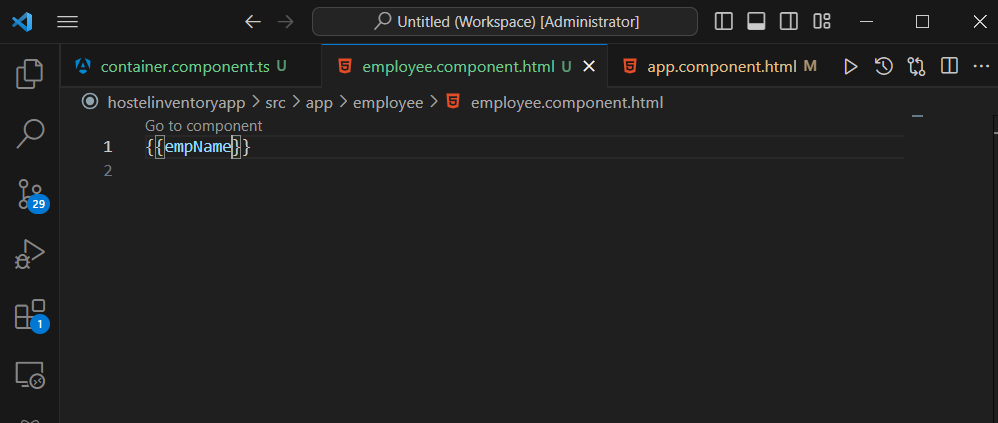
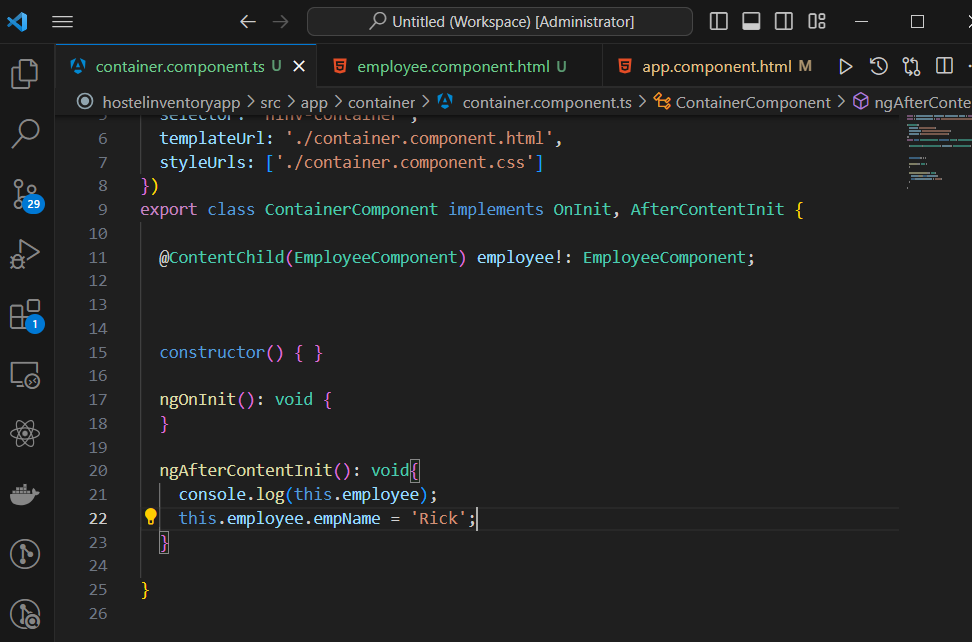
Dirty means nothing has changed in this component.







ng content can help you pass html elements like props

**ngAfterContent**

**Dependecy Injection**

Dependencies are services or objects needed by classes to perform some function.

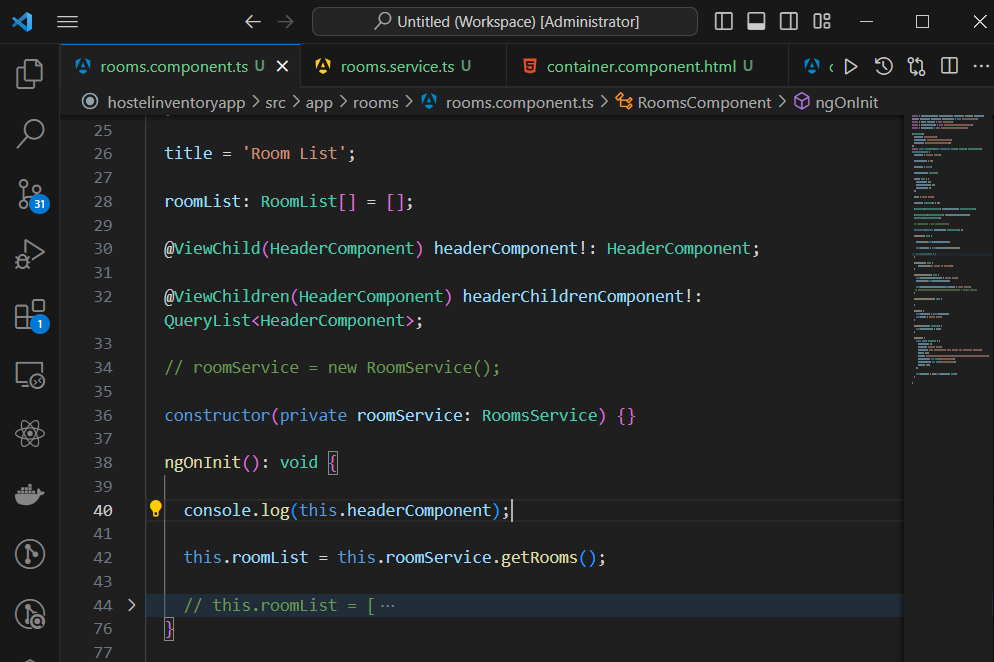
It’s a design pattern.

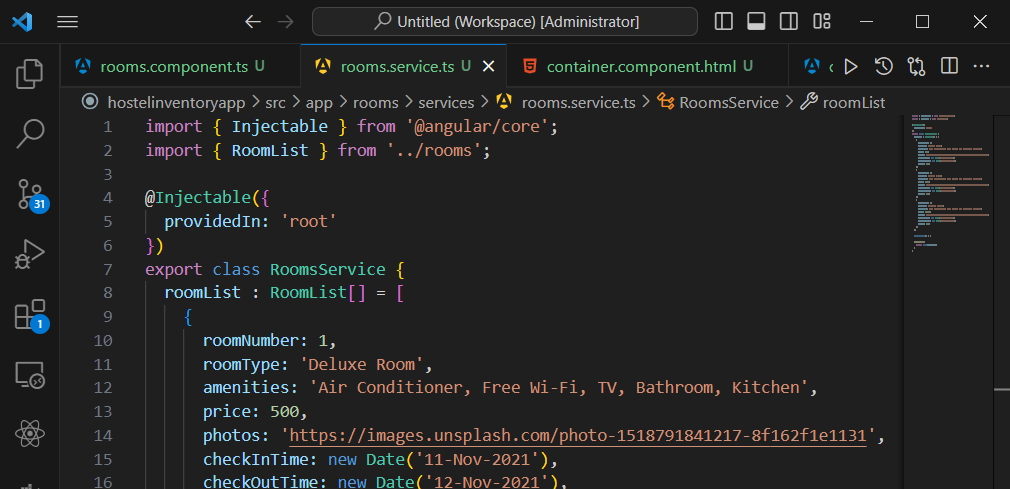
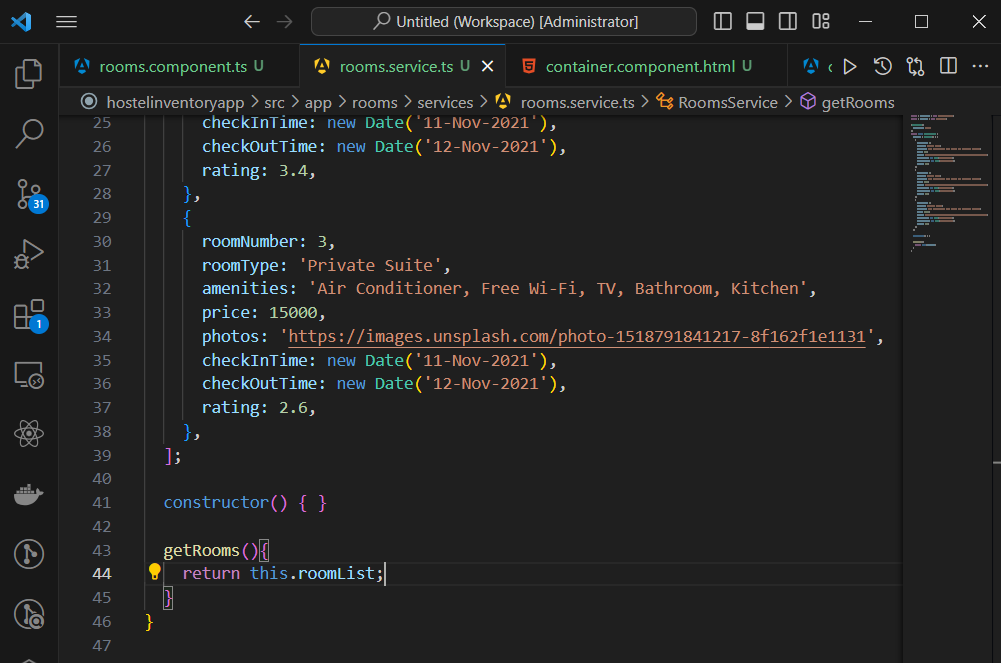
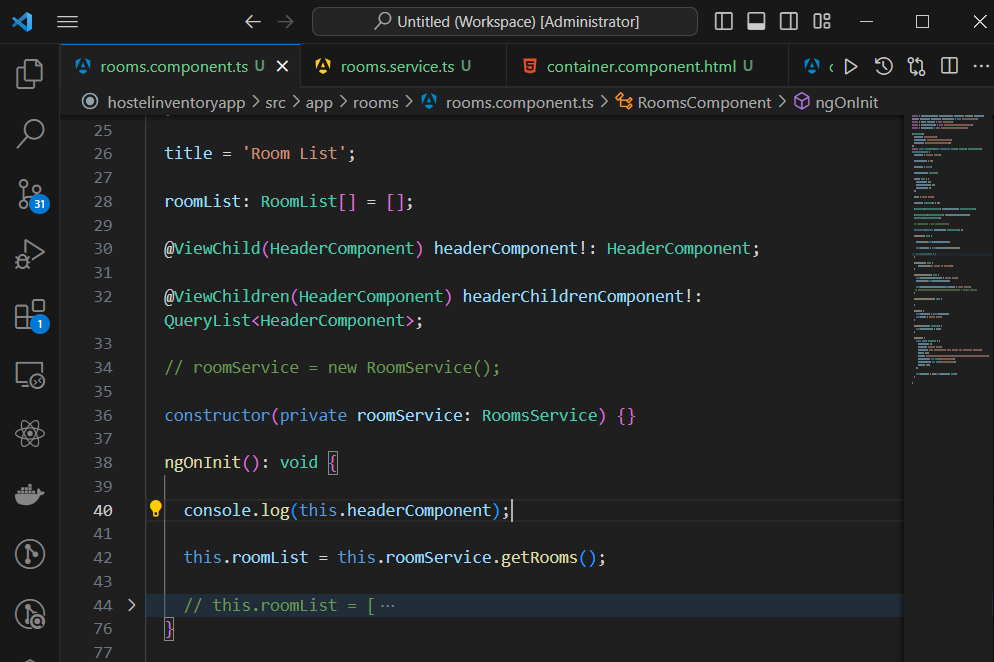
Angular has built-in dependency injection support.

**Creating and consuming services**

Services are used for keeping reusable code. We place our business logic here.

* ng g s <<service\_name>>
* Injecting a service
* Consuming a service

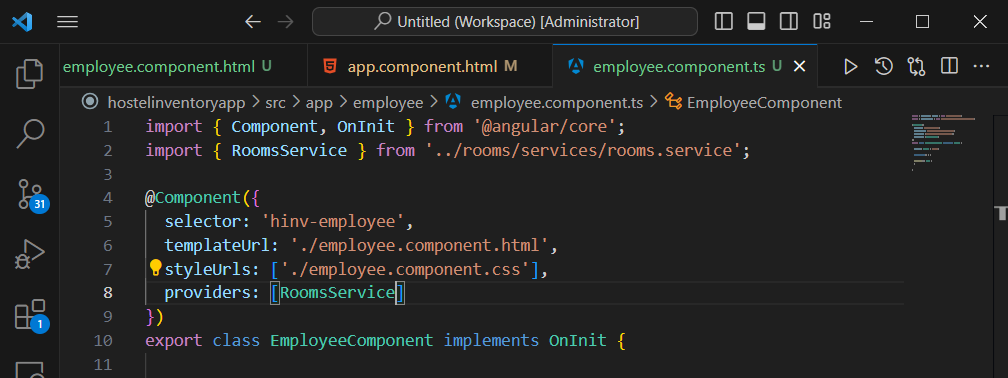


****

The providedIn: ‘root’ in the service is used to register the service automatically. NullInjector error shows up when your service is not registered.

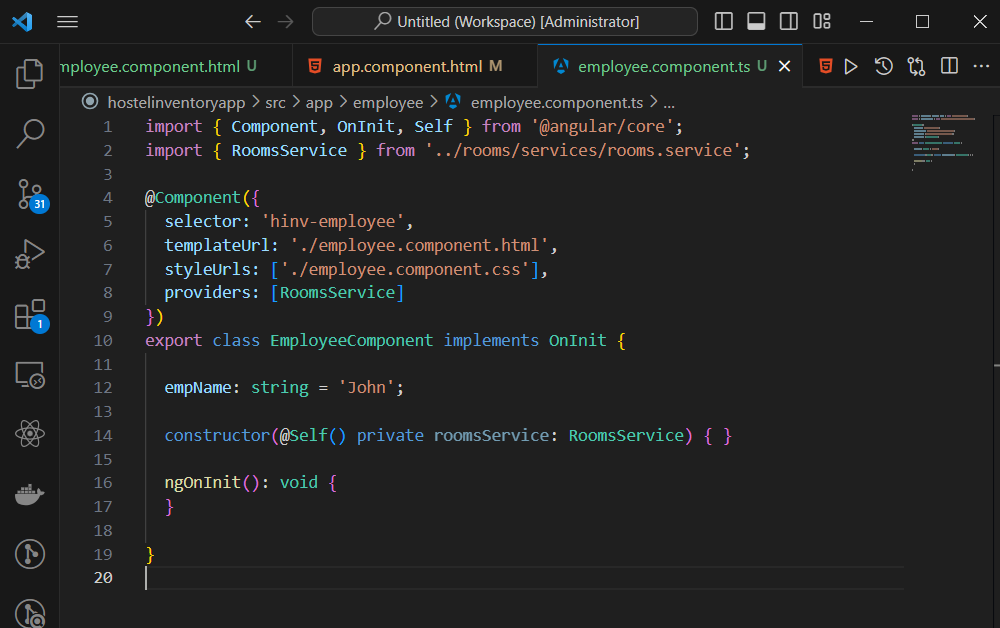
By default, instances created in Angular are singletons.

You can create a local instance of a component using;



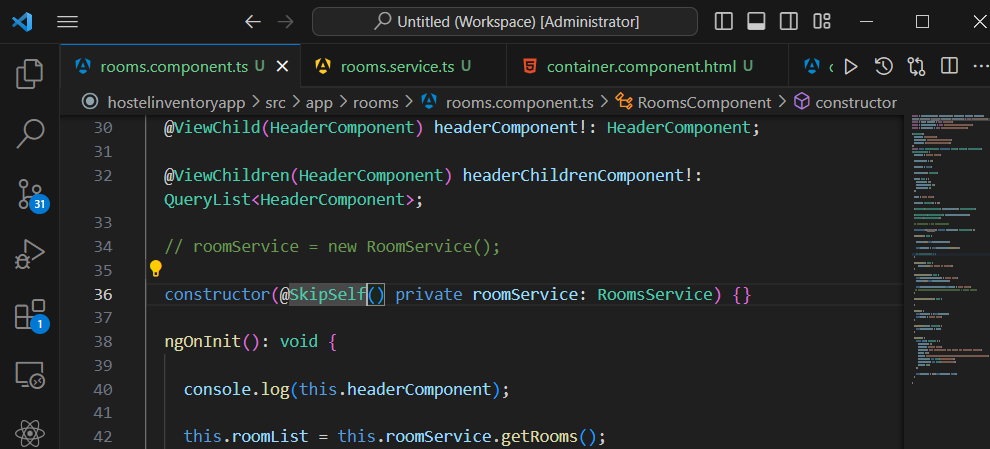
**Resolution Modifiers**

* Self
* SkipSelf
* Optional
* Host

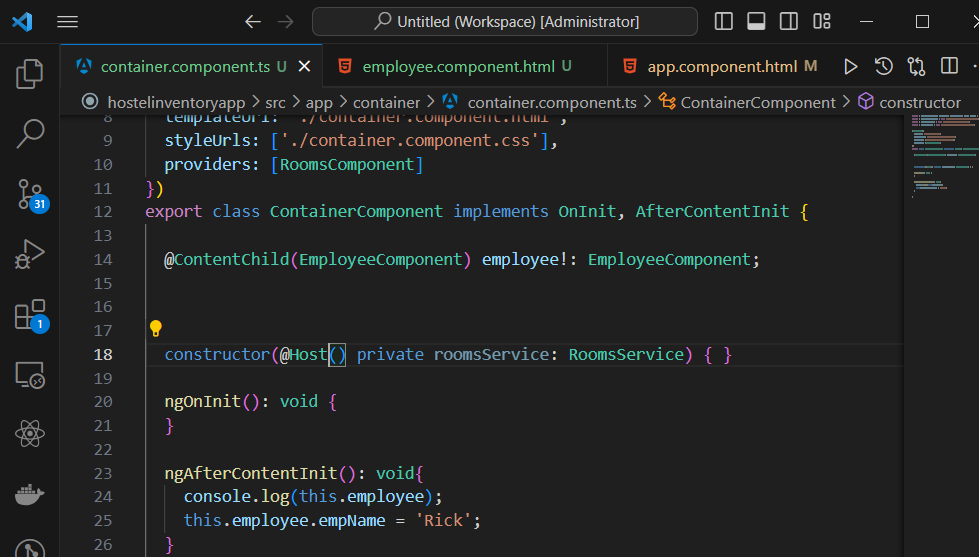


It means that this service should be available at this level otherwise it should throw an error.

@SkipSelf means skip myself within the entire resolution tree.

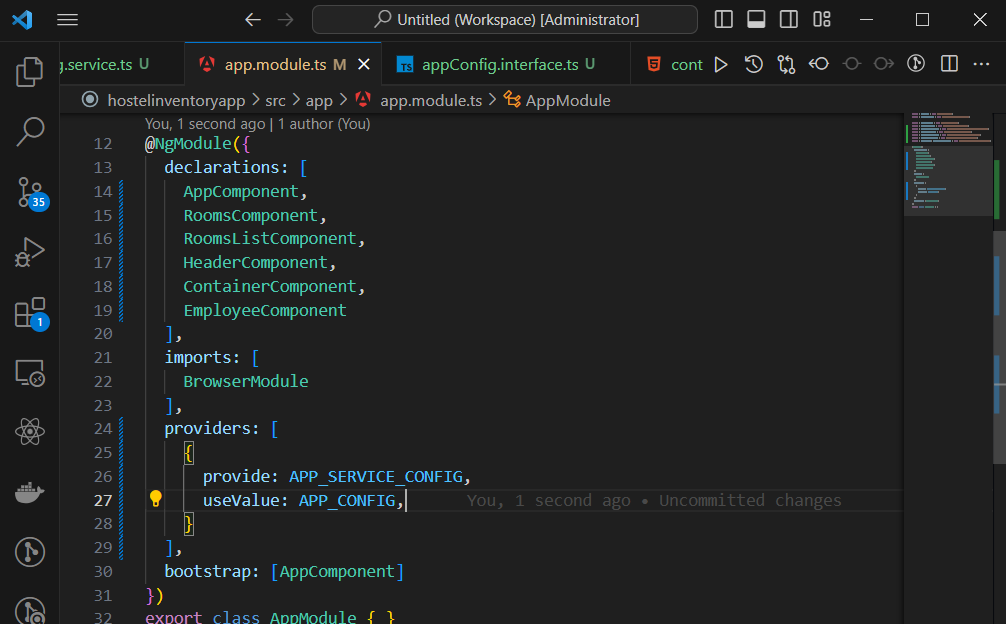


@Host means start looking for this service at this level.



**Dependency Injection Providers**

**NOTE:** Whenever you change something in any environment file, make sure it’s adjusted in all environment files

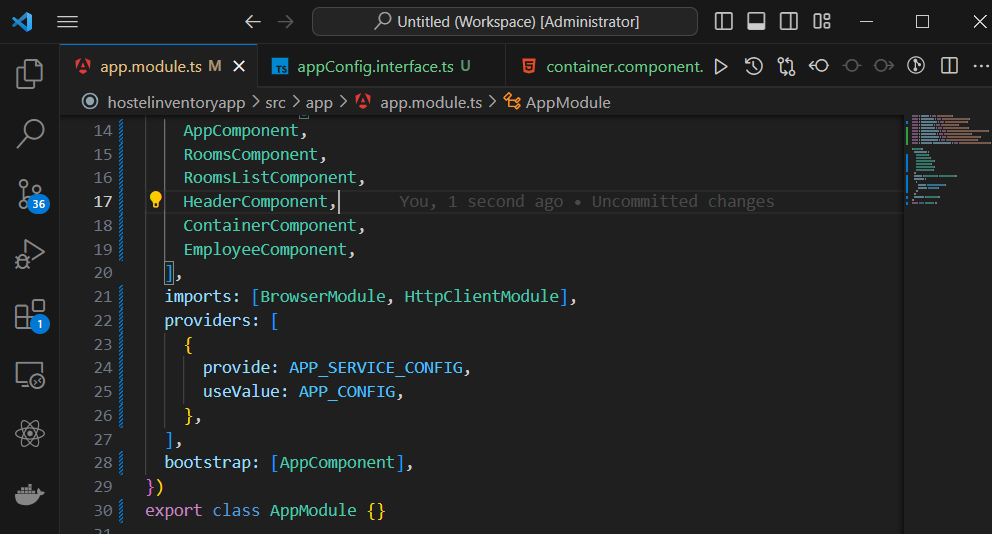
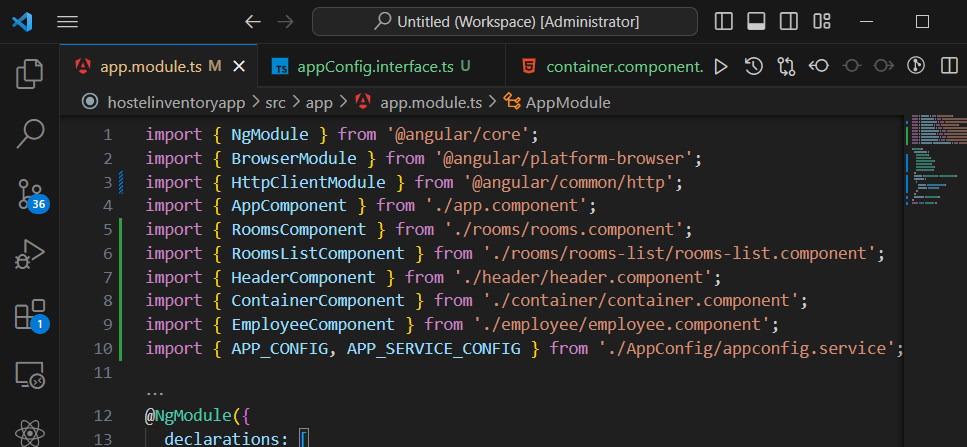
You register a service in the app.module.ts file

**Angular Http and Observables**

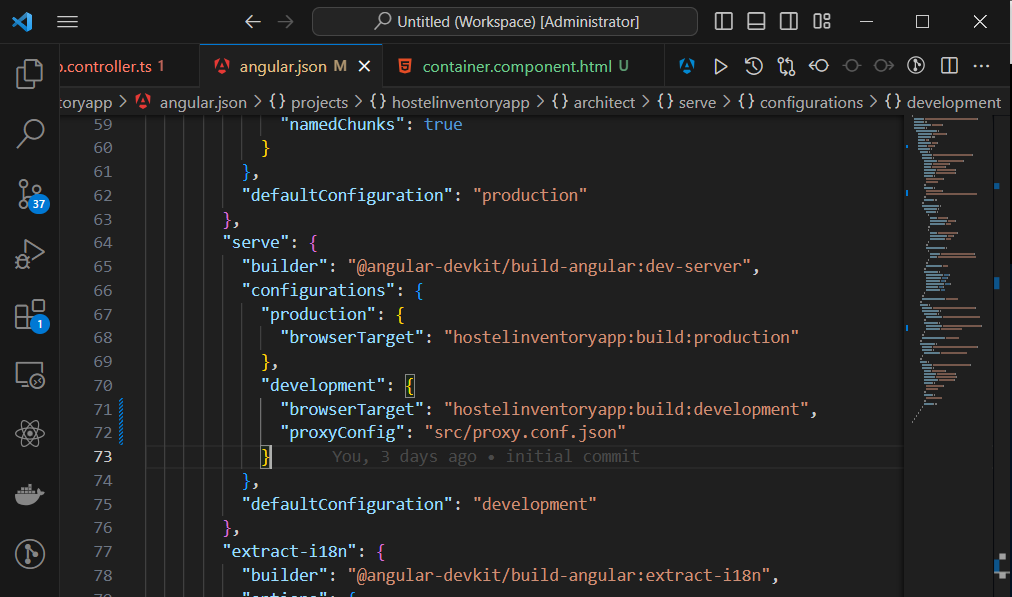
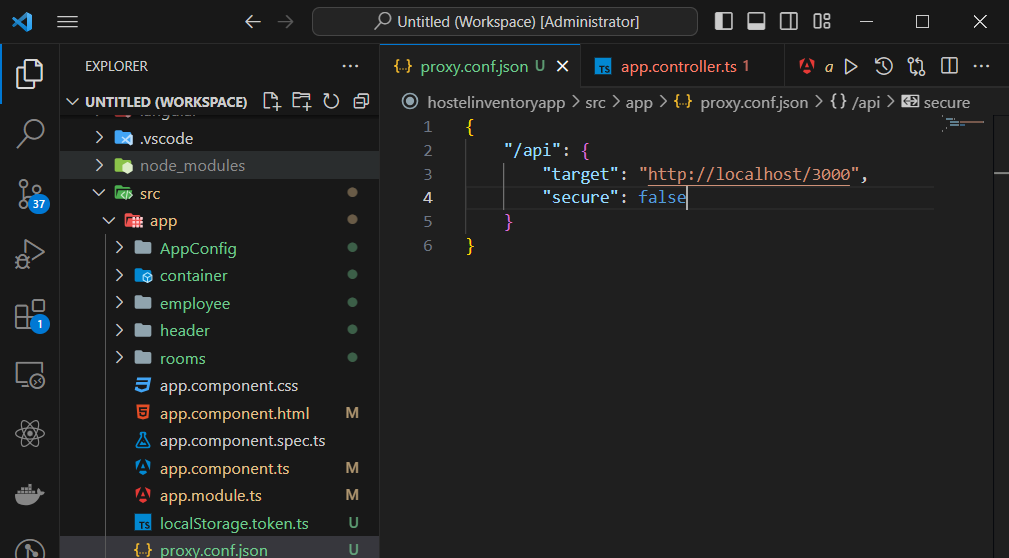
HttpClient is the service provided by Angular to interact with APIs

HttpClient internally uses RxJs.

* We need to import HttpClientModule
* You need to explicitly import it from @angular/common/http



Don’t forget to add your proxy.conf.json file and also modify your server settings in the angular.json file

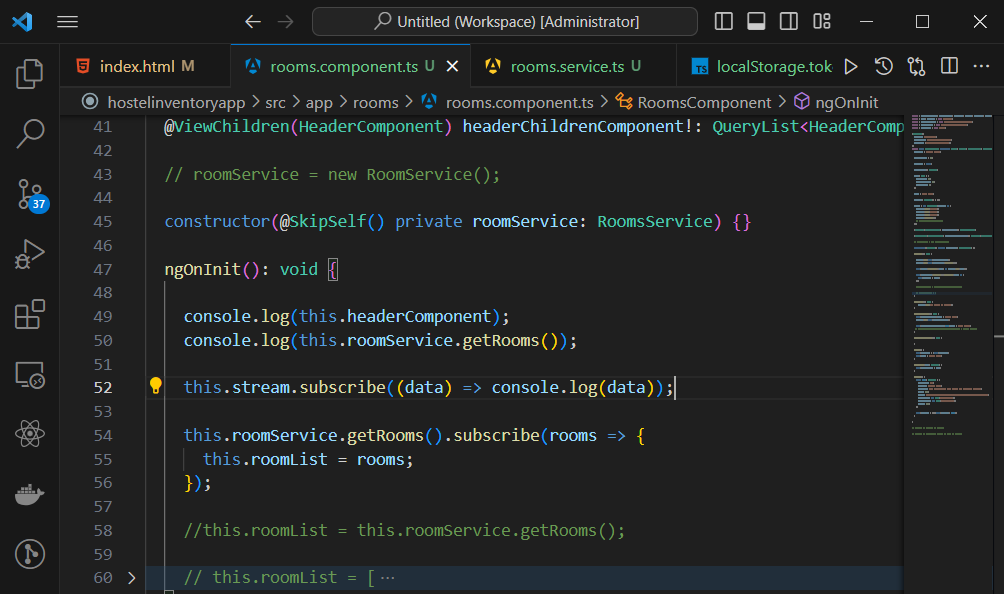
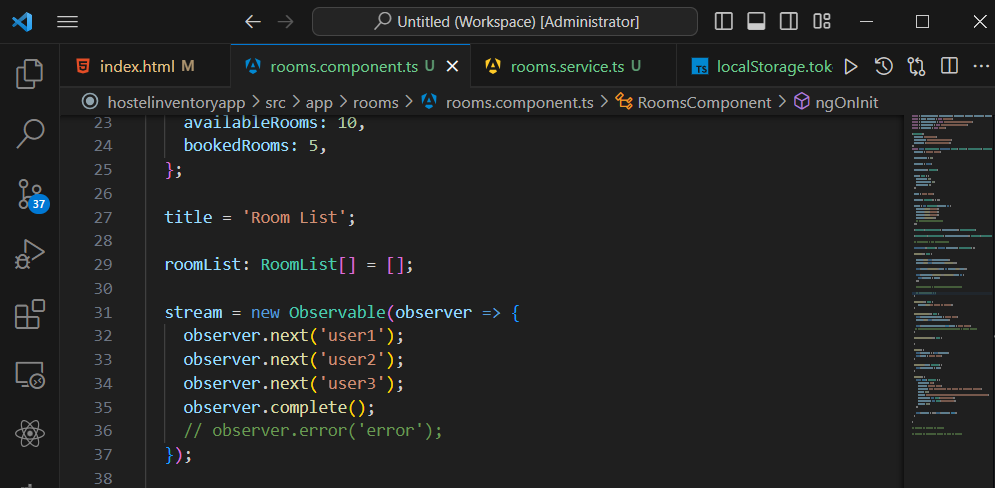


**RxJs and Observables**

RxJs is a library for writing Reactive Programming.

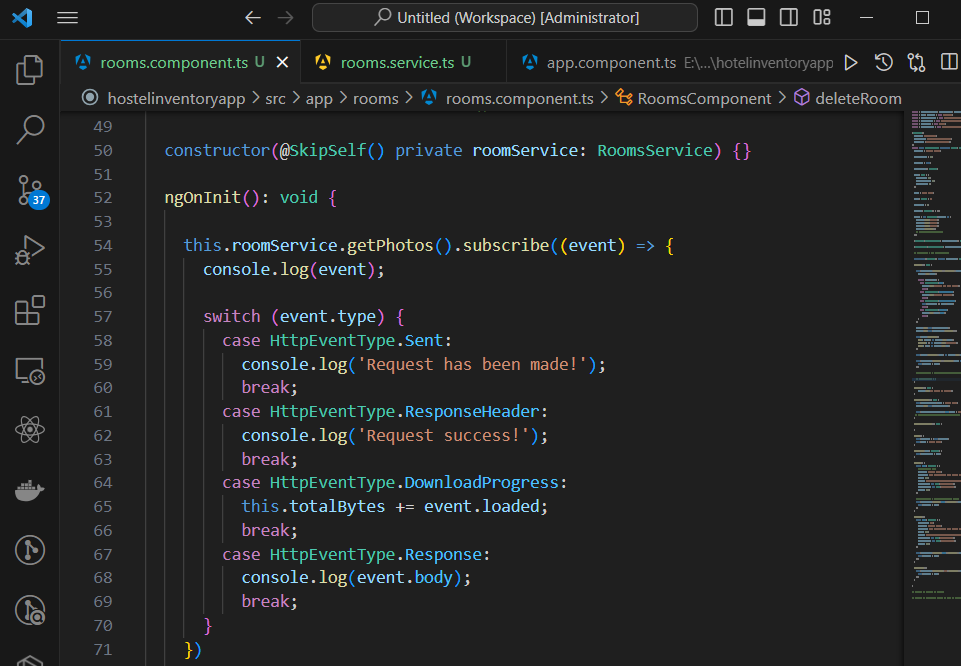
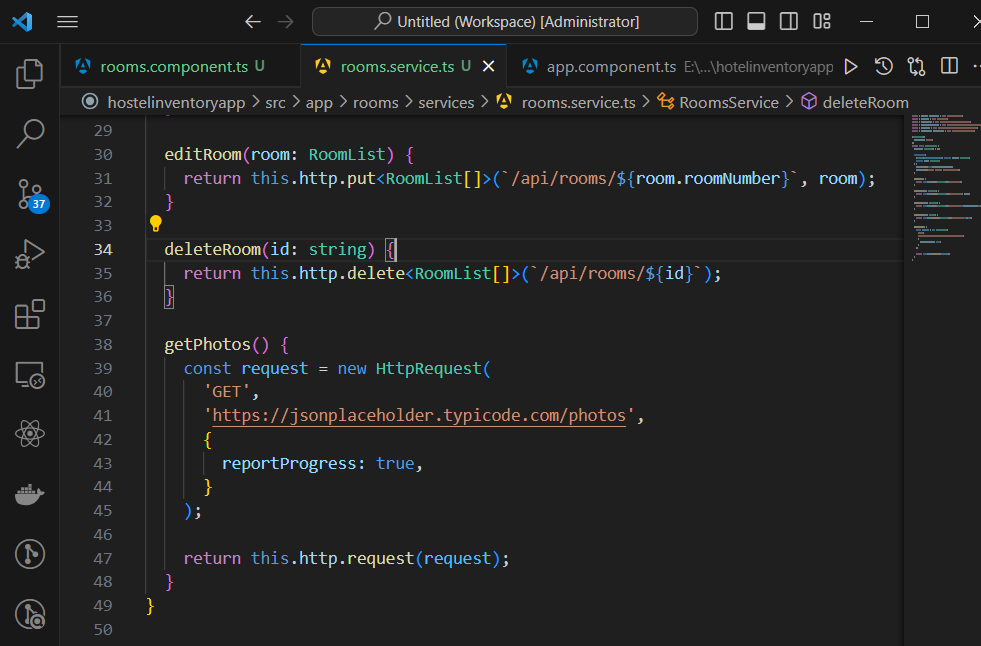
Observables are stream of data to which we can subscribe.

Subscribing to streams



**NOTE:** The api request methods are in the room service

You can utilize the HttpRequest method which offers much more functionality.

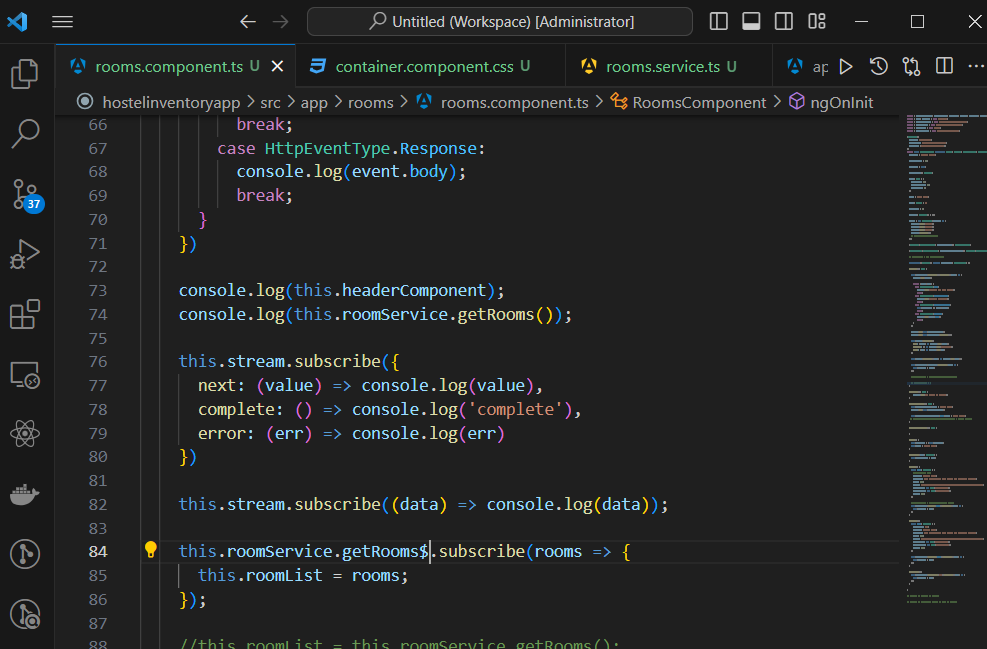
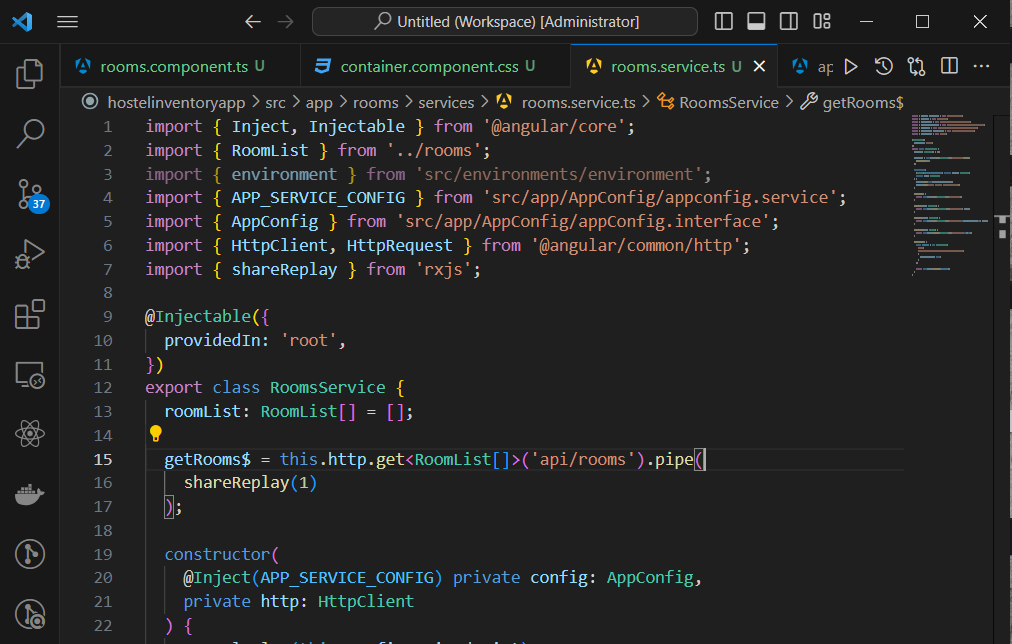


**RxJs operators**

* ShareReplay
* CatchError
* Map operators

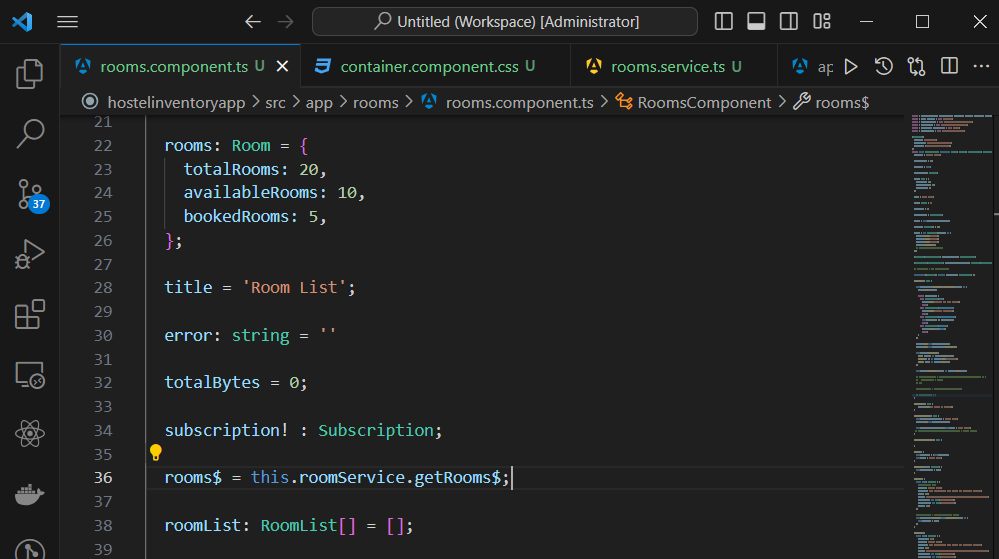
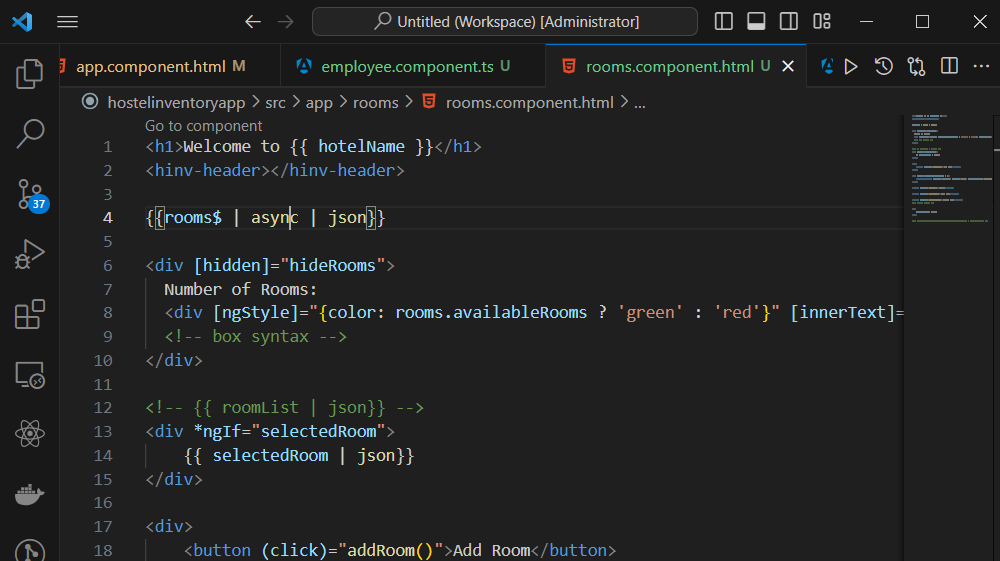
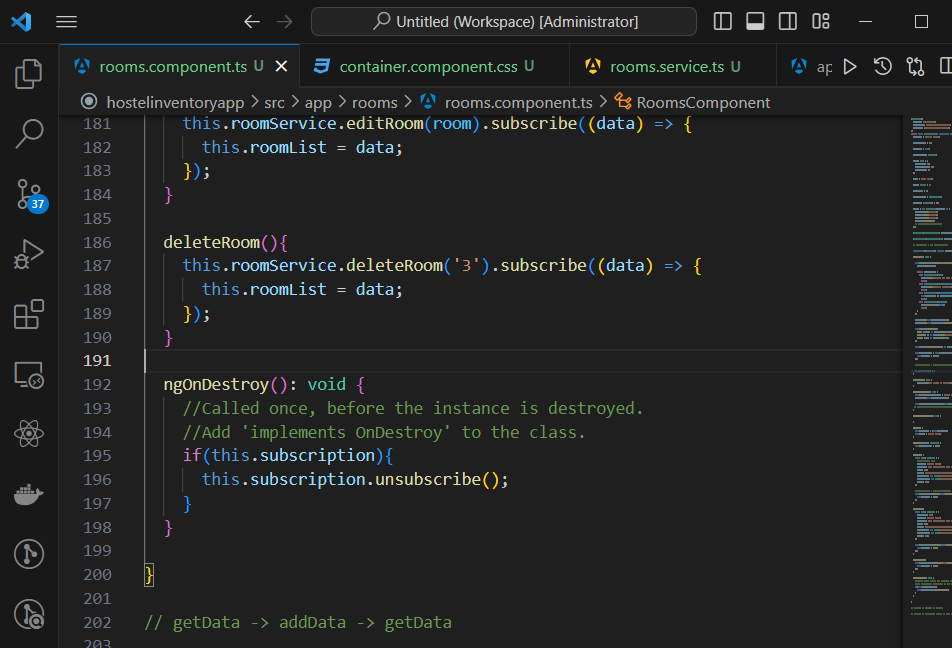
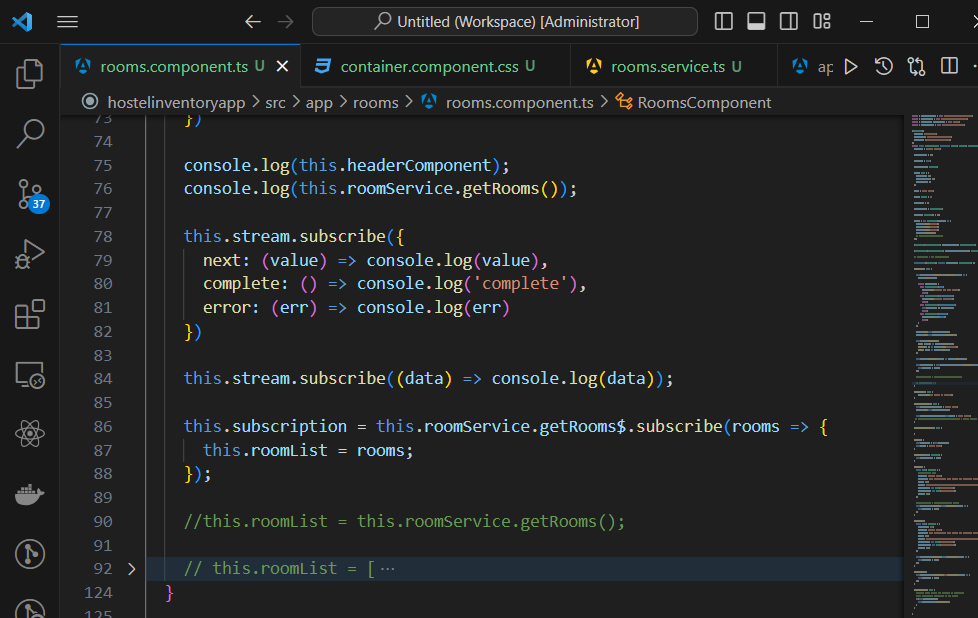
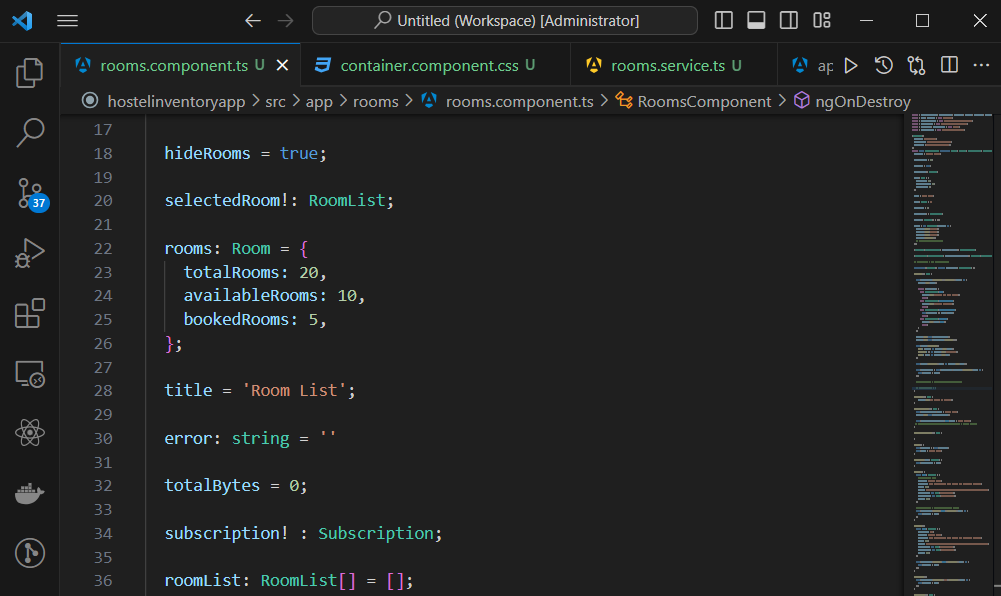
getRoom$.... the $ sign is used to indicate an attribute that’s a stream

Use shareReplay to cache the first request.

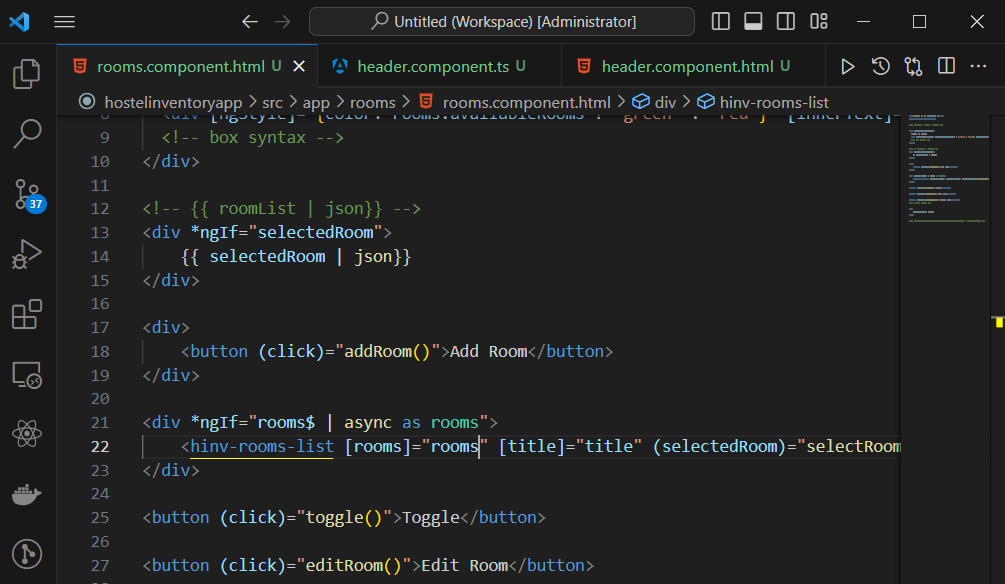


**Async pipe**

Whenever you subscribe to a stream you should unsubscribe to it.



Async pipe unwraps the retrieved data for us automatically as well as carries out the an onDestroy for us



**Http Interceptors and APP\_INITIALIZER**

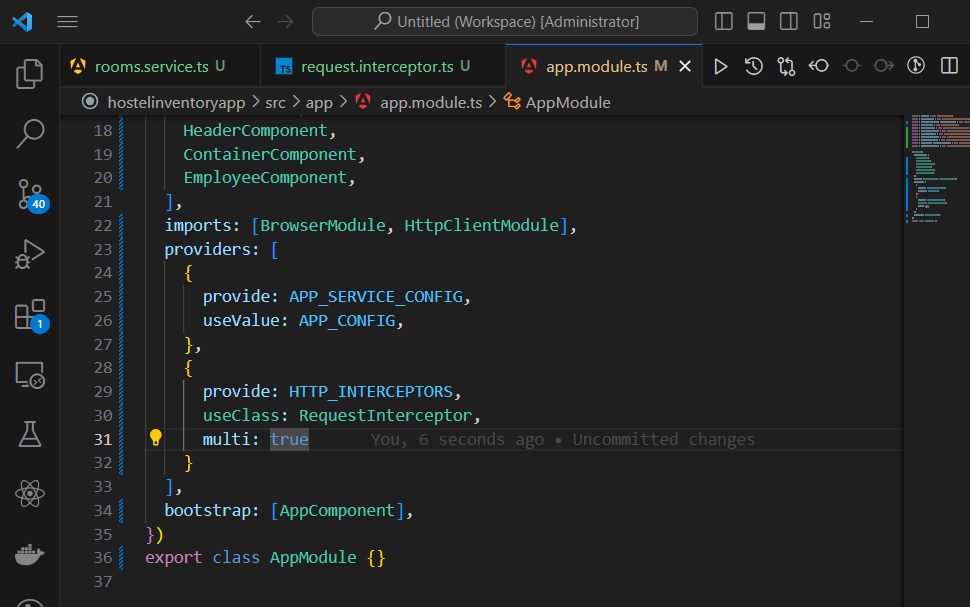
* Allow us to intercept request
* We can change data and add headers.
* APP\_INITIALIZER allows you to inject function as application startup.

4, 12

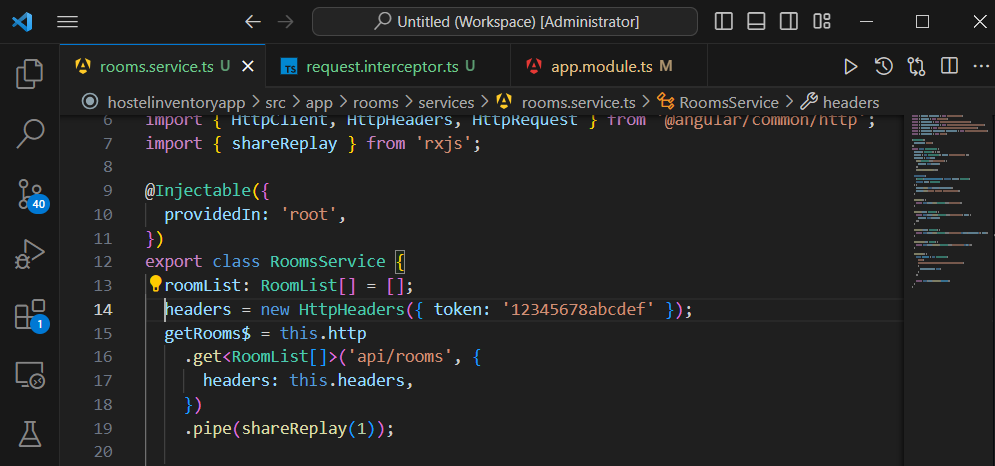
To generate an interceptor type in the command line;

* ng g interceptor request

And make sure to register it in the app.module.ts

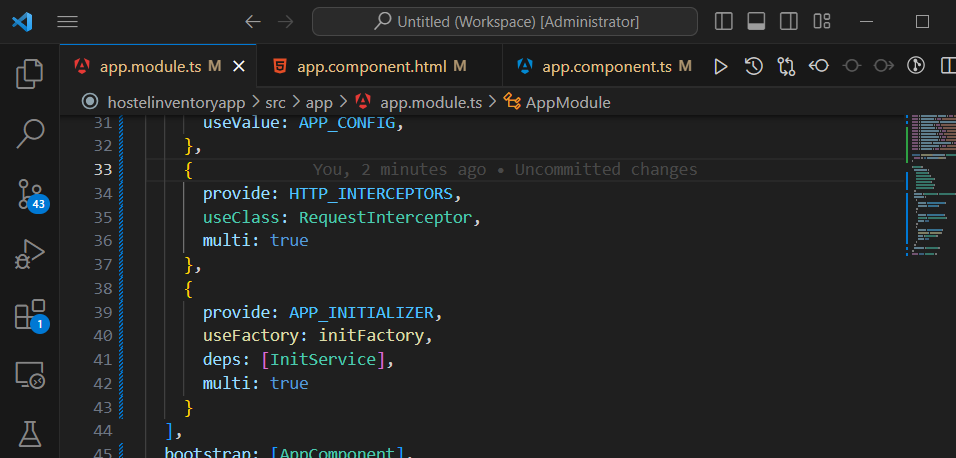


Adding headers to requests



With interceptors, you don’t modify the original request. You have to clone it then modify it

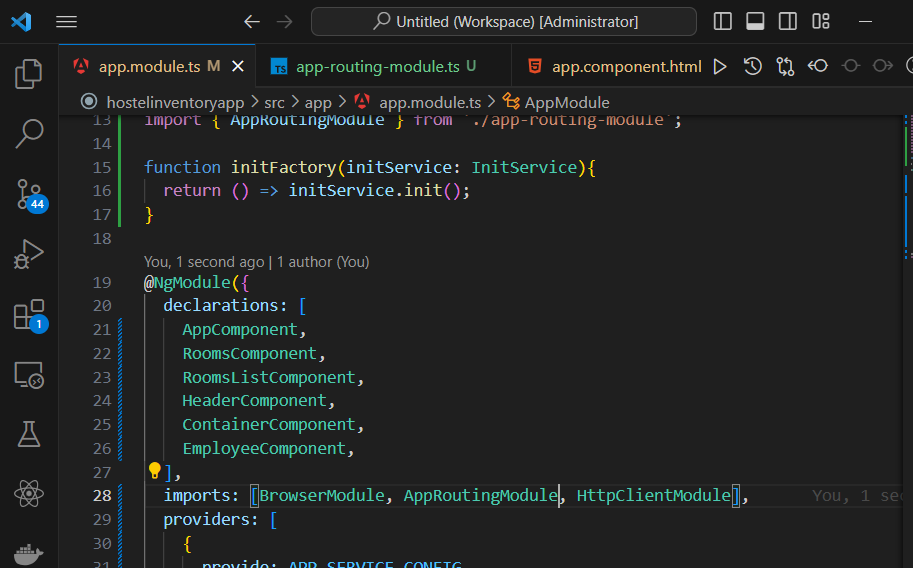
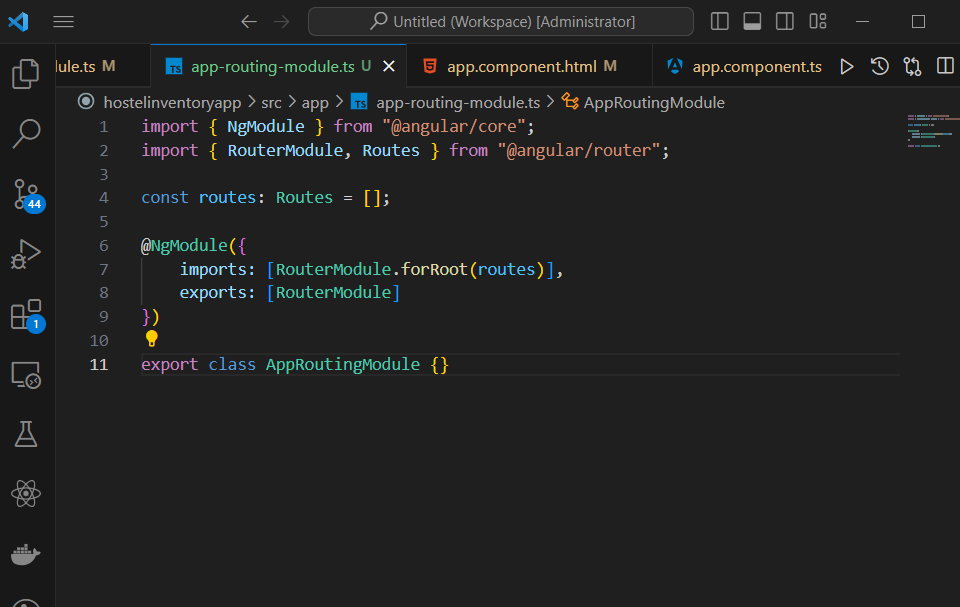
Registering an App Initializer



**Angular Router**

* Provides the functionality to add Routes
* Provides SPA functionality
* You can configure nested routes.

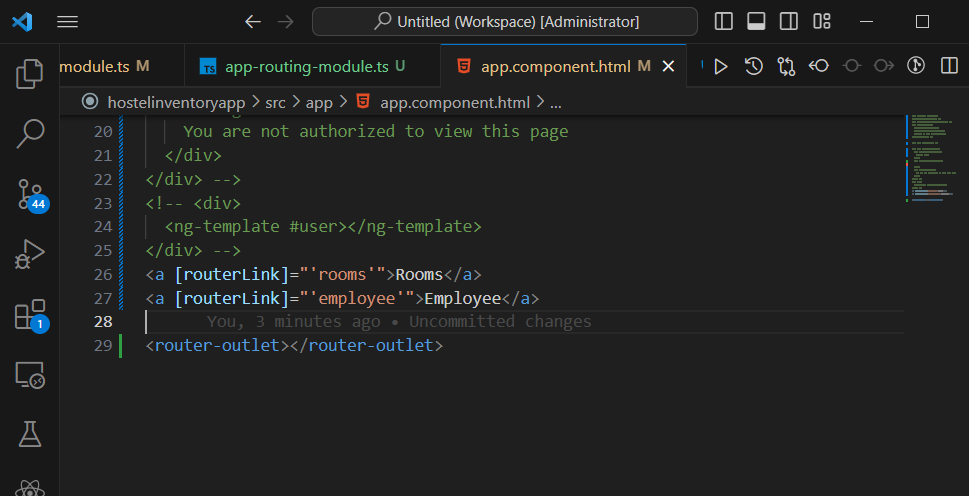
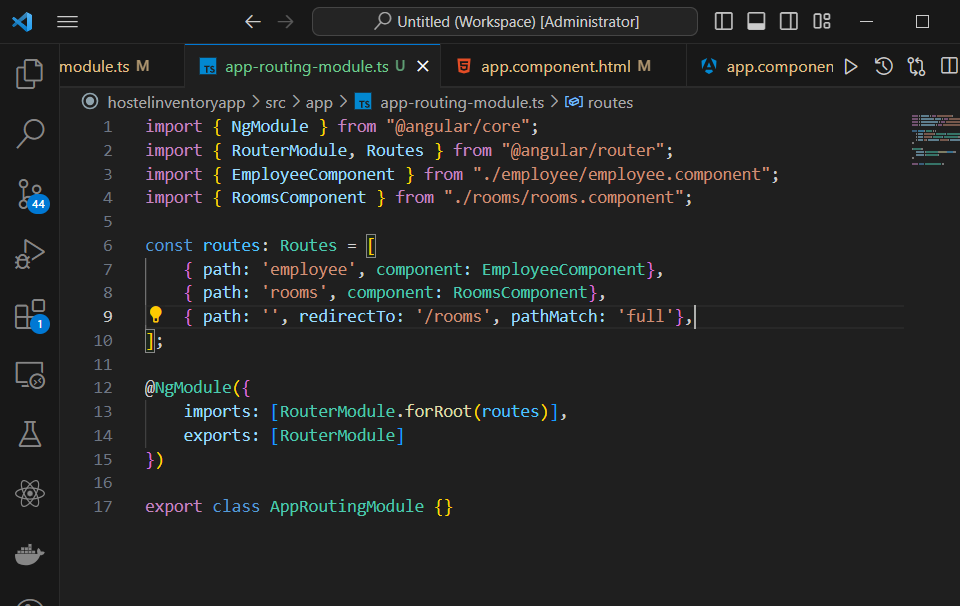
To use it:

* Import Router module

Note:

* only configure forRoot once.
* Using href attribute in SPA will make them MPA.

When you configure routing, you have to place the router outlet. Angular will add the component you navigate to just after the router outlet.



Generating wildcard routes for 404 not found



Dynamic Routes

**Angular Material** 9: 50:00

Go to material.angular.io

* ng add @angular/material
* ng generate @angular/material:navigation app-nav for navigation by material

**Setting up template driven forms**

* Add FormsModule to the import section of app.module.ts

**NOTE: 10:43:00 –** Two-way data binding i.e. connecting backend to frontend

* Pristine means fresh state of a form and dirty means the form has been interacted with.

Custom Directive with Forms

* ng g d hover
* A use case is to change behavior of DOM on hover.
* Use HostListener to listen to events.

They are similar to components but don’t use template code.

@HostListener is used to attach listeners to your DOM.

SCAM – Single Component and Module i.e. for every component you should have a module.

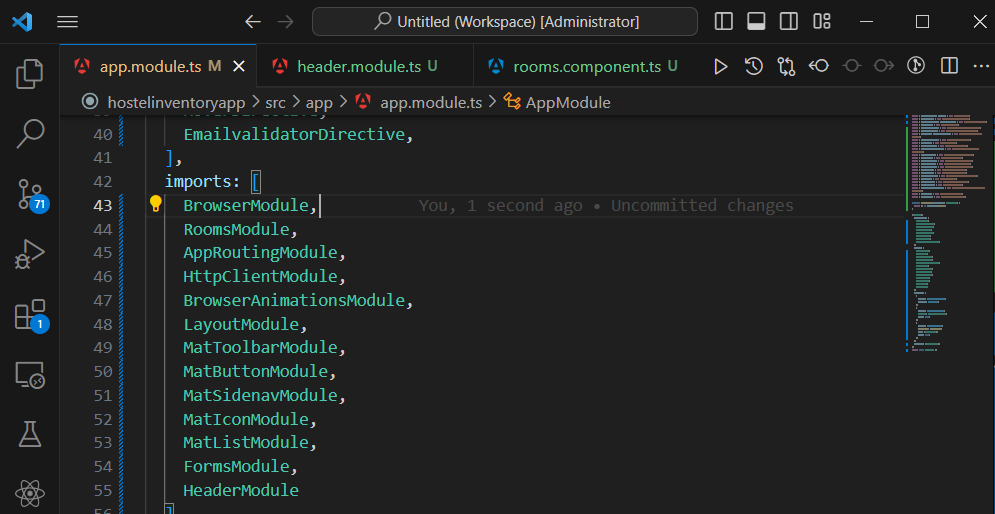
**Generating feature modules**

ng g m rooms --flat=true

ng g m rooms --routing --flat=true

When you’re going to add a module somewhere always read the error, make sure to export it if it’s being used in other modules and always add the necessary modules it uses

**NOTE: Always register your feature modules before AppRoutingModule**



**Note: To build the project**

* **Run ng build -c=production**

The main.js file is the one that is served to your user.

**Lazy loading 12:25:00**

A lazy module should not should not know about any other module it should just stick to its own properties. It’s used to help prevent the scenario of giving an entire chunk of data to the user to download.

ng g m booking --route=booking --routing --module=app a command to make it lazy loaded.

**Make sure the booking component exists**

In case you’re creating new code to first create the module then the component:

* ng g m search
* ng g c search

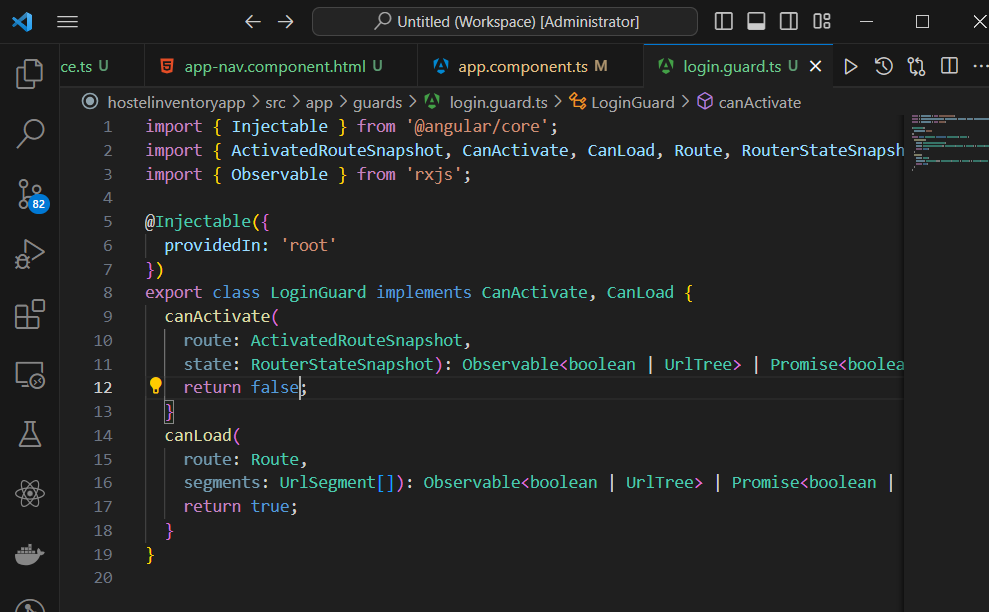
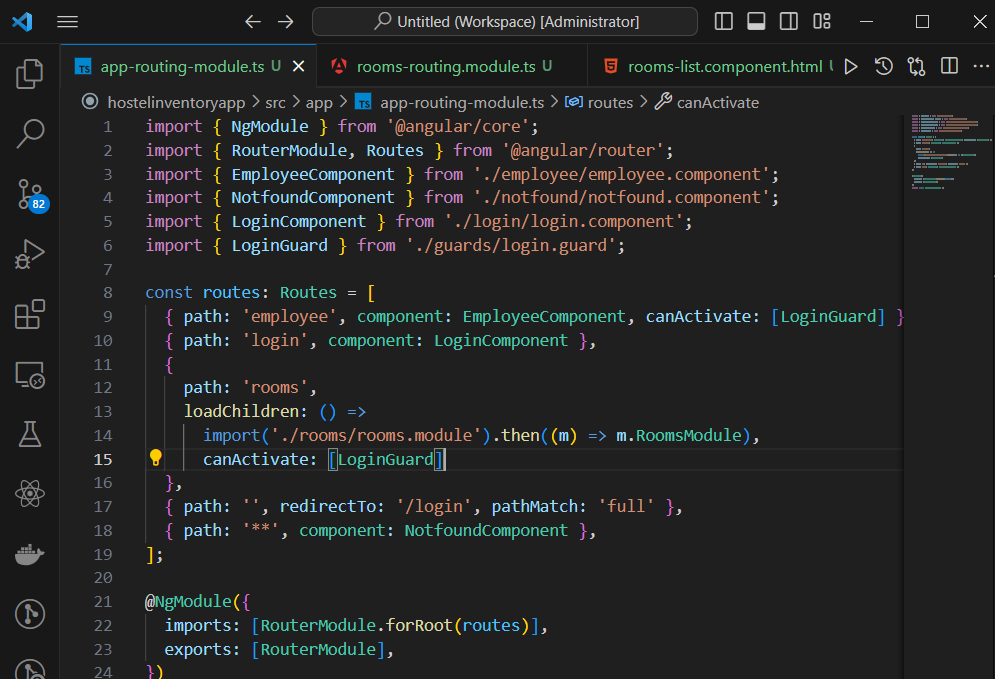
**Route Guards**

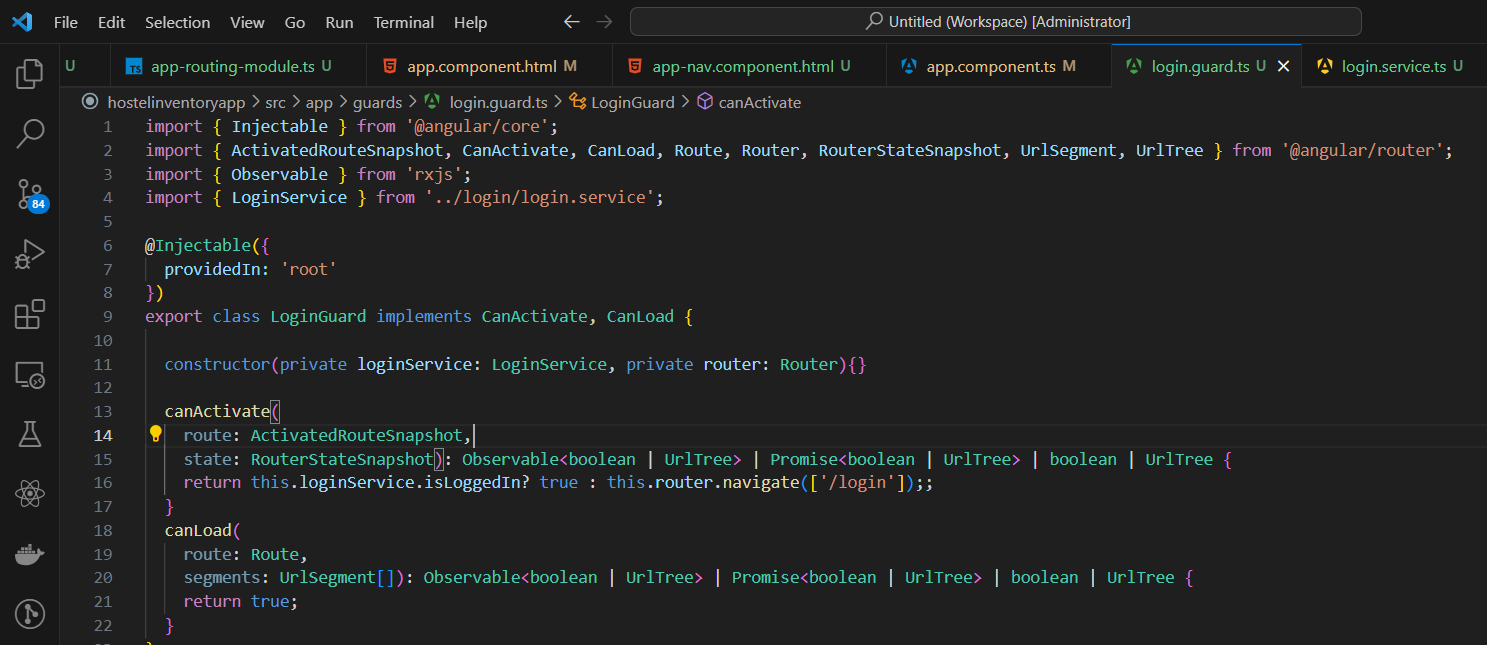
Include:

* caActivate
* canActivateChild
* canDeactivat
* canLoad
* Resolve

To generate routeGuards type the command.

* ng g g login





**Reactive Forms**

* Import the ReactiveForms module
* 13:40
* 14: 30

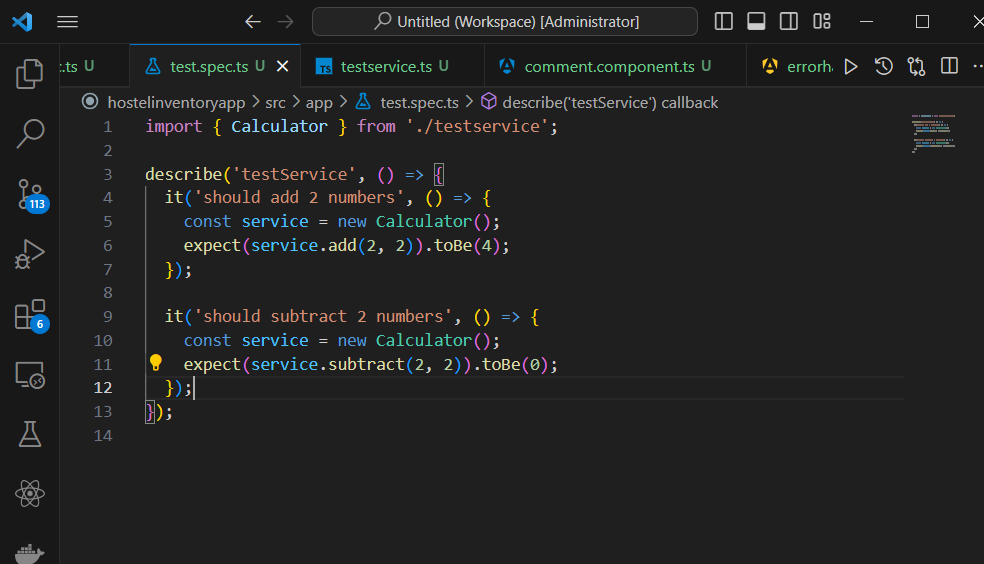
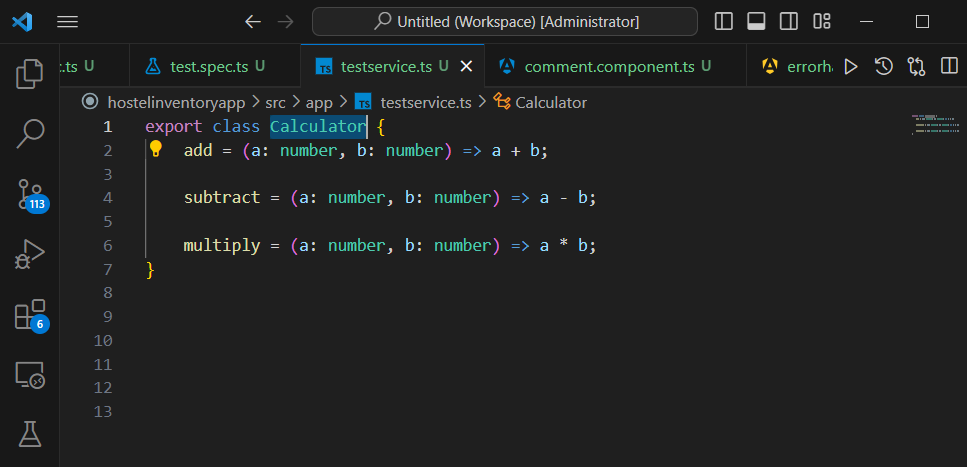
To route to the bookings, go to localhost:4200/booking/1

* Don’t do filtering and sorting at your front end… do it at your API level
* To create a pipe type ng g p <<pipe\_name>>

**NOTE:** Resolve guard is related to data prefetching

**Testing in Angular** -43

* Angular uses karma plus jasmine to write the test cases
* ng test is the basic command



* ng test –code-coverage = used to see how much of your code is being tested. You don’t have to test 100% of your application. You should focus on the critical code / logic that needs to be tested.

**NOTE: Learn more about github actions**