**Topics to Cover in Today’s Session**

1. CanActivate and CanDeactivate
2. Observable
3. HttpClientModule and HttpClient
4. Performing CRUD operation

Today we will look at how we can make sure the routes we create in Angular are accessed by the right people and that we prevent unauthorized access to routes that are private

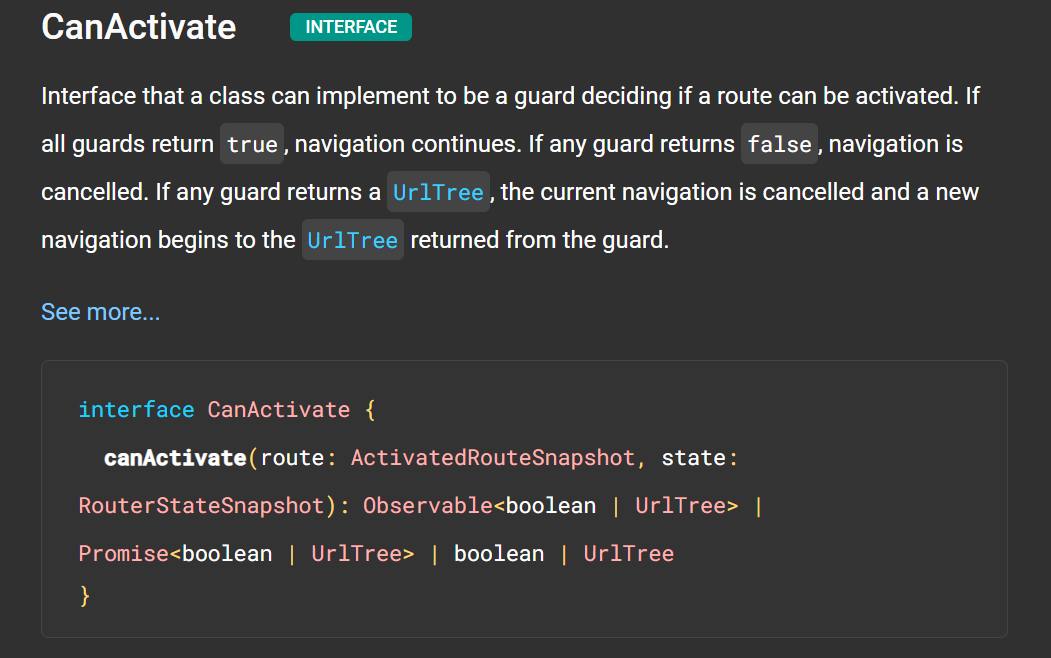
**What Are Route Guards?**

Angular route guards are interfaces provided by Angular which, when implemented, allow us to control the accessibility of a route based on conditions provided in class implementation of that interface.

Guard can return either a boolean or Observable<boolean> or Promise<boolean> which resolve to a boolean at same point of time.

Here are some types of Angular guards: CanActivate, CanActivateChild, CanLoad, CanDeactivate and Resolve.

CanActivate Route: 🡪 **Check whether user can visit.**



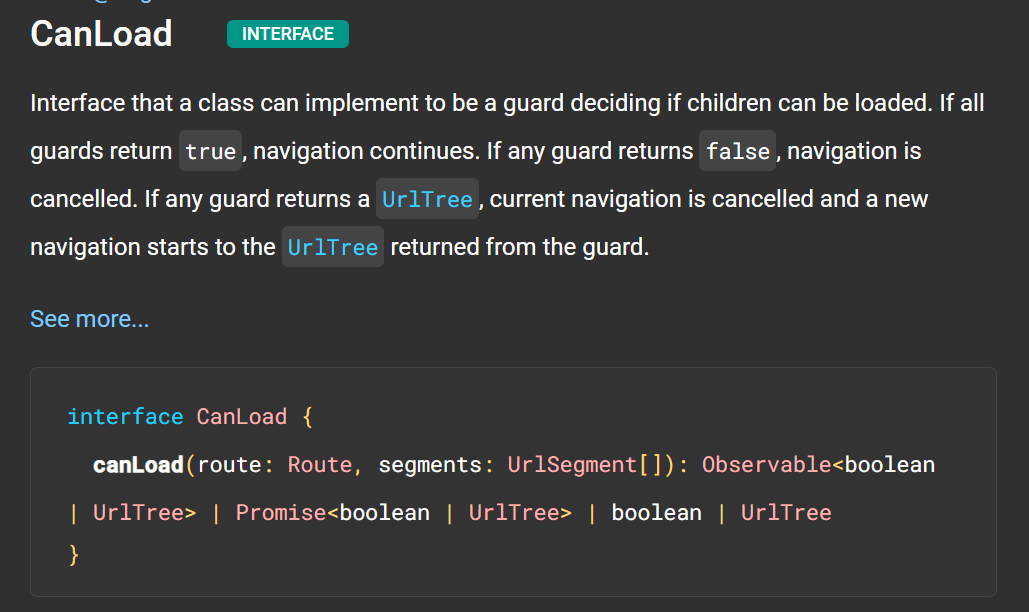
For detain Description visit: <https://angular.io/api/router/CanActivate>.

CanActivateChild:🡪 Check whether user can access children of routes.



For detail Description: <https://angular.io/api/router/CanActivateChild>

CanLoad: 🡪 Use to check whether user can access routes that lazy loaded



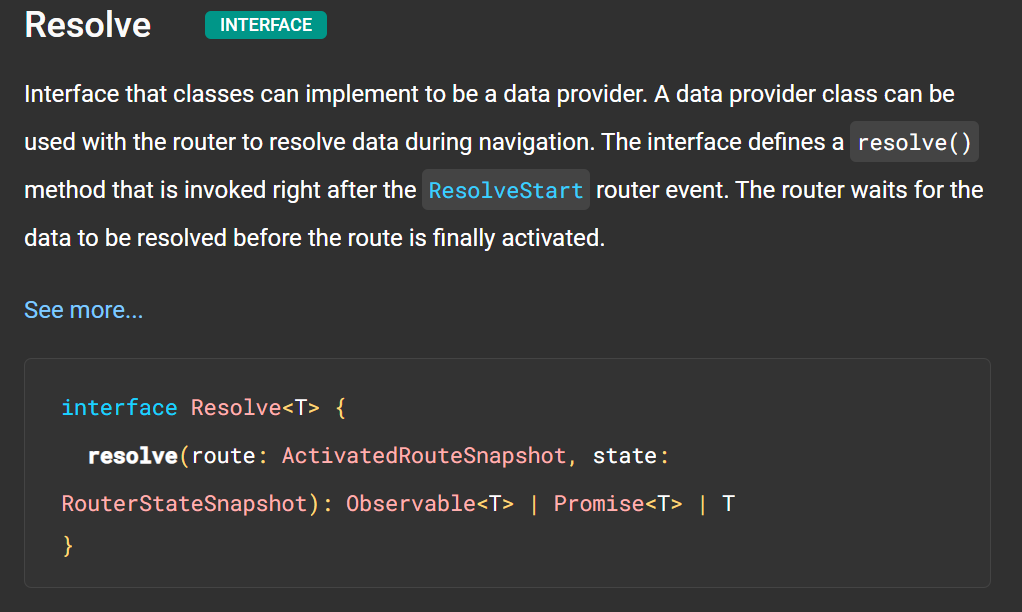
For detail description: <https://angular.io/api/router/CanLoad>

CanDeactivate:🡪 Checks whether user can exit a route



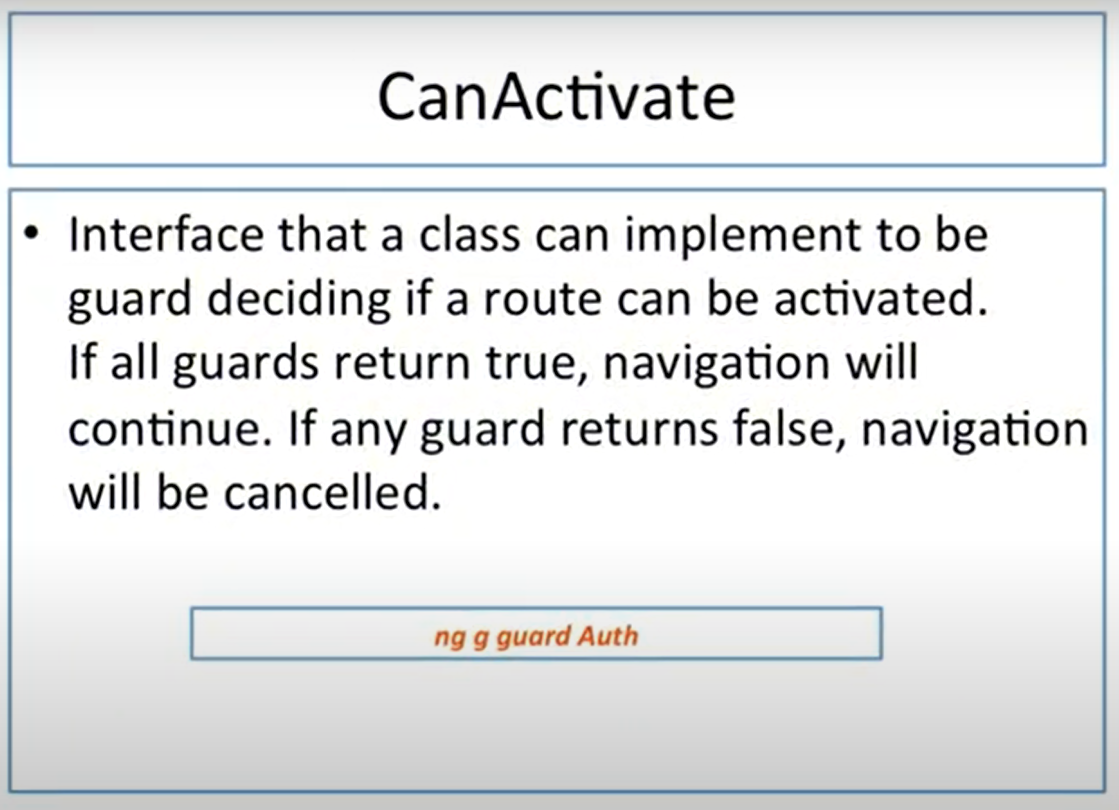
For detail description: <https://angular.io/api/router/CanDeactivate#description>

Resolve:🡪 Performs route data retrieval before route activation.

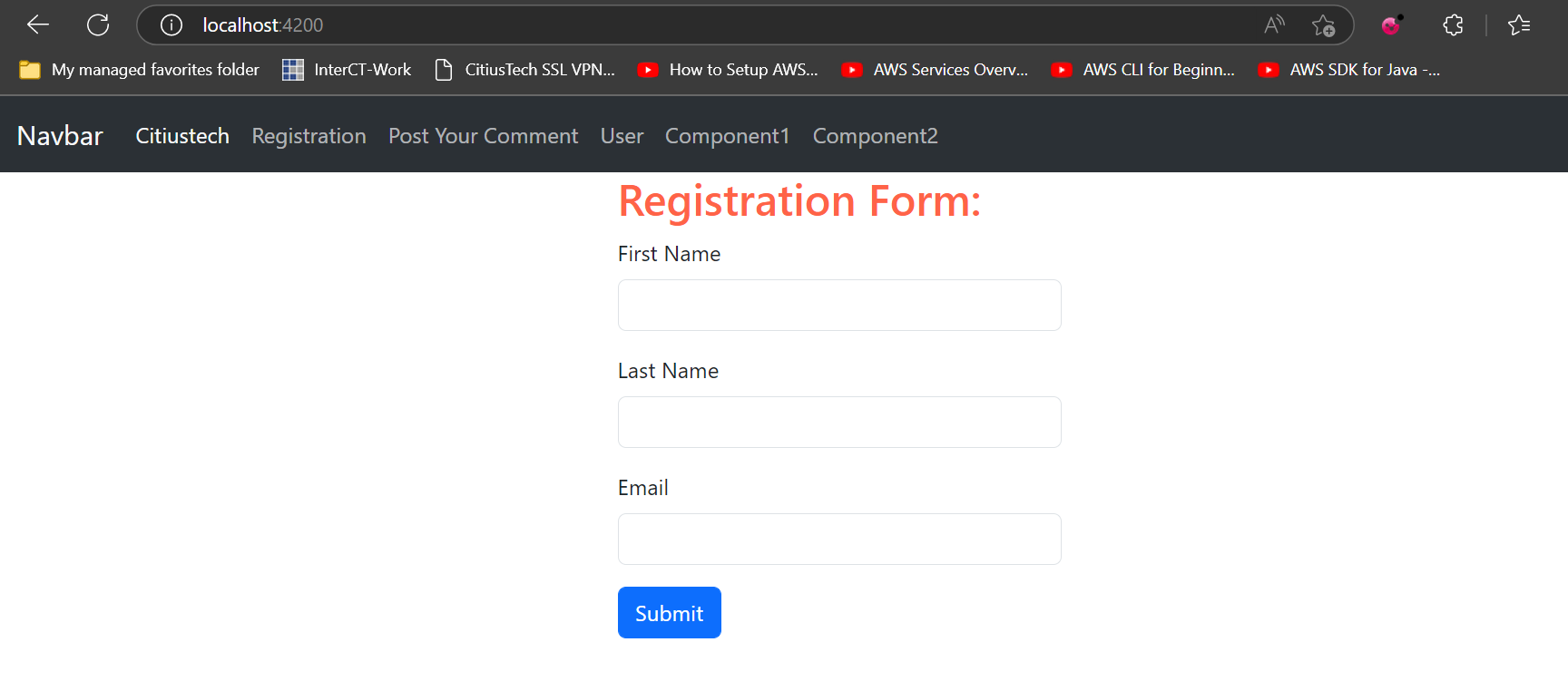


For detail description: <https://angular.io/api/router/Resolve>

CanActivate:



Here we will use same project which we used in last project.



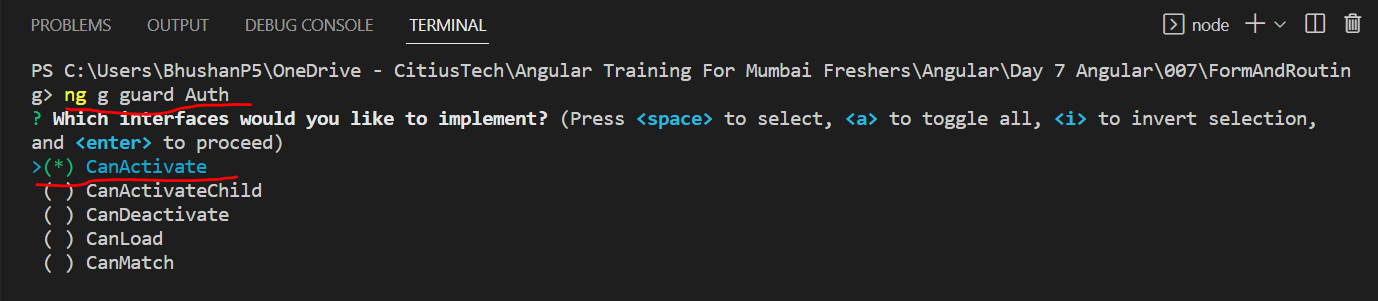
Our goal in this demo is, we can access User component directly but now we will have to provide restriction. Only valid users can only access this component.

So, to achieve our goal, we have to follow given below steps.

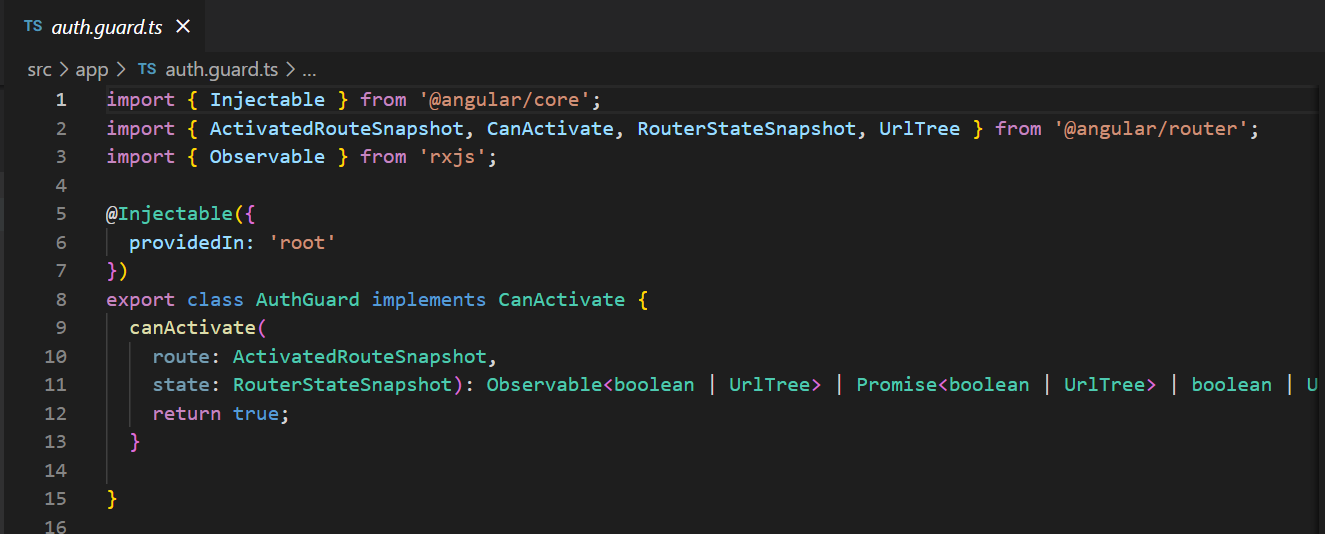
Switch back to VS Code and start implementing our code.

**Step 1:** Create guard for our application. To create guard fire command.

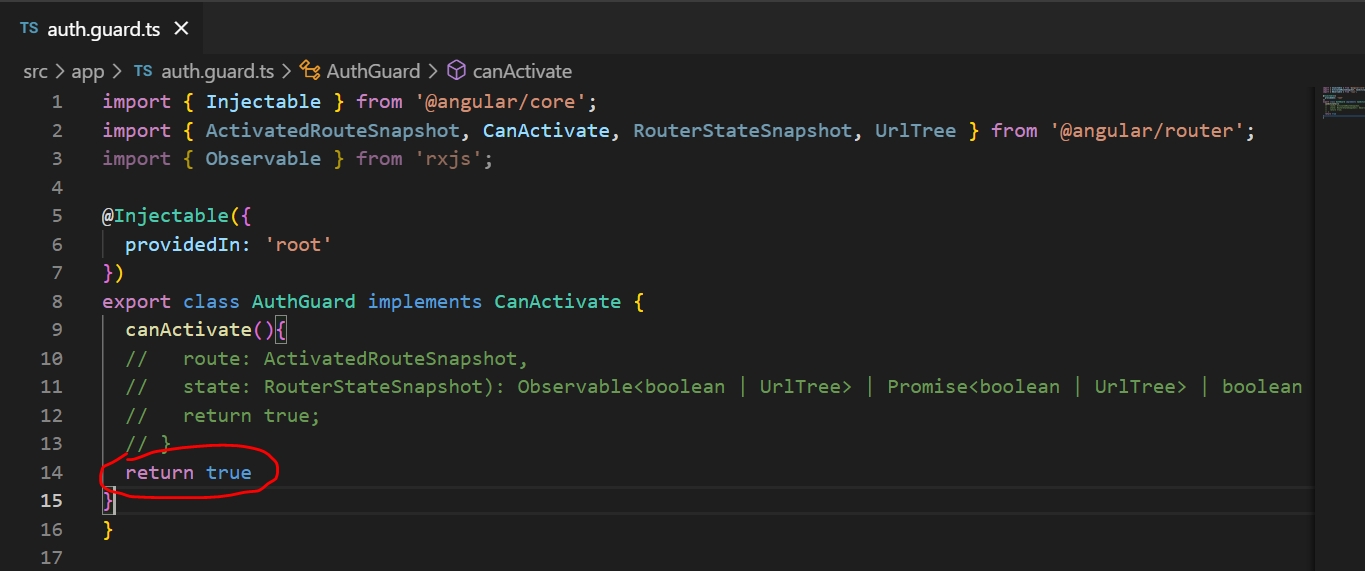
ng g guard Auth



It has created guard by the name Auth



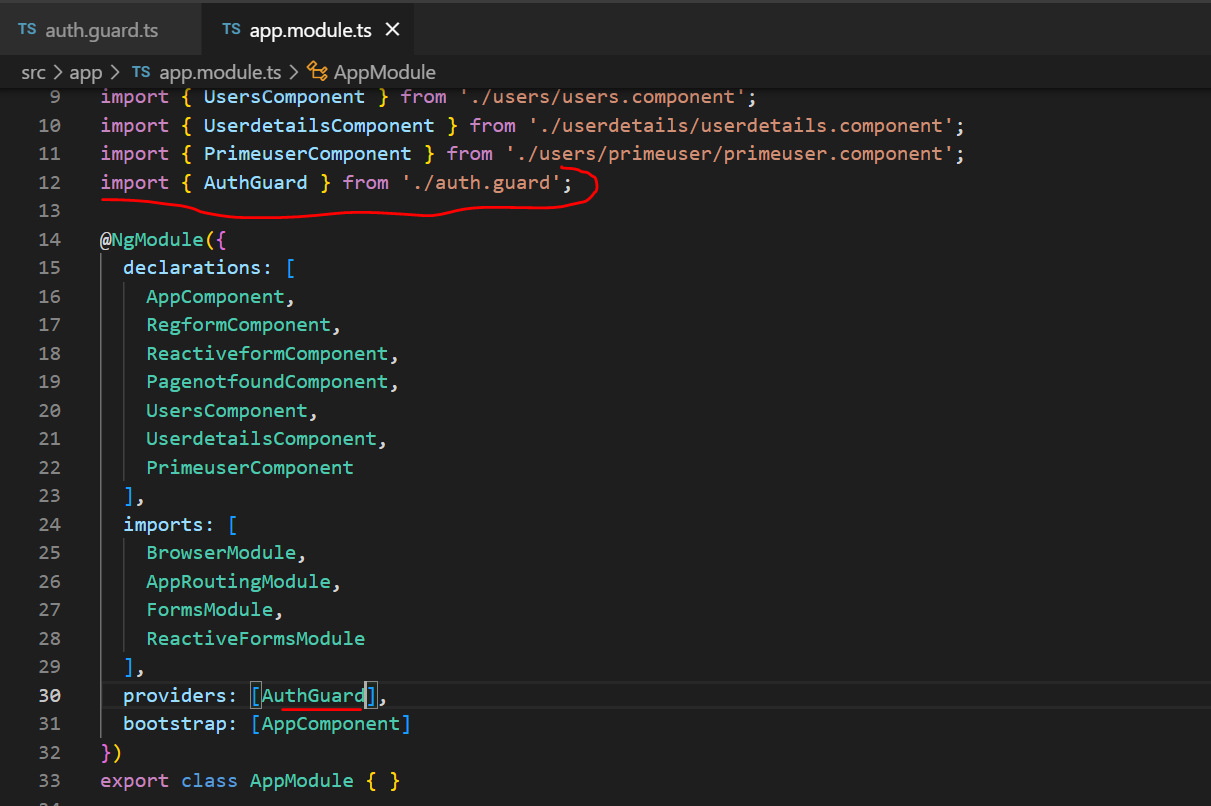
Do one simple change, we will return true statically.



Inside this method we have to define our logic on basis of which it may return true or false. For this purpose we will use service.

Now, move to our app-routing.module.ts file.

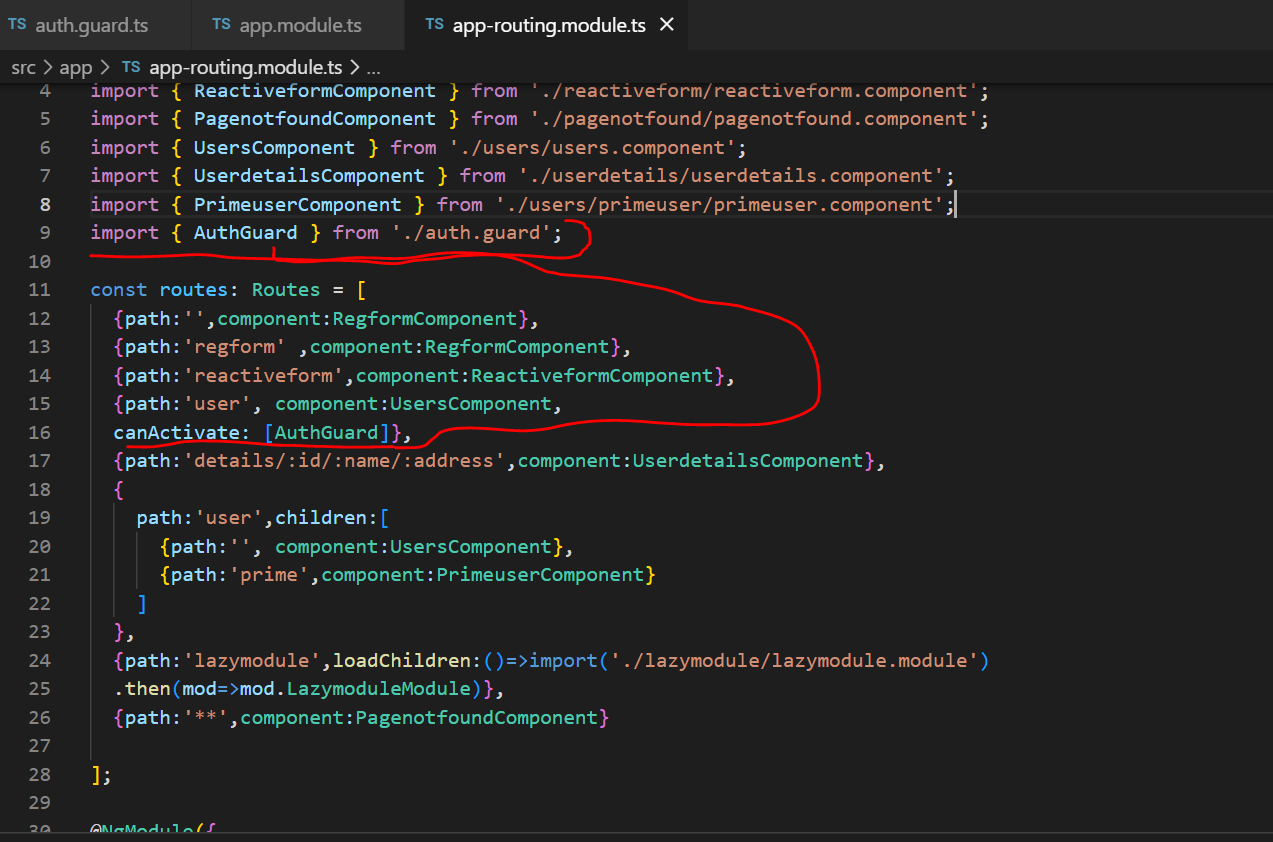
**Step 2:** In app-routing.module.ts file, select the route for which we will have to implement guard. So, in our case we will use user component. If we will open our auth file then it looks like service. It has @Injectable decorator so, like service we will have to import this guard in app.module.ts and add in providers array.



**Step 3:** Next step is, add this guard on route. So, for achieving this we will open app-routing.module.ts file. Here we have to protect our user route. So, whichever guard we have to activate for restriction we have to call our guard which we created.

So, here for user route I have added one more property and that is CanActivate. Which assign with array of Guard. Mean we can have multiple guards before activating this route.

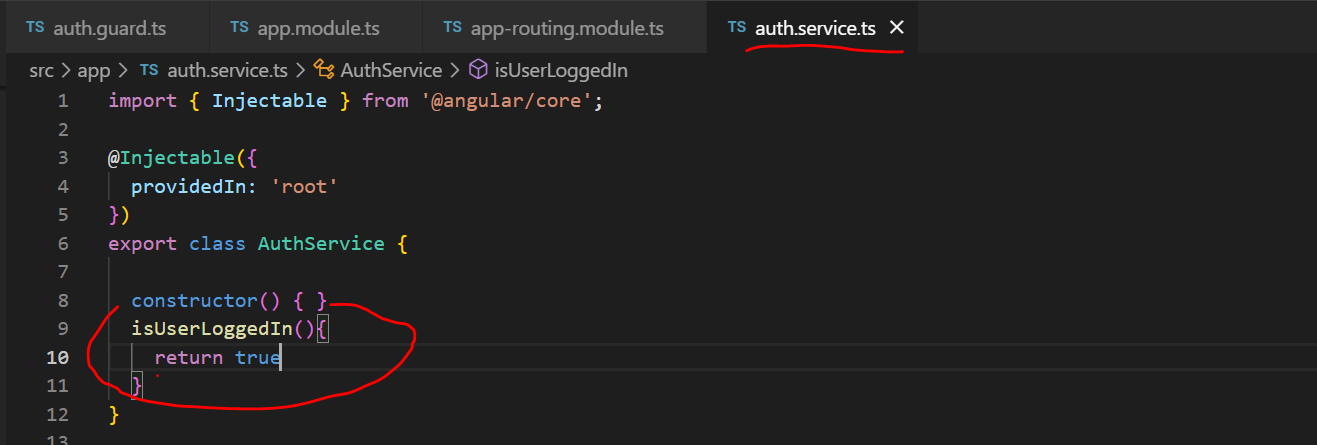
But here we have added only one guard so, we will use only one.



Now, we will use respective service file for creating function to check is user loggedIn or not.

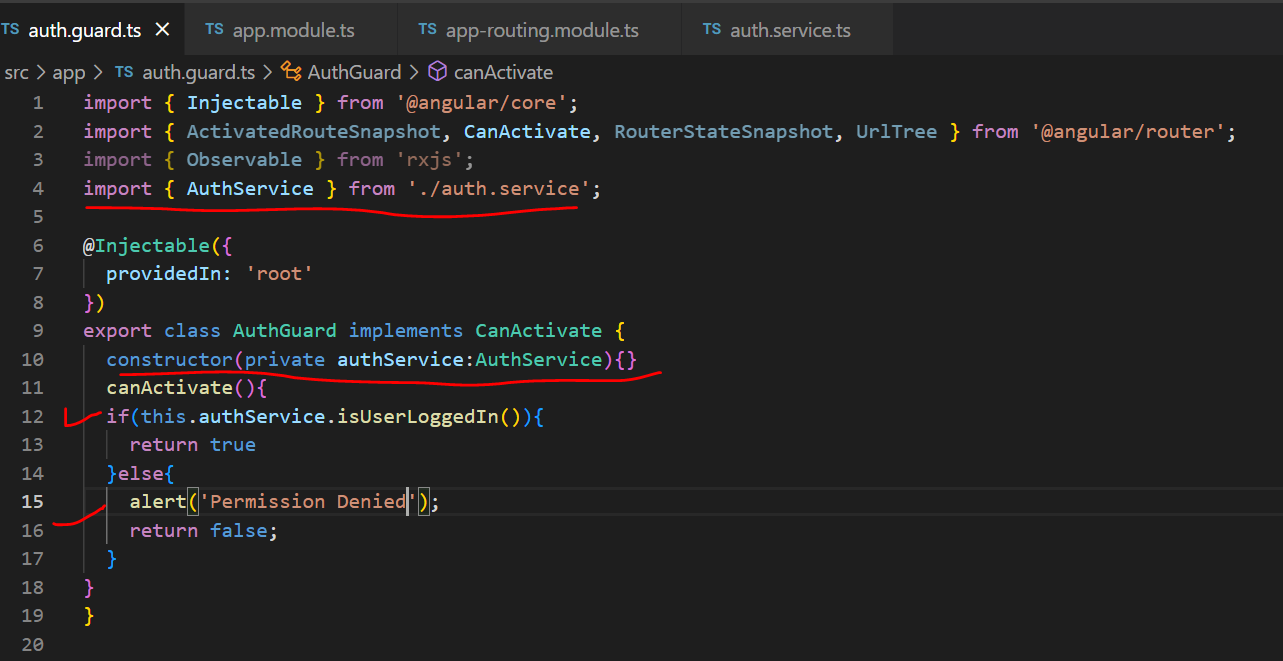
For this we will open guard.service.ts and add a function to checked Login successful or not?

**Step 4:** Open guard.service.ts file create a function to implement logic for login.



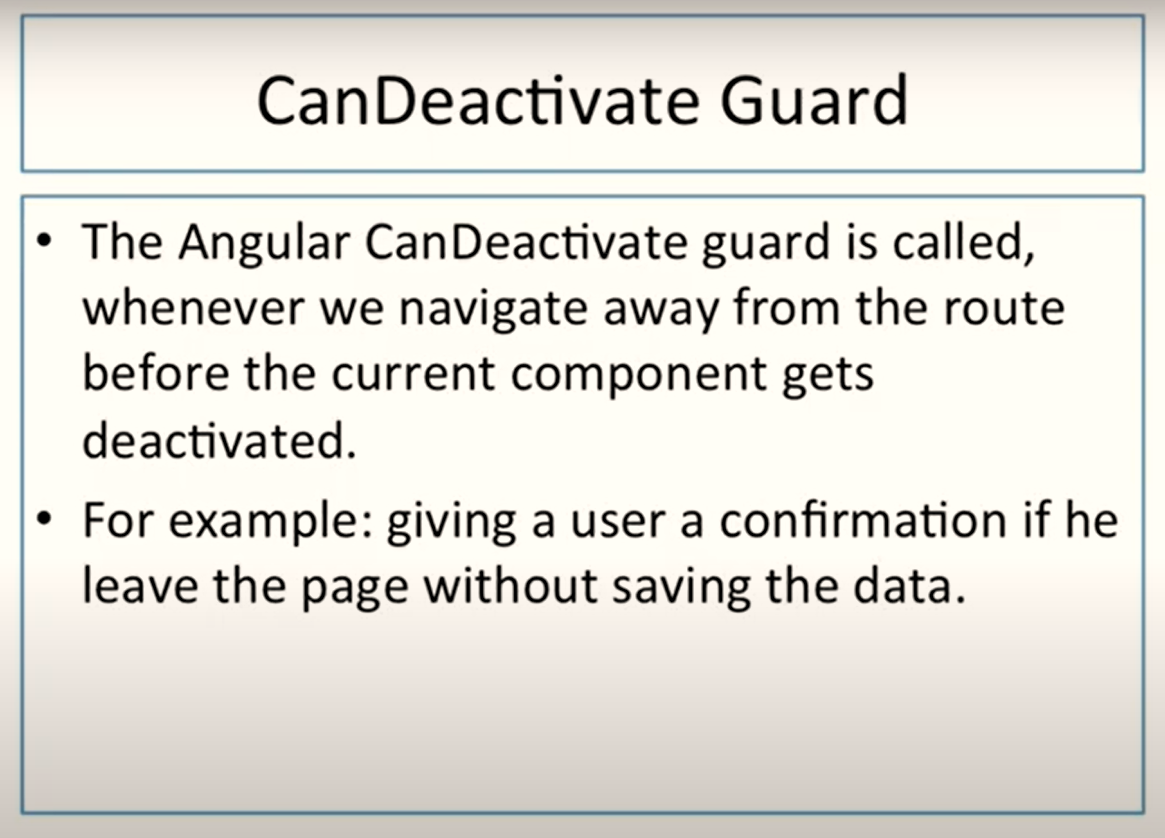
So, in this function we can specify whether user is authenticated or not.

**Step 5.**  Now, we will open auth.guard.ts file and inject service in constructor



If function in service will return true then this CanActivate allow user to access route and if it’s false then it restrict access.

CanDeactivate:



Assume we have a registration form or reactive form; I have filled up text boxes but before saving them I am clicking on other button so other route will be called. But I don’t want it should happen. I want when I will try to route to other page and I am filling the form then I should get notification.

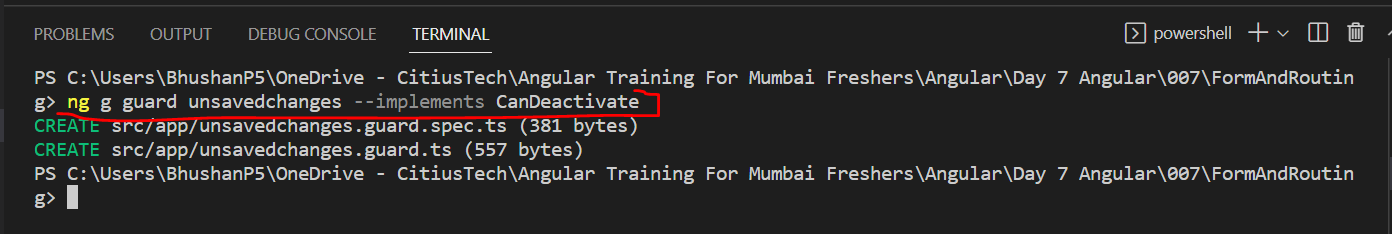
This behaviour we can achieve using **CanDeactivate.**

Here, I am using our previous project only and we will use canDeactivate on regForm.

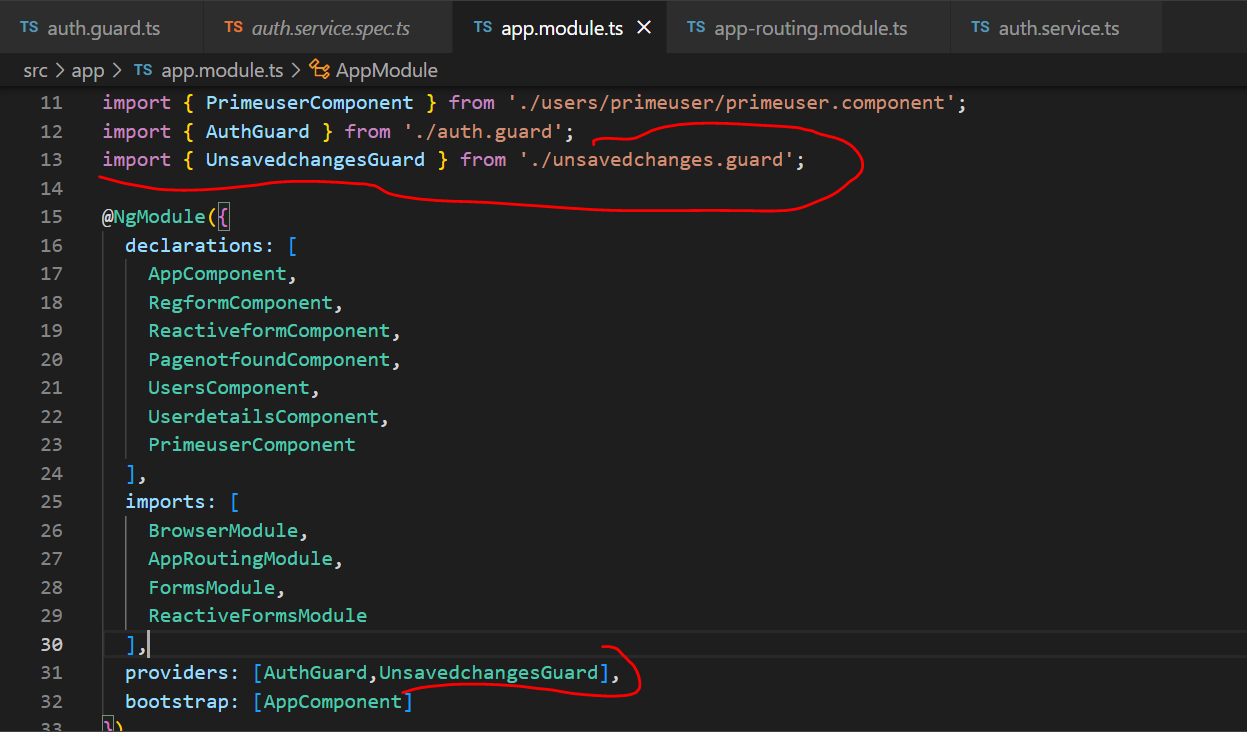
We will use Our previous project in which we have already created reactive form.

We have already created reactive form.

For this purpose we will add guard with implementation of canDeactivate

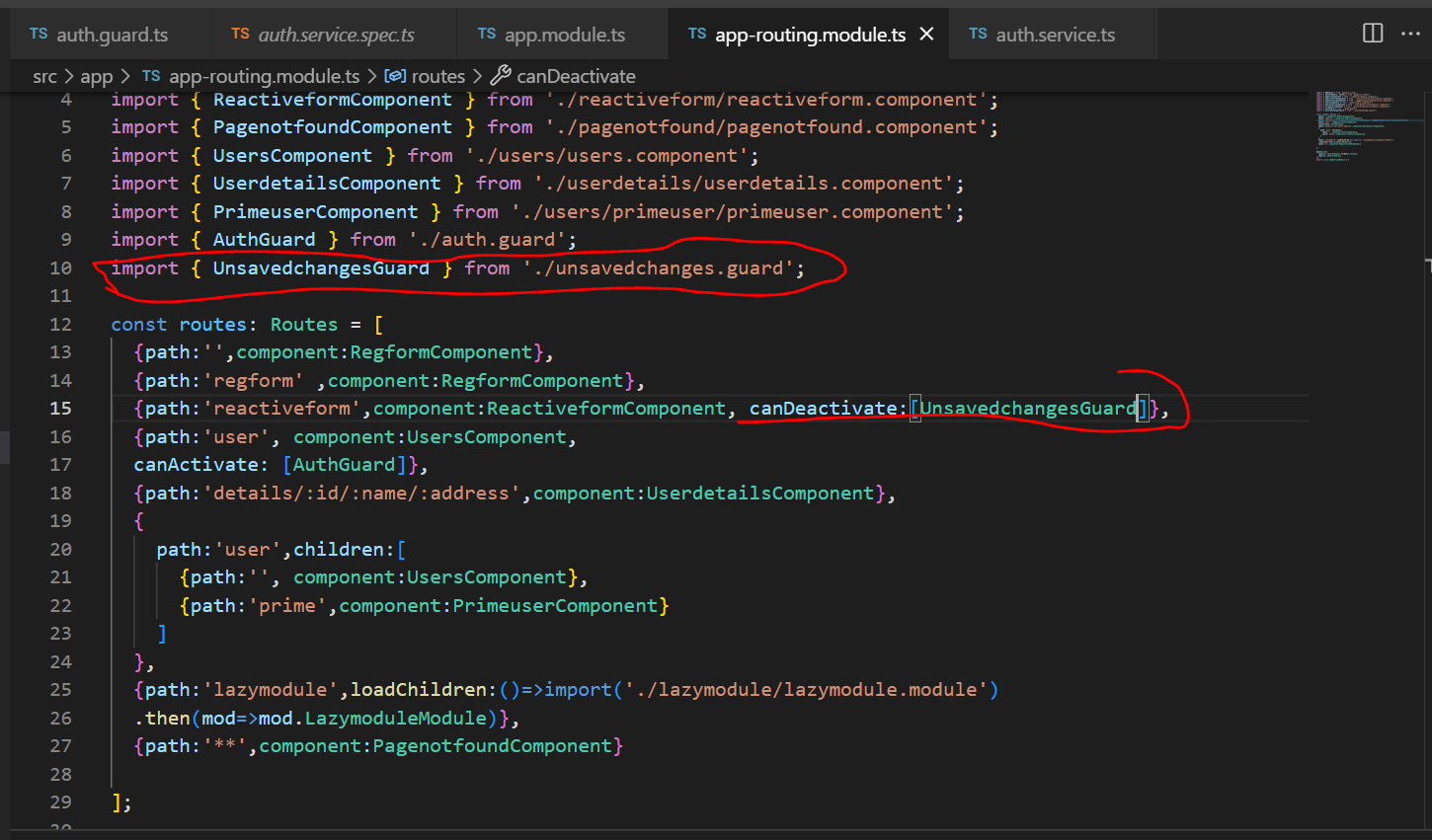


Now, import this guard in app.module.ts

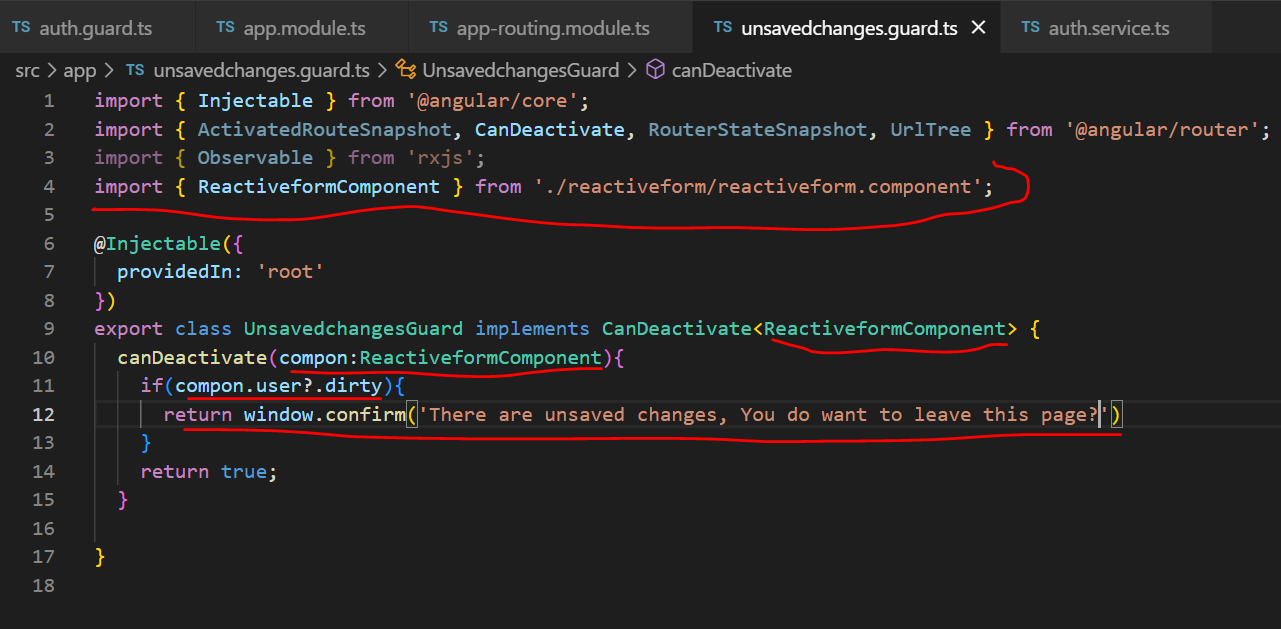


Now, open app-routing.module.ts

And add canDeactivate route



Now, open unsavedchanges.guard.ts



**Observable:**

In simple words, if we have to define observable then we can say Observable use to perform asynchronous operation and handle asynchronous data.

We have already seen How to handle asynchronous operations in java script, we have use promises for same purpose.

So, what is asynchronous operations and data.?

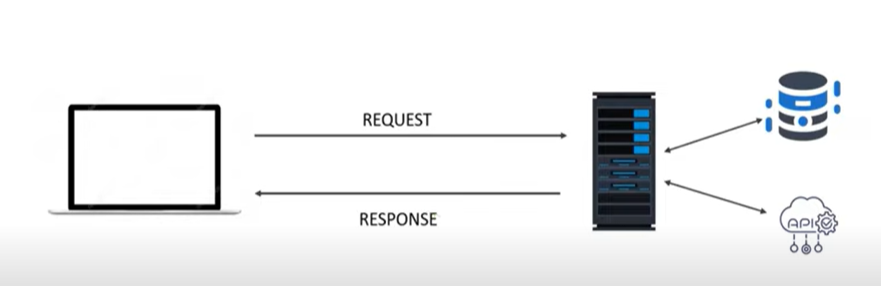
Ans: We know that Java script is single threaded programming language. It means code will get executed line by line. When certain portion execution will be completed then next portion of code will get execute.

Let’s understand it with example: Consider that our application is getting data from remote resource/server using http, then this might be possible that it may take some time to get data but till that time rest execution will be in waiting state. It will get executed only when http request will get completed.

Here we can say synchronous code is blocking in nature and therefore asynchronous programming comes in picture. Asynchronous code get executes in the background without blocking execution of main thread. So, in such scenarios we can use asynchronous programming because it won’t block execution of main thread.

So, asynchronous programming we can achieve using either by promise o by observable.

Let’s understand what differences between both of them are



1. Promise will collect all the data and send complete data at once but Observable will create packets of data which is also called as Stream of data. Those streams will be shared by Observable.
   1. Promise will definitely share the data so, data maybe actual data or if there will be any issue then promise will return error data
2. Promise will return you the complete data even if there is no code using that data but observable send the data if it is requested by any code.
3. A promise is native to java script while Observable is not native feature of angular or Java script. It s provided by another java script library called as RxJs- Reactive Extension for Java Script.

So, Technical definition of Observable is as follow:

An Observable is a function that converts ordinary stream of data into observable stream of data, Observable can also be considered as wrapper around ordinary stream of data.

As it’s native to RxJs, We should know what is RxJs?

RxJs is library of Java script use for working with asynchronous data stream.

For details you can visit official web site of RxJs.

RxJs has two main players

Observer

(Subscriber)

Observable

(Stream of data)

Observable keep sending the stream of data and if your code want to use the data emitted by Observable then we will have to subscribe it.

Let’s Understand it with Example.

First, we will create a new project for today’s demos.

For using Observable we need import RxJs library in our application and this RxJs will get installed automatically when we create new project.

Step 1. Import

import { Observable } from 'rxjs';

Step 2. Create Object of Observable.

  myObservable = new Observable();

To this initialization we have to pass call back function. This callback will receive arguments which is observer

 myObservable = new Observable((observer)=>{

  });

This observer will be injected by RxJs library. This observer is nothing but subscriber which is waiting for the data.

 myObservable = new Observable((observer)=>{

    // Log to console

    console.log("Observable Started")

    // Let's emits some data

    observer.next("John");

    observer.next("James");

    observer.next("Kevin");

    observer.next("Keanu");

    observer.next("Tom");

    observer.next("Tina");

    // This data will be emited if there will be subscriber.

    // If there won't be subscriber data won't be emited.

  });

Let’s create subscriber for this method.

We will create subscriber in ngOnInit method.

AppComponentClass🡪

xport class AppComponent implements OnInit{

  title = 'CRUDDemo';

  myObservable = new Observable((observer)=>{

    // Log to console

    console.log("Observable Started")

    // Let's emits some data

    observer.next("John");

    observer.next("James");

    observer.next("Kevin");

    observer.next("Keanu");

    observer.next("Tom");

    observer.next("Tina");

    // This data will be emited if there will be subscriber.

    // If there won't be subscriber data won't be emited.

    // lets try with some timeout

    setTimeout(()=>{observer.next("John")},1000)

    setTimeout(()=>{observer.next("James")},2000)

    setTimeout(()=>{observer.next("Kim")},3000)

    setTimeout(()=>{observer.next("Keanu")},4000)

    setTimeout(()=>{observer.next("Tom")},5000)

  });

  ngOnInit(): void {

  //  subscribe method takes three optional parameters

  // this.myObservable.subscribe(next, error,complete)

  this.myObservable.subscribe((val)=>{

    console.log(val)

  })

  }

}

As I told you that subscribe method accepts three optional parameters.

Next, error and complete.

  // Try with error message now with error method

    setTimeout(()=>{observer.next("John")},1000)

    setTimeout(()=>{observer.next("James")},2000)

    setTimeout(()=>{observer.next("Kim")},3000)

    // error message

    setTimeout(()=>{observer.error(new Error('Data Not Found Please Try later'))},4000)

    setTimeout(()=>{observer.next("Keanu")},4000)

    setTimeout(()=>{observer.next("Tom")},5000)

Final Code with error

import { Component , OnInit} from '@angular/core';

import { Observable } from 'rxjs';

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

export class AppComponent implements OnInit{

  title = 'CRUDDemo';

  myObservable = new Observable((observer)=>{

    // Log to console

    console.log("Observable Started")

    // Let's emits some data

    observer.next("John");

    observer.next("James");

    observer.next("Kevin");

    observer.next("Keanu");

    observer.next("Tom");

    observer.next("Tina");

    // This data will be emited if there will be subscriber.

    // If there won't be subscriber data won't be emited.

    // lets try with some timeout

    setTimeout(()=>{observer.next("John")},1000)

    setTimeout(()=>{observer.next("James")},2000)

    setTimeout(()=>{observer.next("Kim")},3000)

    setTimeout(()=>{observer.next("Keanu")},4000)

    setTimeout(()=>{observer.next("Tom")},5000)

    // Try with error message now with error method

    setTimeout(()=>{observer.next("John")},1000)

    setTimeout(()=>{observer.next("James")},2000)

    setTimeout(()=>{observer.next("Kim")},3000)

    // error message

    setTimeout(()=>{observer.error(new Error('Data Not Found Please Try later'))},4000)

    setTimeout(()=>{observer.next("Keanu")},4000)

    setTimeout(()=>{observer.next("Tom")},5000)

  });

  ngOnInit(): void {

  //  subscribe method takes three optional parameters

  // this.myObservable.subscribe(next, error,complete)

  this.myObservable.subscribe((val)=>{

    console.log(val)

  },(error)=>{

      alert(error.message)

  })

  }

}

Final Code with Complete

import { Component , OnInit} from '@angular/core';

import { Observable } from 'rxjs';

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

export class AppComponent implements OnInit{

  title = 'CRUDDemo';

  myObservable = new Observable((observer)=>{

    // Log to console

    console.log("Observable Started")

    // Let's emits some data

    observer.next("John");

    observer.next("James");

    observer.next("Kevin");

    observer.next("Keanu");

    observer.next("Tom");

    observer.next("Tina");

    // This data will be emited if there will be subscriber.

    // If there won't be subscriber data won't be emited.

    // lets try with some timeout

    setTimeout(()=>{observer.next("John")},1000)

    setTimeout(()=>{observer.next("James")},2000)

    setTimeout(()=>{observer.next("Kim")},3000)

    setTimeout(()=>{observer.next("Keanu")},4000)

    setTimeout(()=>{observer.next("Tom")},5000)

    // Try with error message now with error method

    setTimeout(()=>{observer.next("John")},1000)

    setTimeout(()=>{observer.next("James")},2000)

    setTimeout(()=>{observer.next("Kim")},3000)

    // error message

    // setTimeout(()=>{observer.error(new Error('Data Not Found Please Try later'))},4000)

    setTimeout(()=>{observer.next("Keanu")},4000)

    setTimeout(()=>{observer.next("Tom")},5000)

    // complete message

    setTimeout(()=>{observer.complete()},5000)

  });

  ngOnInit(): void {

  //  subscribe method takes three optional parameters

  // this.myObservable.subscribe(next, error,complete)

  this.myObservable.subscribe((val)=>{

    console.log(val)

  },(error)=>{

      alert(error.message)

  },()=>{

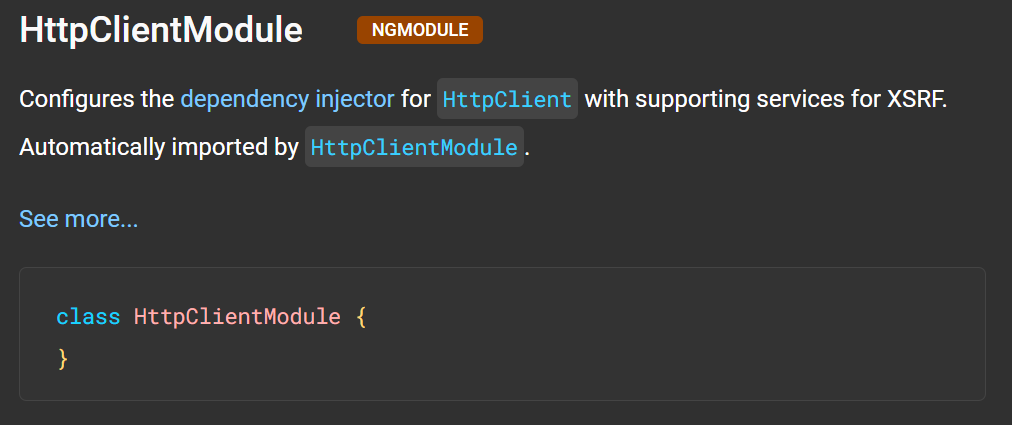
    alert('Date Fetch Completed')

  })

  }

}

**HttpClientModule and HttpClient services**



For HttpClient 🡪 <https://angular.io/api/common/http/HttpClient#description>

**Use Json server for performing CRUD:**

So, here for practice purpose we will create our own API. How to do it.

We will see

1. What is json Server?
2. How to install json server?
3. How to use it?
4. How to make API with it?
5. Test our json server with POSTMAN

A1. Get full fake REST API with zero coding

Follow given below steps

**Step 1.** To install json server

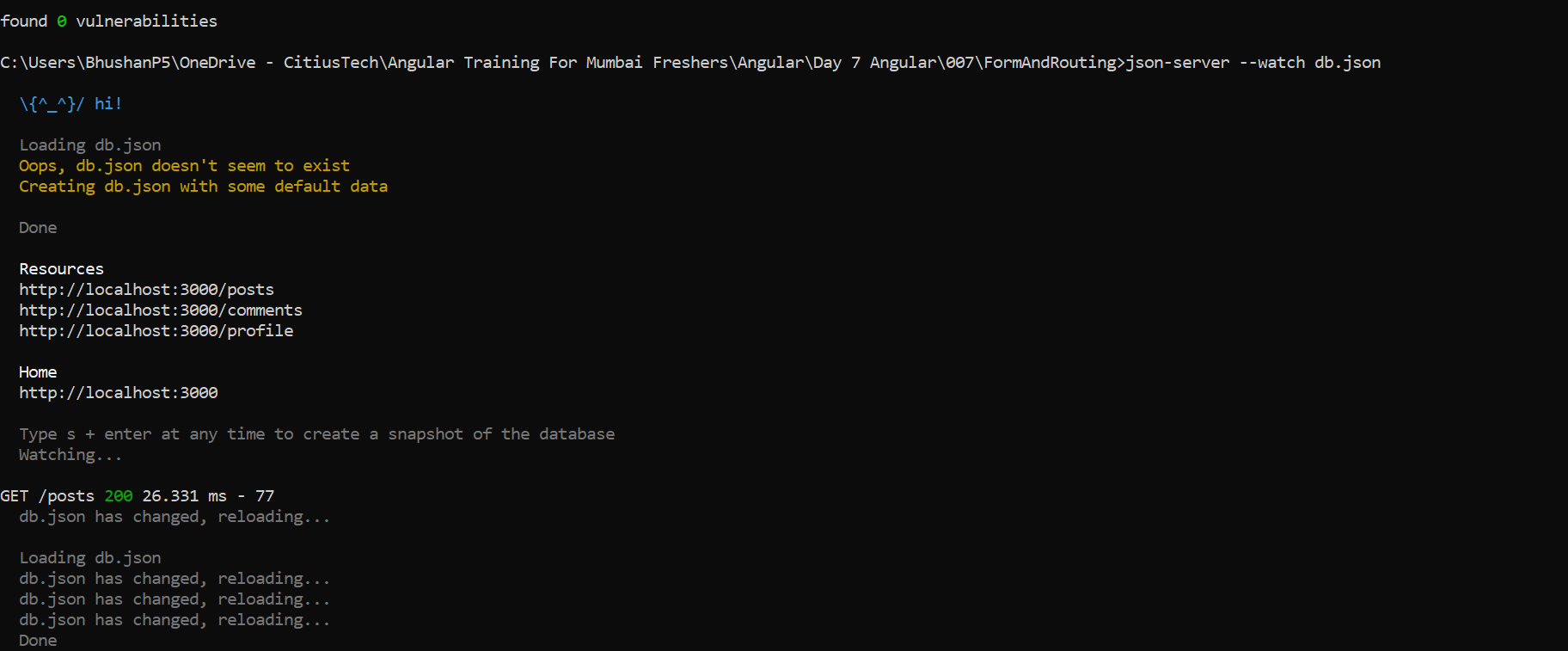
npm install -g json-server

Step 2. We will have to create a json file. This can be created explicitly by firing command

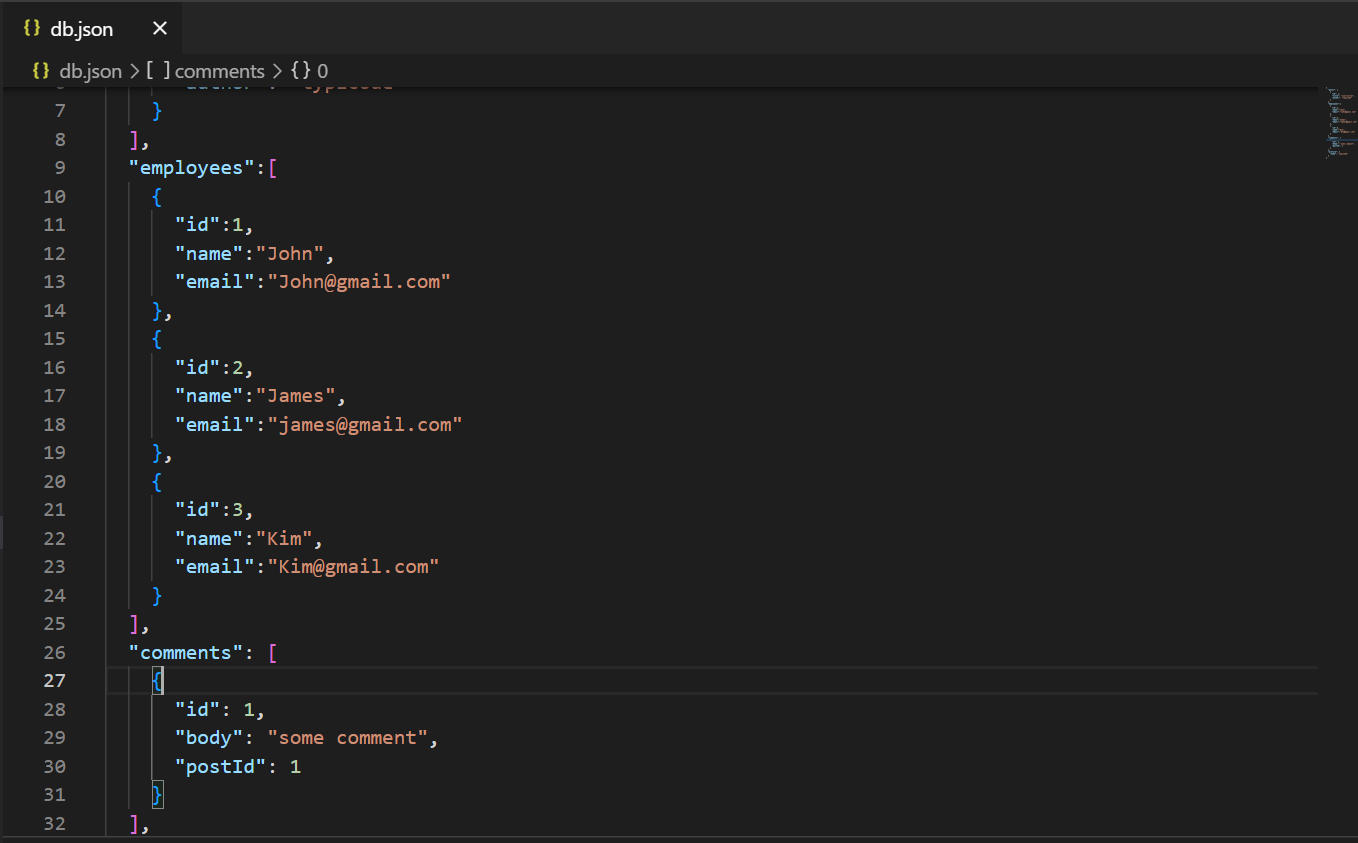
touch db.json

or

json-server --watch db.json



Add some data in this file



Now our web API created and if you will see, it will be working fine in browser.

Task For Today:

Follow and complete CRUD operation in Angular and Angular Material.

<https://www.youtube.com/watch?v=jGbP620NahE>