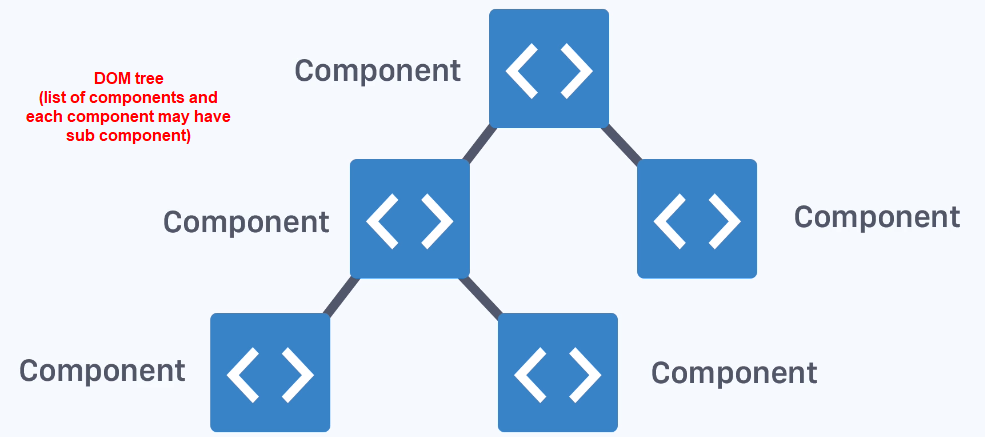
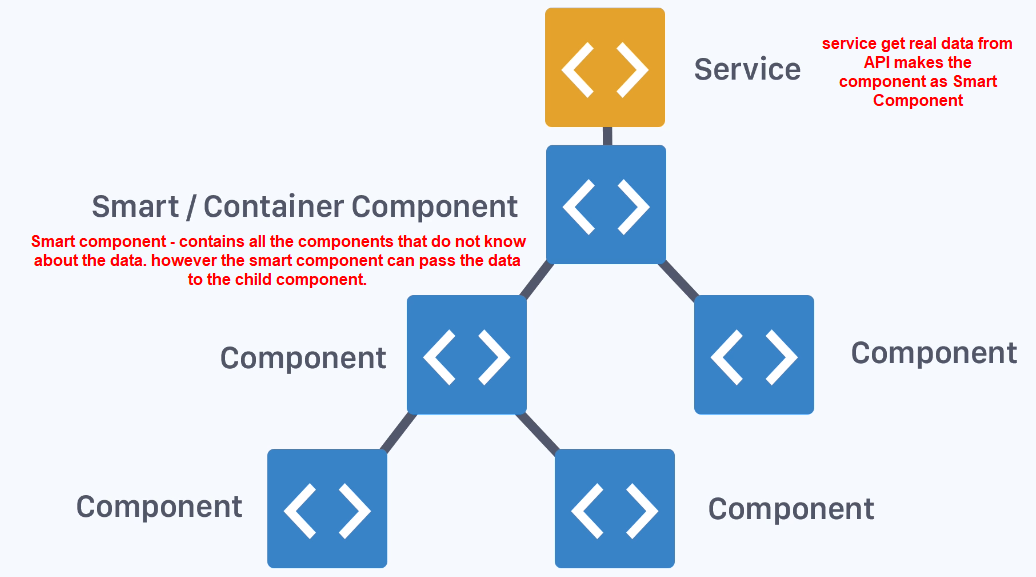
# Component Architecture and Feature Modules

npm install

npm run start





## Difference between Container & Presentational Components – High level overview

Container vs Presentational components or

Smart vs Dumb components

Stateful vs Stateless Components

Stateful/Container/Smart component:

Communicates with services, Render child components.

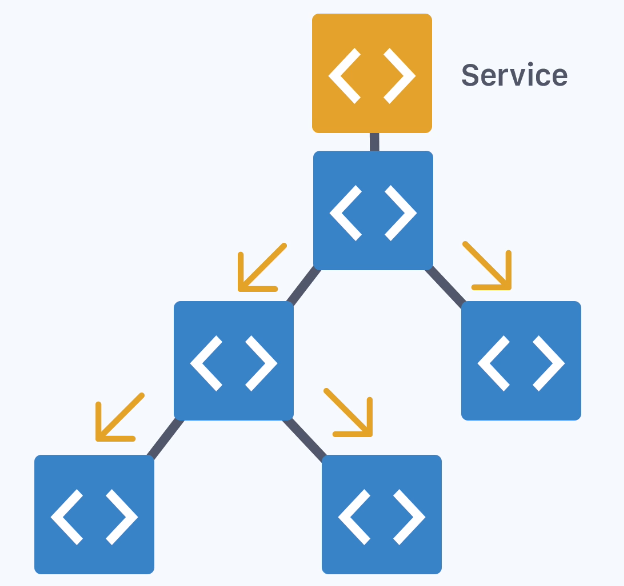
Stateless/Presentational/Dumb component:

Accept data via inputs, Emit data changes via event outputs.

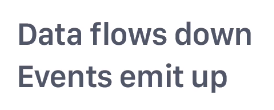
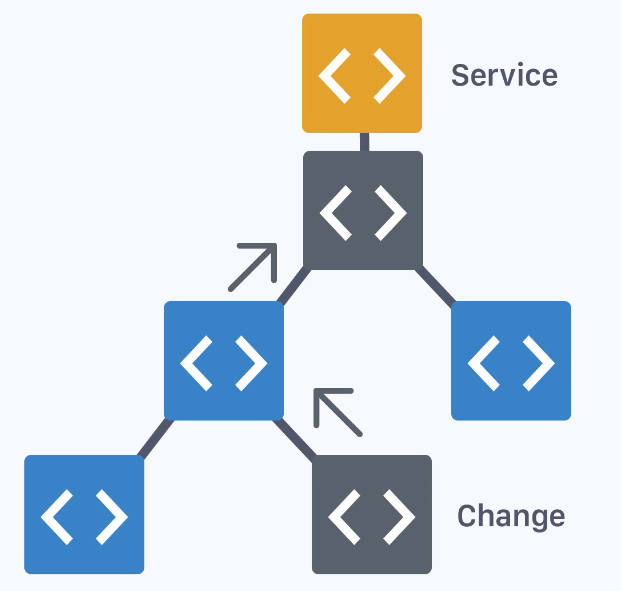
This is tied heavily with best practices of component architecture in angular.

## One-way dataflow overview – high level overview

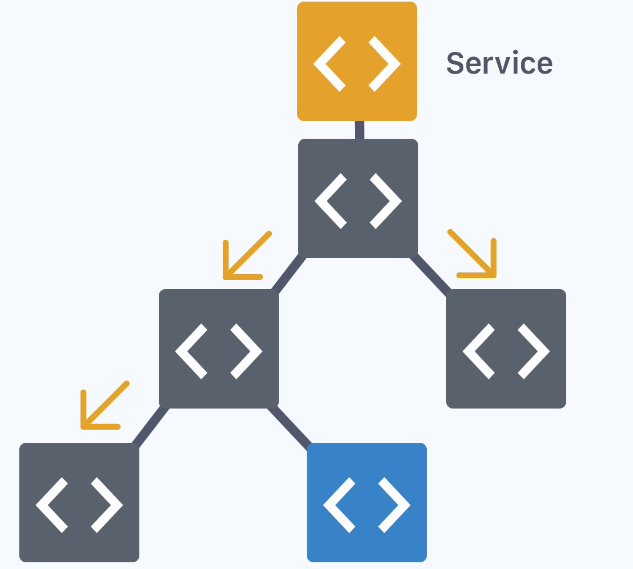
1. Data flows from Smart Component to Presentation Component



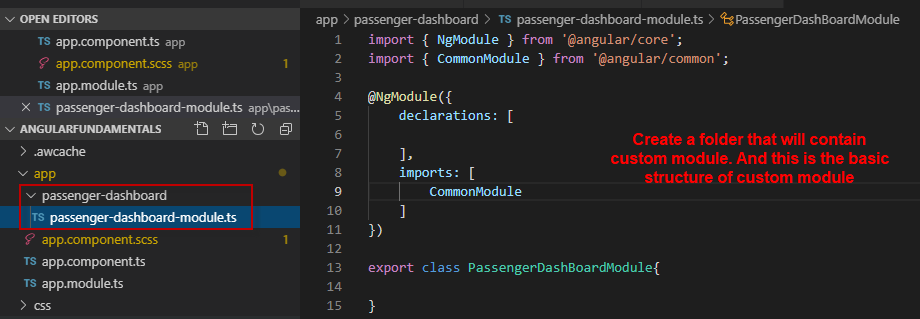
1. Change event in last component emits Up and reaches to Smart component via Presentation components.

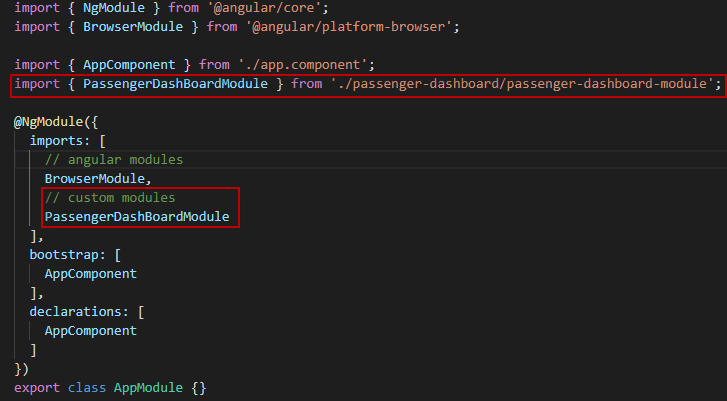
 

1. Then again smart component will push the changed data happened via Service back to presentation components.

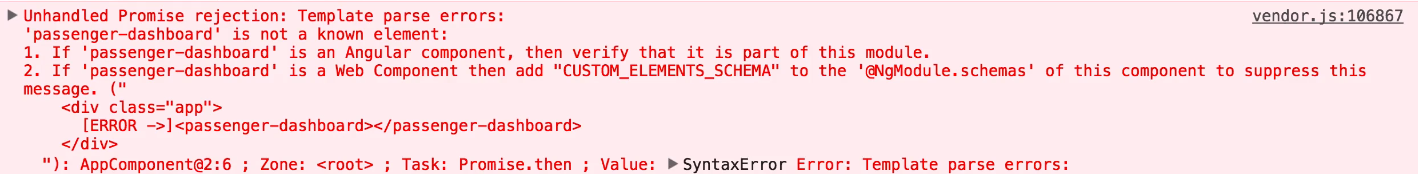


## Feature modules with @NgModule – Custom Module – Custom ng Module





## Creating a container (smart) component

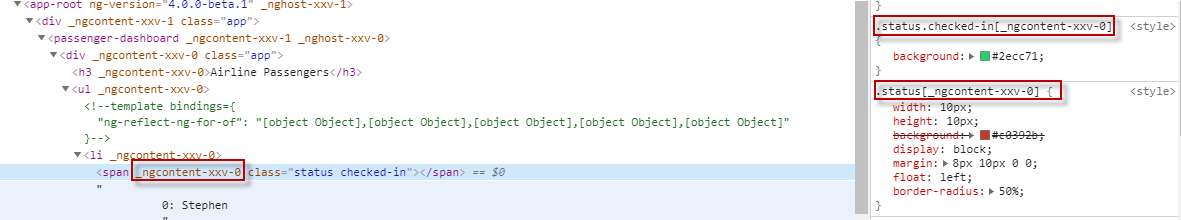


This means that custom module is not exporting the dashboard component, So we cannot use this in another module.

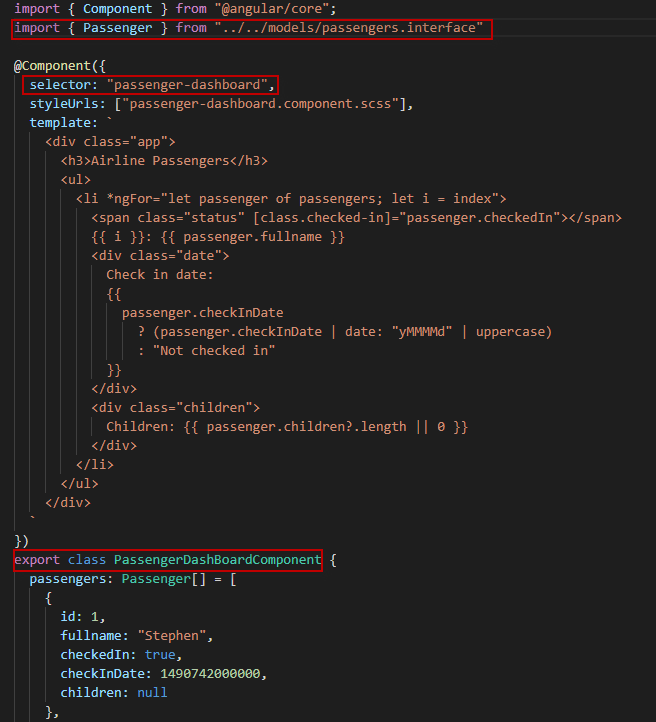
We will get encapsulated style. Styles are encapsulated to a component.



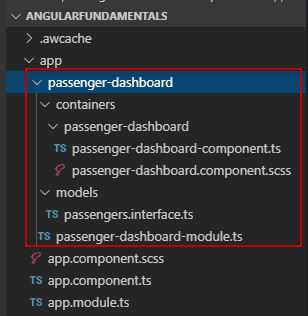
View capsulation. This will not have impact on any of the other modules that will be used in app module. All the styles are scoped to the particular component.



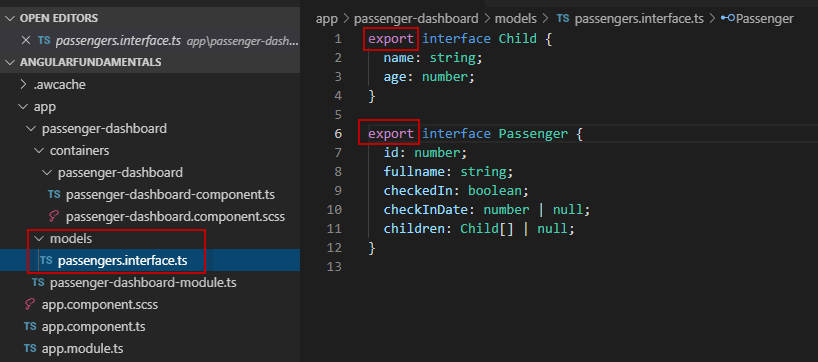
Created our first container component. We called it passenger-dashboard



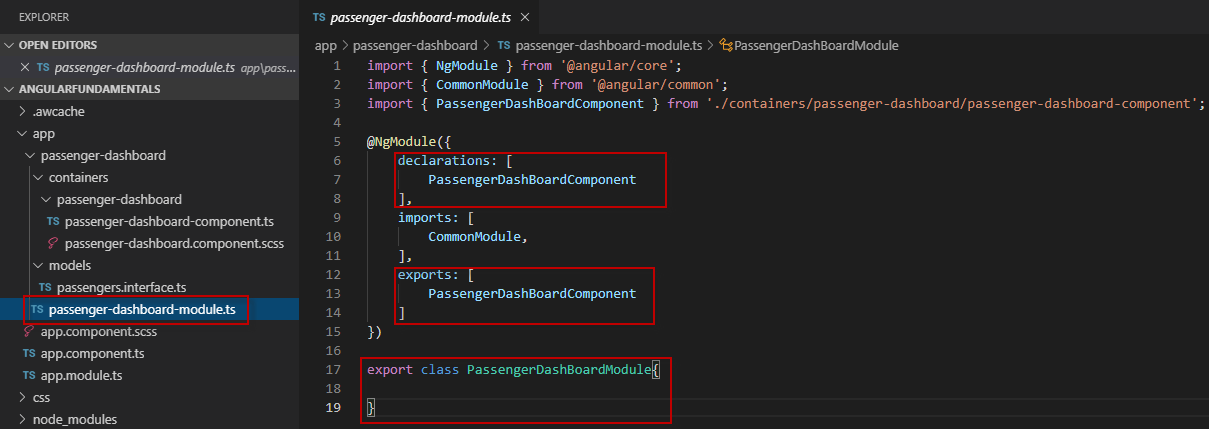
If we have multiple container components, we will be creating multiple folders.



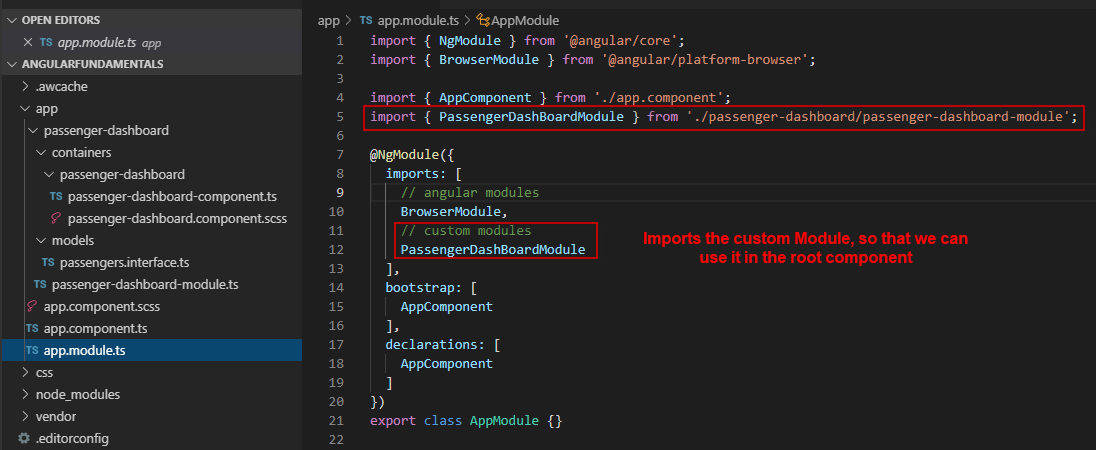
We have created passenger interface, which we can use in multiple files.



We have declared & exported our Dashboard component from our custom module module.ts



We have imported our dashboard module in app (main/root) module.

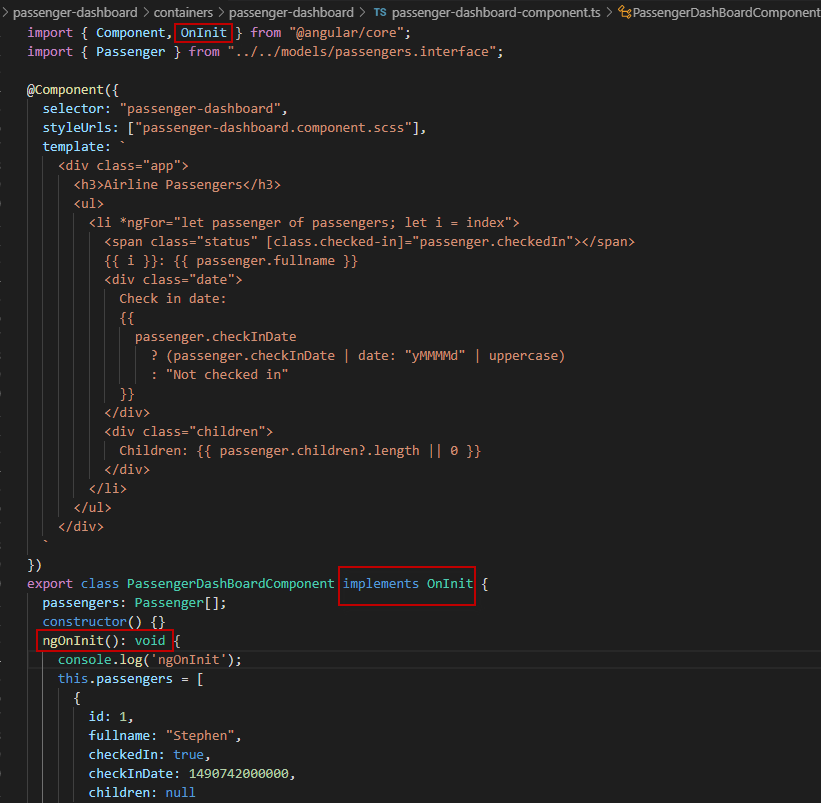


## ngOnInit lifecycle hook

Lifecycle hook is a function that get’s called by angular itself, when something happen.

ngOnInit – will be called when the component is initiated like loading of the bunch of static data before the component is ready.

We can rely on the constructor function for using like a service (implementing http module)



## Presentational (dumb) components

One of the fundamental attributes of container component is it contains data and it renders stateless child component.

How to create child components and how to pass the data from smart component to the child component.

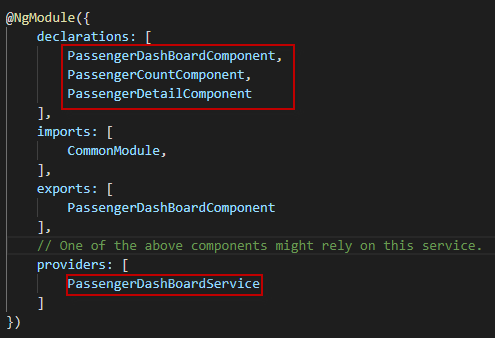
ngOnChanges lifecycle hook

If the data changed is instantly reflected, it is not strictly biding by the rules of one-way data flow. The component has its own local state When will click on any action then only we need to update parent data.

This is where we can introduce ngOnChanges

# Services, Http and Observables

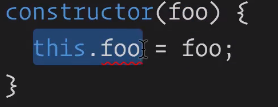
## Data Services and Dependency Injection



Make the service available to any of the components. This is done with something called as dependency injection.

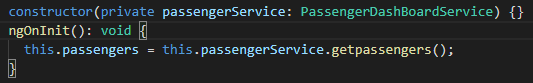
Dependency Injection:

Use the constructor function in ES6. Take the reference and bind it to the local property so that the local property can be used.



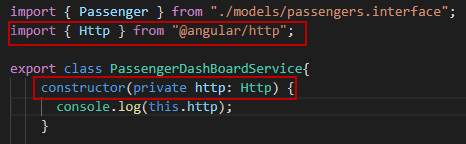
In Typescript:

private keyword in constructor will allow us to inject the dependency



## Understanding @Injectable

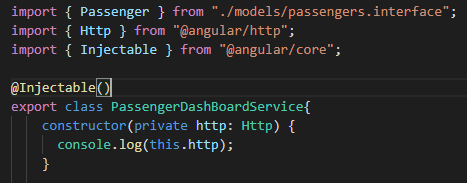
Inject http Service through constructor



Console.log will throw an error

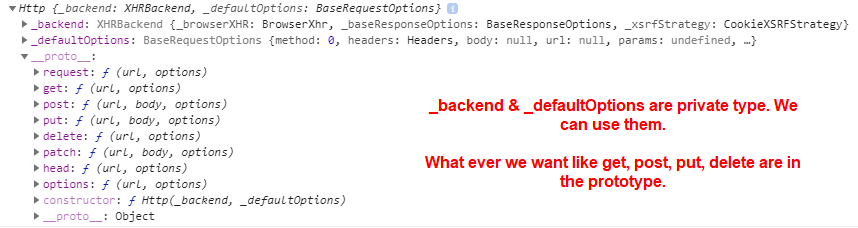


Some services would not be having any dependency, but when we do rely on particular provider such as Http we need to import Injectable.

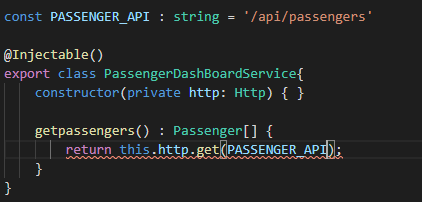


@Injectable will tell angular service that In PassengerDashBoardService we can inject things into its constructor.

This is Http object.



## Http data fetching with Observables



Issue 1:

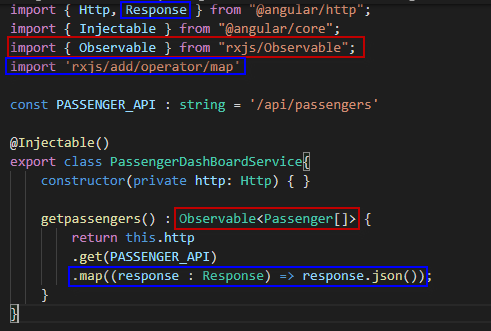
Type 'Observable<Response>' is missing the following properties from type 'Passenger[]': length, pop, push, concat, and 26 more.ts(2740)

Issue 2:

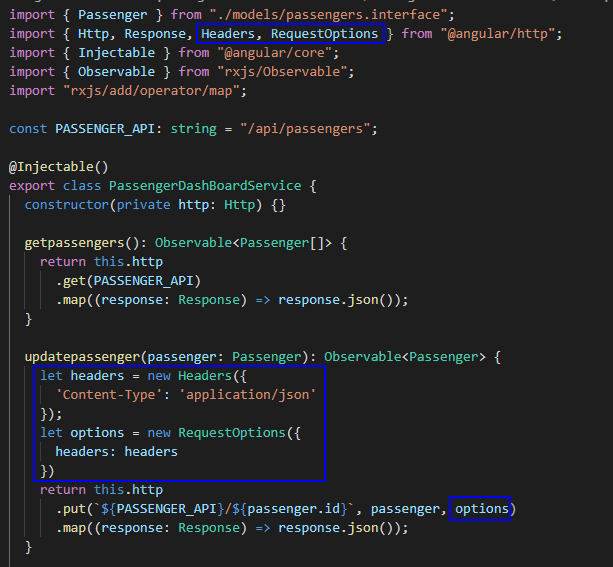
Type 'Observable<Response>' is not assignable to type 'Observable<Passenger[]>'.

Type 'Response' is missing the following properties from type 'Passenger[]': length, pop, push, concat, and 27 more.ts(2322)

Line in read for Issue 1 and Line in blue for Issue 2.

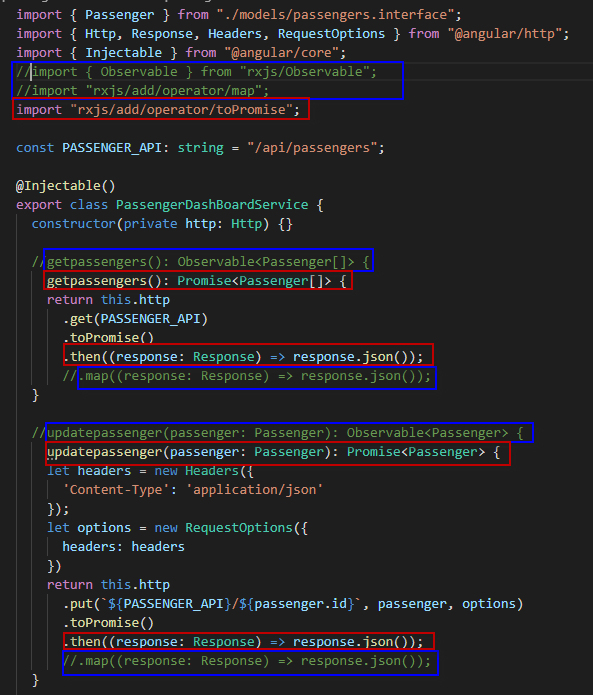


## Custom Headers and RequestOptions

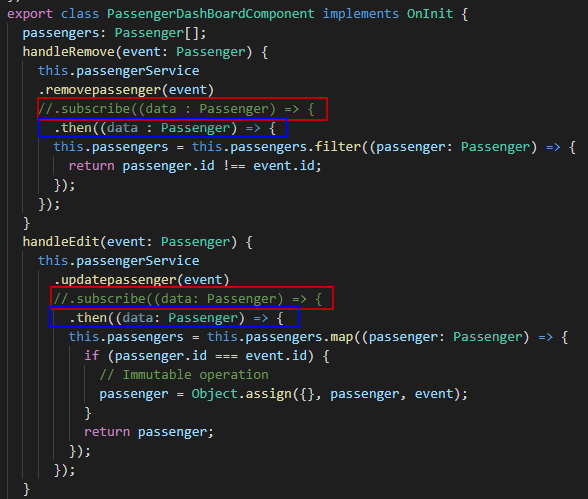


## Http Promises alternative

If we don’t want to use Observable / map in service and subscribe in component that use the service.

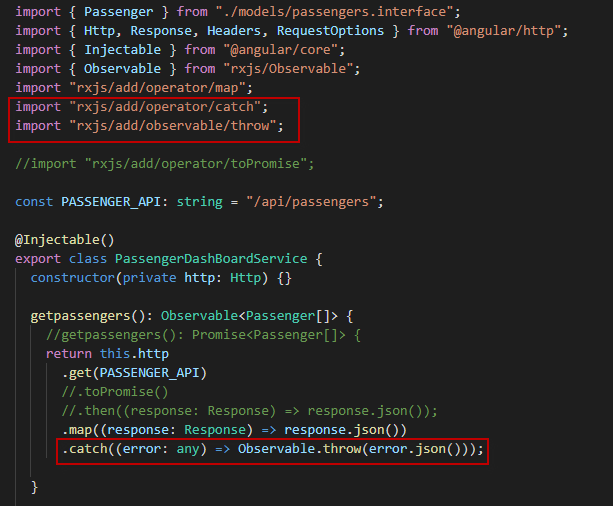


We can go with toPromise



## Observable.throw error handling

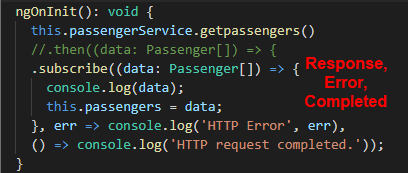
In service



In Smart Component

RxJs subscribe and error callbacks

* a success handler function, which is called each time that the stream emits a value
* an error handler function, that gets called only if an error occurs. This handler receives the error itself
* a completion handler function, that gets called only if the stream completes



# Template-driven Forms, Inputs and Validation

Forms container component / Angular Forms

In Template-driven Form its better if we doesnot assign null and go with optional parameter

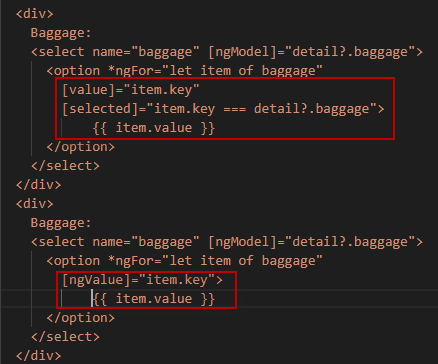
checkInDate?: number;

is better than

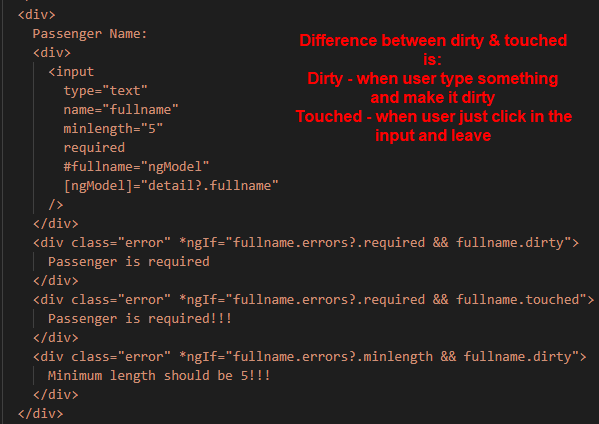
checkInDate: number | null;

What does [ngValue] do?

Below two highlighted points does the same.



Difference between dirty & touched

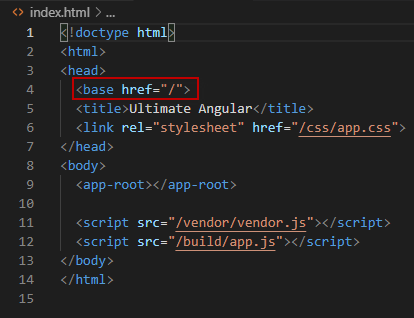


Disable the button click if form error is there

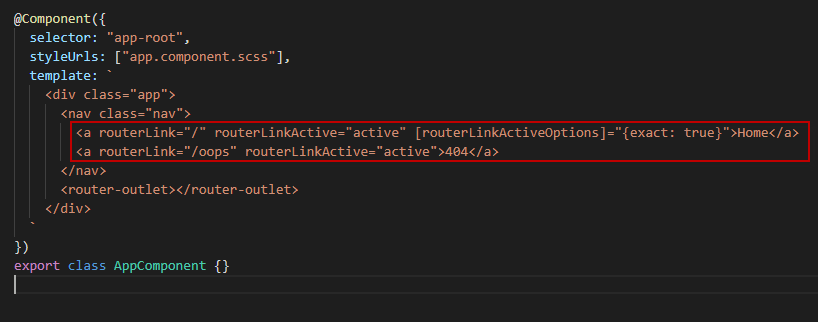


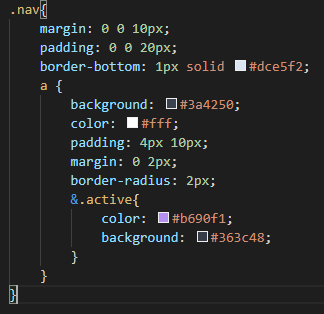
# Component Routing

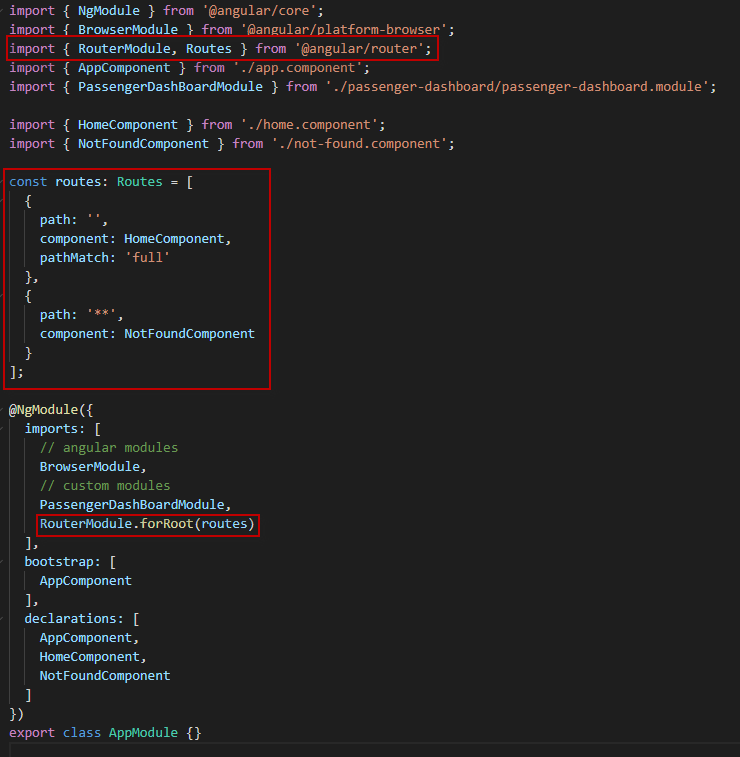
Base href and RouterModule



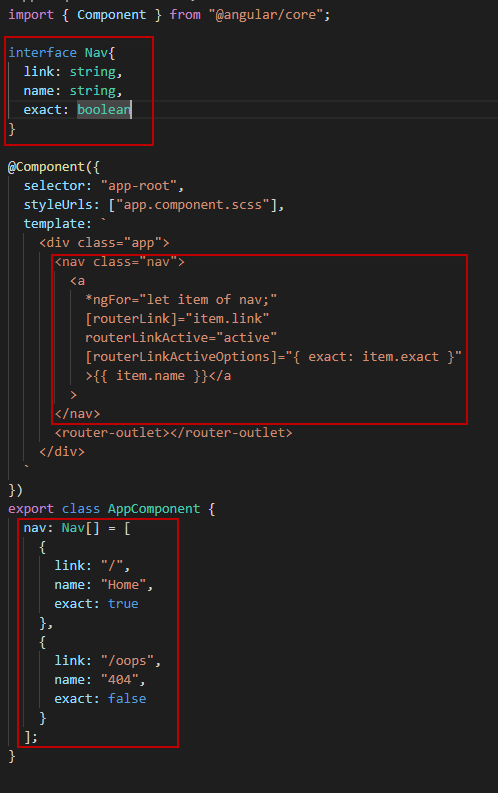
To use the router, we need this included all times. Otherwise angular apps will not work.







## Dynamic navigation with ngFor



## Feature-module routes with forChild()

