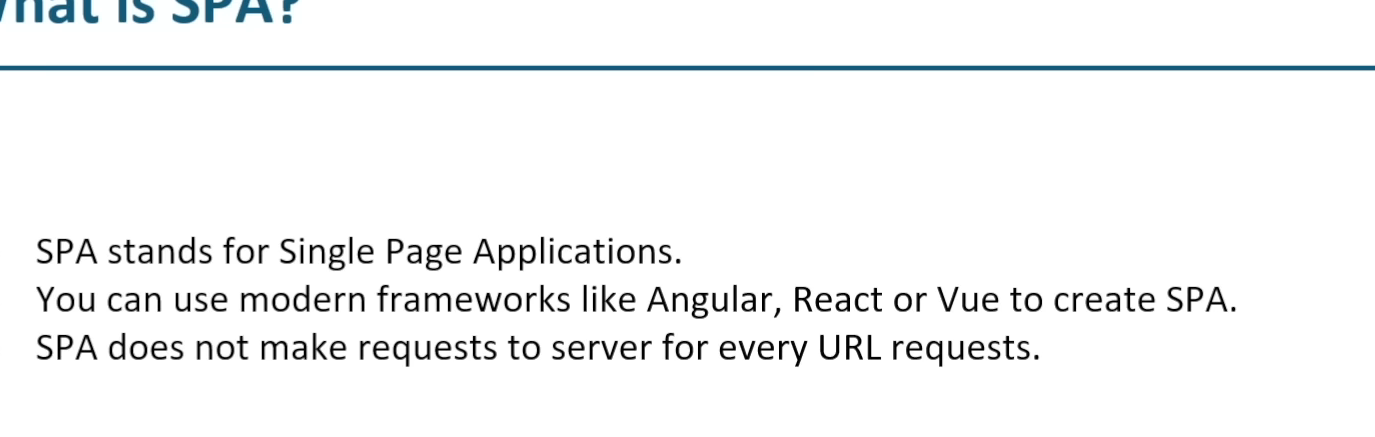
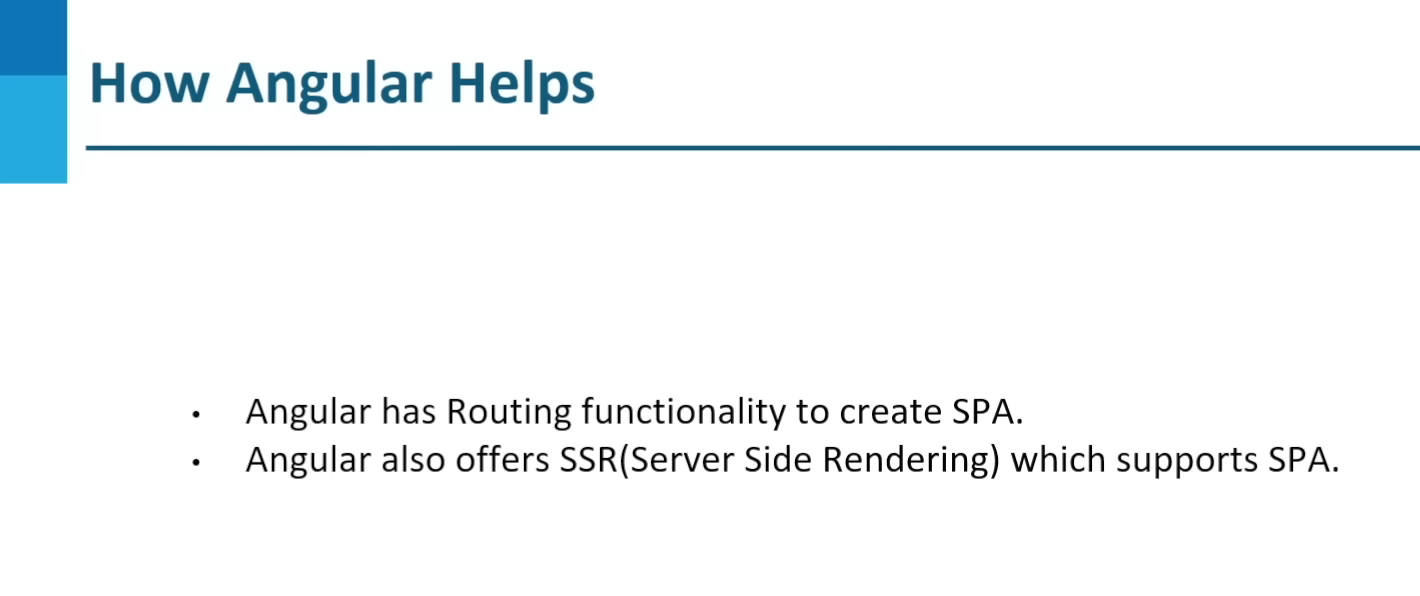
TYPESCRIPT & ANGULAR

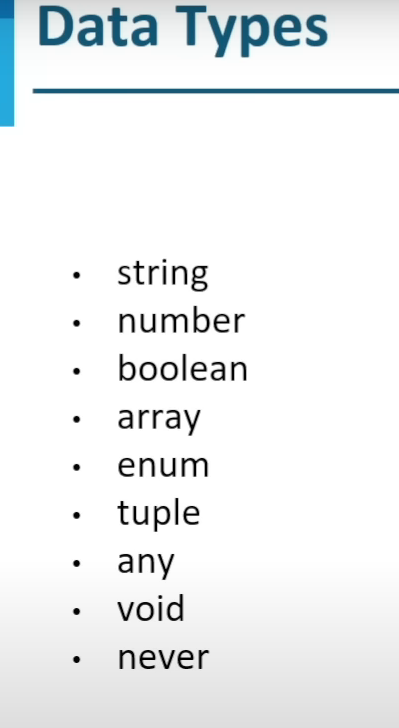
**#Topic: What is SPA**

1. Single page application
2. You can use lot of framework like angular react vue.js to create SPA
3. SPA doesn’t make e





# #TOPIC: DATA TYPES



Interface will not be the part of the production build it will be removed at runtime in TS

So we should go with the class in TS

#TUPLE EXAMPLE :

In arrays we didn’t get the compile time error if we try to access the out of bound index

But in tuple we get one error

Example of tuple is:

Let testNumber : [number,number];

function swapNumbers(n1 :number , n2:number):[number,number] {

  return [n2,n1];

}

swapNumber = swapNumbers(10,20);

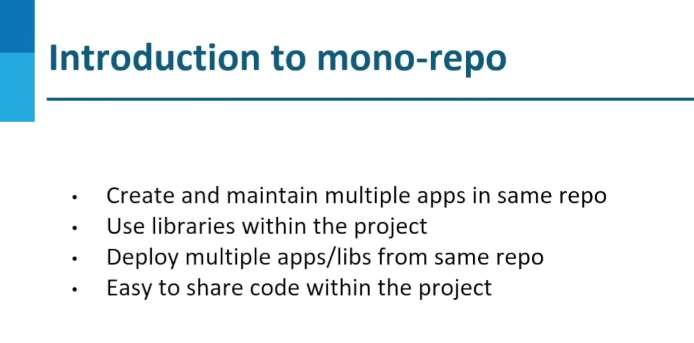
console.log(swapNumber[1]);

#TOPIC : DECORATORS IN ANGULAR:  
This can be applied on class to change the behavior of the class and method or property at runtime

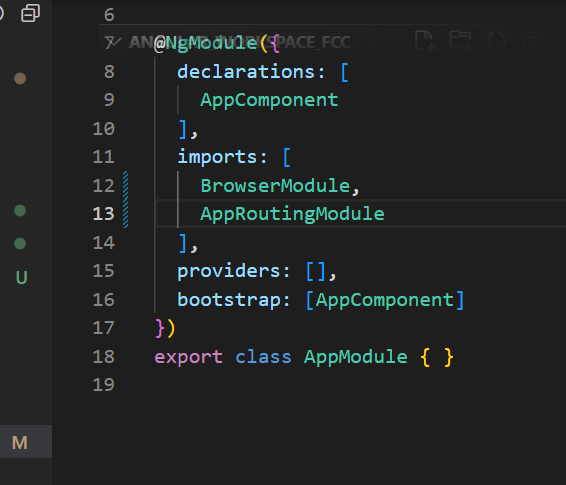
This is used internally in angular

Examples

#Pollyfils.ts fils : this file will help us backward compatibility with the browsers

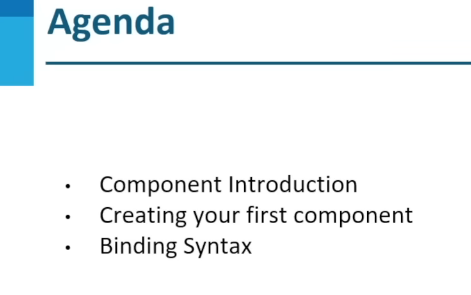


Angular module that needs to be updated in this



In imports array we need to define all the external module

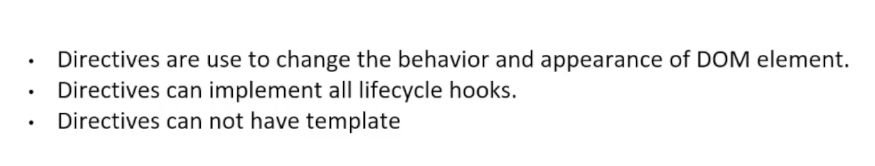
1. @NgModule – is a decorator modifies the behavior of our class
2. Bootstrap:[] when we run the angular which component loads first that we need to define into this bootstrap
3. In the App module we will define all the pipe and custom modules and routing



Binding Syntax :

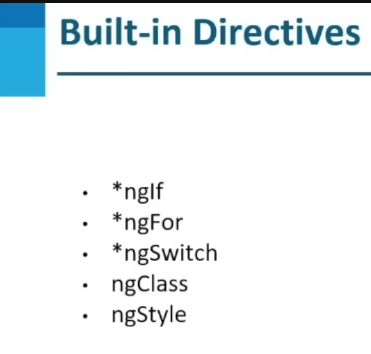
1. Interpolation
2. Property
3. Event Binding

Directives In angular :  
types of Directive



Types Of Directive :

1. Structural
2. Attribute



What is this star mean to differentiate between the Structural and attribute directive

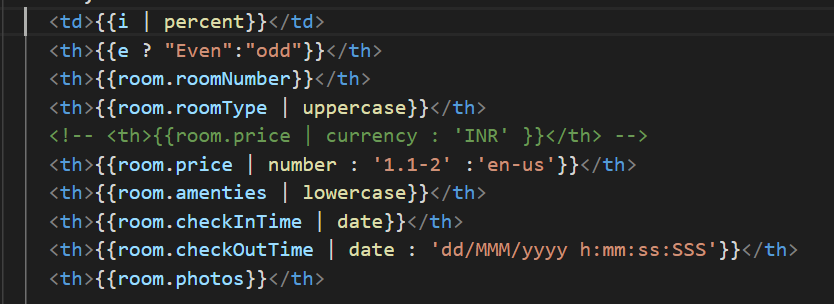
Which will change the dom and modified it this Is known as structural directives

[ngStyles] = “{‘color: predicate ? ‘green’:’red’}”;

We can define the condition and change DOM accordingly

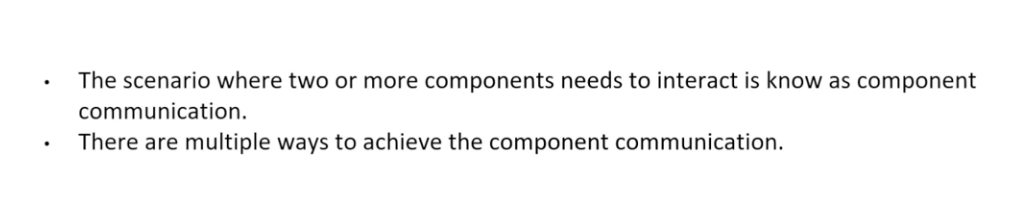
PIPES

* Pipes help us to transform the data from one form to another form
* It doesn’t change the value of the object



Different pipe to transform the data

# Component Communication



There are three ways to interact with other component :



To use the different component proper we can user either @Input and @Output

or we can use the @ViewChild to access the component value

Example :

 @ViewChild('name',{static:true}) name! : ElementRef;

  ngAfterViewInit(): void {

    console.log(this.name);

    this.name.nativeElement.innerText='Hamilton Hotel Companies'

  }

<app-room></app-room>

<div #name>

</div>

With the help of # we can make the dom variable from which we can access the tag of the html element !

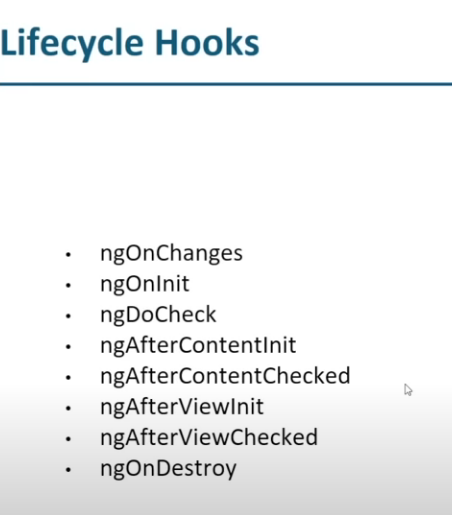
@ViewChild('user',{read:ViewContainerRef}) vcr! : ViewContainerRef;

  ngAfterViewInit(): void {

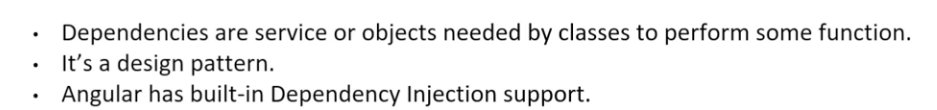
    const componentRef = this.vcr.createComponent(RoomComponent);

    componentRef.instance.numberOfRooms=15;

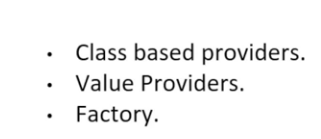
# Life Cycle Hooks :



# Dependency Injection



DI Provider :



Always define the services in the Constructor () and define the service as private so that template should not be able to access that!!

@Injectable({

  providedIn: 'root'

})

We will check this later

