18 December 2025 09:02

Revision -

Angular Framework = container manages Components
Purpose of Decorator - communicate with the framework
@Component - this class should be managed by Framework
@Injectable = this service should be managed by Framework
= the Framework creates the object of the service
+
container can inject the dependency of this service into component

Reusable components = available to other components as tags (defined by selector)
= configurable through the attributes - @Input() - decorator indicates that the value of this property must be populated through the attributes (done by framework)

```
<ReusableCompSelector  displaymessage=""  info="" />  //attribute
Class  ReusableCompClass
{
        @Input() displaymessage:string=""    //property
        @Input("info")  infomessage:string = ""
}
```

Reusable component can send messages to the parent component

```
<ReusableCompSelector   info=""   (colorchange) ="parenthandler($event)"  />

 class ReusableCompClass
 {
                @Output()  colorchange=new EventEmitter<string>() ;


                Func()
                {
                        Colorchange.emit(VALUE )
                }
 }
```

Two way data binding - send data from view to model and viceversa
[(ngModel)] - FormsModule must be imported

Template reference variable - variable works in the html file

```
<input  #inp  type=number  ....  />

{{inp.value}}

<button  (click) ="handler(inp.value)"> OK </button>
```

SEQUENCE of the SYNCHRONOUS CALLBACK

```
1st    let arr = [10,20,30]
2nd     console.log("before")
//synchronous callback
```

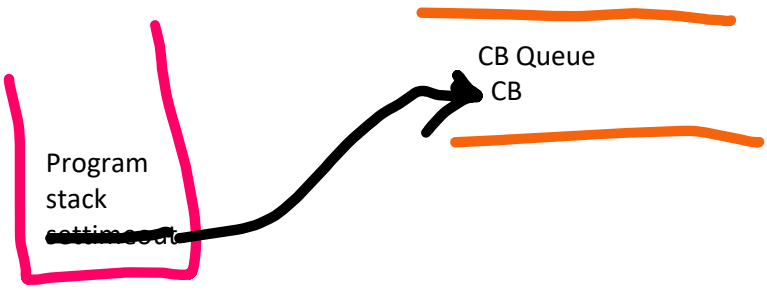
```
3rd  arr.forEach(
       4th,5th,6th (e)=>{console.log("CB called and returns for
element",e)}
       )
    3rd call returns
7th console.log("after")
```

1. Asynchronous Callbacks

```
1st  console.log("before")
// Asynchronous  - the execution is DEFERRED - later
2nd setTimeout(
     CB is PUSHED to a CB queue  ()=>{console.log("CB called after
2000ms")},2000
    )
  2nd call return

3rd console.log("after")

After 2000ms    4th  CB
```



After the program stack is empty then CB queue is executed as per the event

_____

If you want to do something after the CB runs and returns

Where will you write that code ?

Thumb rule --- write it in the Callback , NOT in the sequential code

_____

Promise = helps us handle POST callback code

Promise = something that will happen in FUTURE !!

Promise has states = 1. PENDING 2. RESOLVED 3. REJECTED

Promise is a object

It has functions

then

catch

In the **then function we will REGISTER TWO CALLBACKS**

One in the event that promise resolves
One in the event that promise is rejected

These callbacks will called after the result of promise arrives

_____

async , await ------

```
async function myf1()
{
        await new Promise(….) ;   // await waits for the promise to resolve
        //some code here  ###  - this code will run after the  promise resolves

}
```

_____

AJAX = Asynchronous Javascript API for XML

1. SPA

FIRST HTTP REQUEST (http://localhost:4200) --------- First http RESPONSE
browser <===== index.html + images in public folder + main.js +css **< web server**

SUBSEQUENT HTTP REQUESTS ----------
SECOND HTTP REQUEST and SECOND http Response FETCHES DATA from server
browser <---------demand DATA usually as JSON -->**web server**

AJAX helps us in firing the HTTP REQUEST Asynchronously
WITHOUT affecting the current view of the USER
User gets a SEAMLESS experience !!!!

In built HttpClient library for AJAX in Angular
External libraries like axios is also used

_____
For trying out AJAX we will use a REST SERVICE that provides some data

GET ,POST ,PUT ,DELETE requests

1. Cd earth
   npm install axios

2. Create a component ajaxexample = ng g c ajaxexample

3. open the html
   add a button - on click - add a handler

4. open the TS file
   Implement the handler and fire AJAX get request
   show the data in console log

_____
In the handler ----------

use Promise + then(CB1 , CB2)
OR
use await + Promise

Rest Service = returned data of 30 user records in one GO

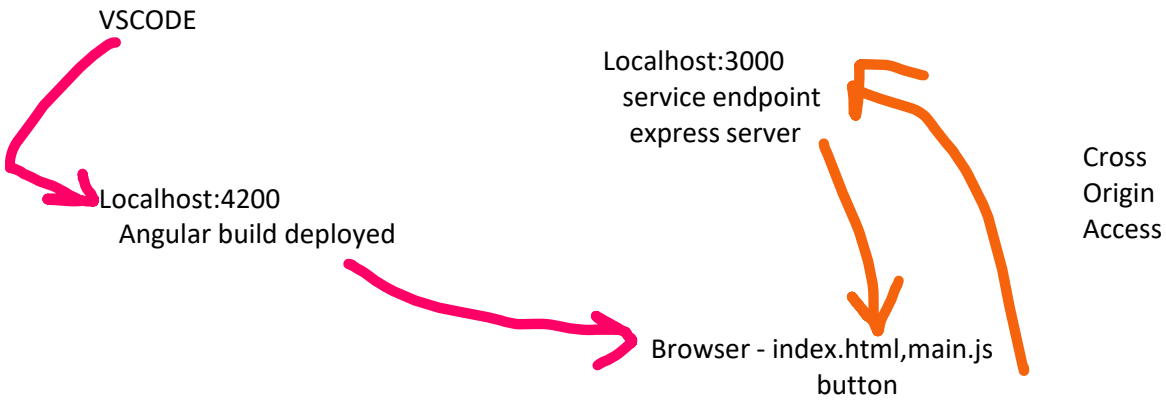
Rest Service = streaming data --- staggered response
ONE HTTP-REQUEST ----------ONE SERVICE CALL
<----------part1
<----------part2
<----------part3
THE RESPONSE is completed !!!

Create a folder MYSERVER
cd myserver
npm init }} to initialize a node project
npm install express

create a file MYSERVER/server.js
Copy the code that I have shared on the chat

node server.js }} RUN

Open the browser http://localhost:3000/stream
check if the data is streaming



Observables
Observer Pattern

Observers/Subscribers

observable ---- it is EMITTING DATA [cricket match]

Subscriber subscriber subscriber

RxJs library - gives the API for Observables !!!!

```
cd MYSERVER
npm install rxjs
```

TO WRITE CODE for ASYNCHRONOUS CALLBACKS

1. promise + then
2. async await promise
3. Observable promise

PIPES in Angular = VIEW Level convenience code

P1 | P2 = output of P1 is piped to P2

LIFE CYCLE of the COMPONENTS -----

WHO writes the Component - Programmer
WHO manages the Component - Framework
WHO manages the life cycle of the Component
- Framework

Component Life Cycle

1. Framework - find the component
2. It will create the objects of the component
   by calling the constructor
3. It will inject the dependencies
4. track the changes in the @Input fields
5. track changes in the different properties
6. Do the change detection and call for rerendering
7. Destroy the Component (make it out of scope)

Life cycle Hooks !!!

Hook = callback functions

Programmer provides the functions and angular will call them!!

Programmer wants some code to be executed
at some life cycle events /points

we are interested in executing ajax call just as my component view is loading

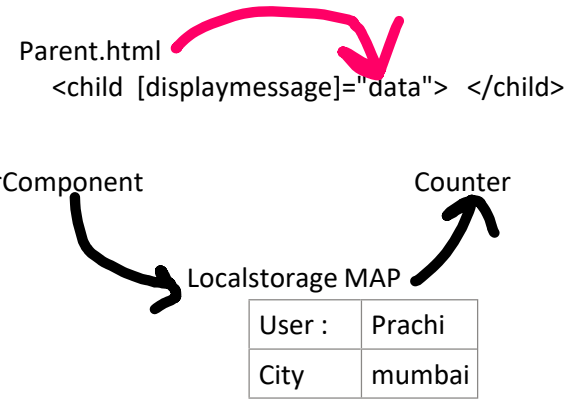
Just before the component disappears from the UI fire ajax call for saving log information

---

Data Storage ----------
For other components

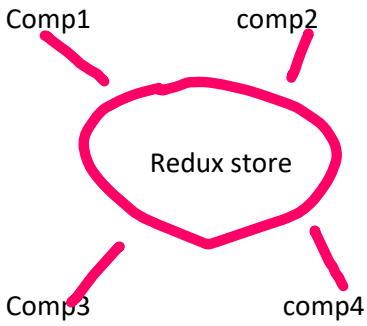
1. Localstorage = storage provided by the BROWSER
   MAP = each entry in the map is a key.value pair !!!
   Name="alpha"

2. Redux Store = Observable = it can be built using the redux libraries



Very careful - do not save passwords , auth keys
Usernames , sensitive data - store by encrypting

Structured way to share data between components - Observable



@Component
{
}
Classs MathsComponent
{
private utility: MathUtil = inject(MathUtil)
OR
constructor(private utility:MathUtil) { }
}

Name of the dependency = utility
Type of the depedency = MathUtil
Yes MathUtil is a Service -
created using @injectable - it is available to Angular

Who creates MathsComponent object ? Created by Angular
Angular will also created object of MathsUtil and set it in the utility variable
This is called as DI