HTML5

HTML5:

* **Simplified DOCTYPE declaration** – specifies the version of the application
* <!DOCTYPE html>
* //determines the version of HTML
* //is used so that browser understands the version
* //Otherwise the browser will take the default version
* **Semantic Tags**

Used to represent the separate tag instead of using <div>. it is same as <div>. It is used to identify what kind of page content

<nav></nav>

<header></header>

<footer></footer>

<article></article>

<section></section>

<aside></aside>

* **Form Controls and attributes**

-*text , password, radio, checkbox, button*

-*textarea*

-*select*

-*email, date, datetime, color, search, range , number*

“You can validate the content without using any kind of scripts like jquery”

\_\_\_\_\_\_\_\_\_\_

Previous - <input type=”Summit” value =”Click me”/>

New -> button is the container where we can implement anything inside the container

<button type=”submit”> Click me </button>

**Attributes**-> used for validation

1.placeholder Attribute – replaces the Label

2.required

3.min, max

4.maxlength

5.pattern – regular Expression

6.NoValidate- Applied in the form – html5 have builtin validation for Form inorder to avoid that we will use this attribute. Or if you want the custom validation from the external like angular.

7. Formaction , Formmethod – used in the “Submit”

//Normal form

<form action="index.php" method="post">

    <button type=”submit”> Submit </button>

</form>

//form post - causes problem - controversy

<form action="index.php" method="post">

    <button type=”submit”> save to file </button>

    <button type=”submit”> save to DB </button>

</form>

<form action="index.php" method="post">

    <button type=”submit”> save to file </button>

    <button type=”submit” formmethod="GET"> GET From DB </button>

</form>

// when u click "save to file" it will post to index.php

//when u click "save to DB" it will post to indexDB.php

<form action="index.php" method="post">

    <button type=”submit”> save to file </button>

    <button type=”submit” formmethod="indexDB.php"> Save to DB </button>

</form>

* **Media elements**

**-**video

<video width="320" height="240" autoplay>  
  <source src="movie.mp4" type="video/mp4">  
  <source src="movie.ogg" type="video/ogg">  
</video>

**-**audio

<audio controls>  
  <source src="horse.ogg" type="audio/ogg">  
  <source src="horse.mp3" type="audio/mpeg">  
</audio>

**-**canvas 🡪 typically used for drawing 2D and 3D elements using the javascript -> online games

<canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;">  
</canvas>

* **Storages**
* **Local Storage -**  store the data locally into the browser (localstorage.set, localstorage.get) – cache the data locally. Stored in key value pair format
* <body>
* <div id="result"></div>
* <script>
* *// Check browser support*
* if (typeof(Storage) !== "undefined") {
* *// Store*
* localStorage.setItem("lastname", "Smith");
* *// Retrieve*
* document.getElementById("result").innerHTML = localStorage.getItem("lastname");
* } else {
* document.getElementById("result").innerHTML = "Sorry, your browser does not support Web Storage...";
* }
* </script>
* </body>

**Session Storage –** The sessionStorage object is equal to the localStorage object, except that it stores the data for only one session. The data is deleted when the user closes the specific browser tab.

<body>

    <div id="result"></div>

    <script>

        if (sessionStorage.clickcount) {

            sessionStorage.clickcount = Number(sessionStorage.clickcount) + 1;

        } else {

            sessionStorage.clickcount = 1;

        }

        document.getElementById("result").innerHTML = "You have clicked the button " +

            sessionStorage.clickcount + " time(s) in this session.";

    </script>

</body>

* **Database**
  + **IndexedDB**

"IndexedDB is a low-level API for client-side storage of significant amounts of structured data, including files/blobs. This API uses indexes to enable high performance searches of this data. While DOM Storage is useful for storing smaller amounts of data, it is less useful for storing larger amounts of structured data. IndexedDB provides a solution."

* + **WebSQL**
* **Real Time Applications**
  + SSE – Server Sent Events (Old and one way messaging)
  + WebSocket (SignalR) [ws Protocol]– introduced by W3 along with HTMP. Every communication depends on the https protocol.

If there is any change in the data in the database. It will not notify the client. The client needs to refresh the page. But this is not a real time thing. Eg) stock market . In this case the server needs to push the data automatically. So we use WS protocol. It needs more coding (server address, subscription etc) . Microsoft have created SignalR- it is the library. It does the work automatically.

But some of the browser does not support Websocket. So it will switch back to SSE that is a old mechanism.

  <nav>

        Page Navigation Links

    </nav>

    <header>

        Page Header

    </header>

    <aside>

        Side bar content

    </aside>

    <section>

        <article>

            <h2>Article Header</h2>

            <p> This is article text</p>

        </article>

        <article>

            <h2>Article 2 header</h2>

            <p>Article Content</p>

        </article>

    </section>

**JQUERY**

* Javascript Library—Javascript is object based language
* Dom manipulation , data convention

*ECMA Script*:

* It is a version which have different feature
* Also called as ES – vs 2017, sql 2014
* ES6 also called as ES2015

***ES6 introduced***

* Let Keyword

Var -> Global scope – belongs to the nearest function

Let -> Local Scope – Belongs to the particular block

* Class
  + Constructor
  + Member methods
  + Properties
* Template strings -> changes the lengthy text , then we can use `kldfkdk`

**Line Breaks**

`string text line 1

string text line 2`

var a = 5;

var b = 10;

console.log**(`Fifteen is ${a + b} and not ${2 \* a + b}.`);**

* Arrow Functions
* var sum = (x, y) => x + y
* var dosomething = (x, y) => {
* //multiple lines of code here
* Return x + y;
* }
* Let result = sum(1,2);
* Rest and spread operators

**HTTP**

-http Methods

* GET – Get data from server
* POST – posting data back to the server. Create a new resource
* PUT- update the existing data. Replaced by the new entry
* PATCH – it is used to update the existing data partially – eg) age of the particular person
* DELETE- Delete the existing record

-http Headers

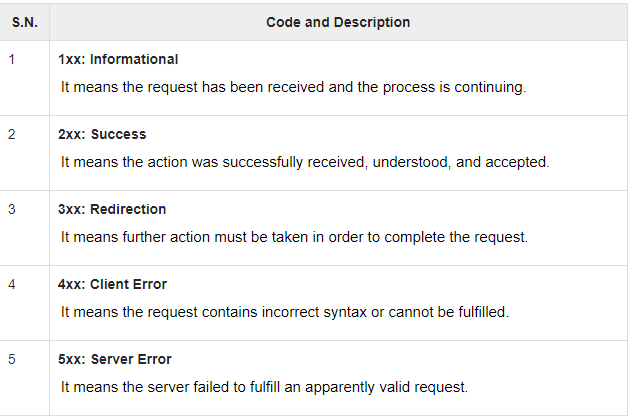
* Every request and response have a set of headers
* CONTENT-TYPE -> header field indicates the media type of the **associated representation**
  + Content-Type : Request – type of data you are sending to the server
  + Content-Type : Response – Data sent by the server
* ACCEPT-TYPE ->  field can be used by user agents to specify **response** media types that are acceptable.
  + - MIME format- Multipurpose Internet Mail Extensions (MIME) – represents what type of data we are sending / receiving
    - Majortype/ minortype
      * *Text*-
        + Plain – text/plain
        + Xml- text/xml
        + Json- text/html
        + Javascript- text/javascript
        + Css- text/css
      * *image*-
        + image/jpg
        + image/png
        + image/bmp
      * *audio*-
        + audio/mp3
        + audio/ogg
        + audio/wav
        + audio/midi
      * *video*-
        + video/mp4
        + video/ogg
        + video/wav
      * *application* – covers the formats other than the above types- The application content type allows for the transmission of application data or binary data, effectively allowing for specific handling of application data.
        + application/json
* AUTHORIZATION :”Scheme Value”

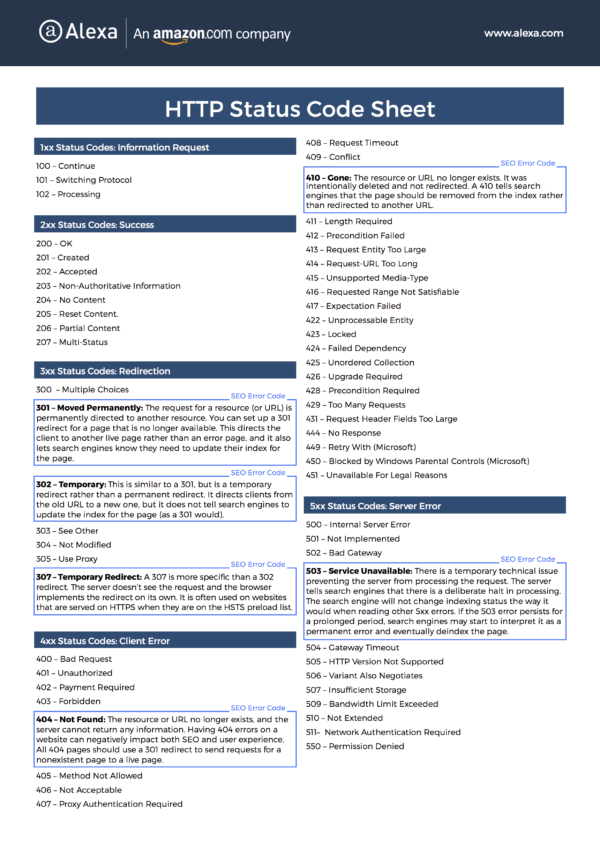
**Authorization: <type> <credentials>**

-Basic auth => “basic 989dfdsfdsfdf”

-Token auth => “Bearer tokenvalue”

* HTTP STATUS CODE

- 



API development does not have redirection.

**DAY 2**

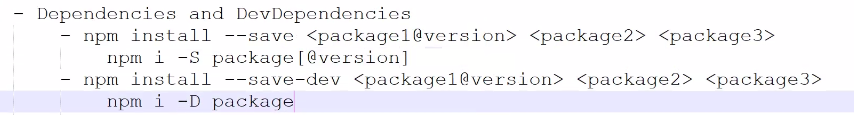
**BOOTSTRAP**

* Responsive UI library
* NgBootstrap can be used along with the angular
* Get bootstrap using
  + Source Files
  + CDN urls
  + Package managers – npm, nugget,yarn, bower etc

Dependencies are required in the dev dependencies and production dependencies.

Eg) Angular libraries are required in – dev and production

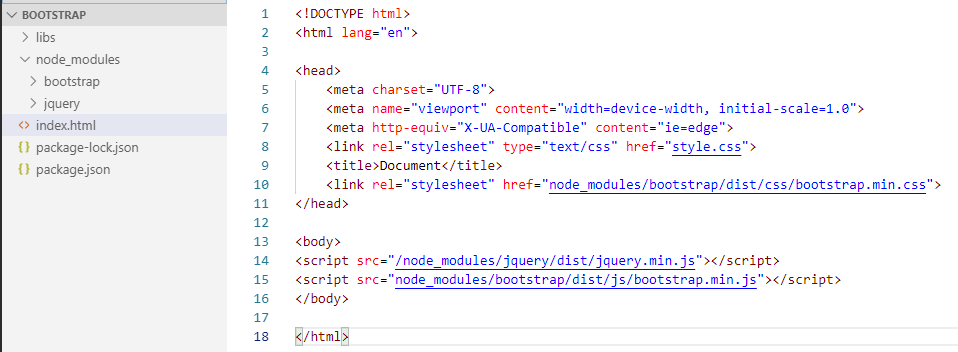
Testing libraries like karma are not necessary in the production machine



In order to install bootstrap .. u need to install jquery

**npm i -S jquery bootstrap**

you can either specify the version of the bootstrap like **npm i -S jquery bootstrap@3.4.1**



**JAVASCRIPT**

* It is the Interpreter
* It is not type safe
* It will not throw any compile time error

**TYPESCRIPT**

* It is superset of javascript
* It is the compiled language
* It is typesafe
* Fully compatible to ES6 features
* File extension is .ts
* We use “**tsc”**  for compiling the typescript file. Eg) **tsc main.ts**. This will change the typescript file to javascript
* Javascript interpreter is built inside the browser
* Run **node main.js**
* We can Control the compiler configuration using **tsconfig.json**
  + Set the target version of JS in tsconfig.json
  + Set whether source map file is required or not

[ when we compile the .ts file, then the file will be converted to the minified version of the javascript. Where it is difficult to debug. In order to debug ,we can map the typescript file in the sourceMap file]

* Install the typescript Globally
  + npm install –g typescript

PS C:\Users\Hvuser\Karthika\Day 2> **cd TypeScript**

PS C:\Users\Hvuser\Karthika\Day 2\TypeScript> **tsc index.ts**

PS C:\Users\Hvuser\Karthika\Day 2\TypeScript> **node index.js**

{"name":"Karthika","age":25,"dateOfJoin":"2019-08-21T18:30:00.000Z","salary":10000,"isMarried":false}

PS C:\Users\Hvuser\Karthika\Day 2\TypeScript**> tsc --init**

message TS6071: Successfully created a tsconfig.json file.

iIn the tsconfig.json file will have a STRICT,Then we need to declaration

**Interface:** Used to make sure the structure of the object

**Module:**

* A typescript file that contains a set of exported members
* A exported member can be class, interface ,function and constant.
* Export the members using **export** keyword and import them into another module using import keyword
* Barrel is a way of importing module into another files.
* A barrel use an index.ts file that exports a set of modules from a directory and it can be imported using a single line

**file.ts**

export function sum( a:number,b:number):number{

    return a+b;

}

export function substract( a:number,b:number):number{

    return a-b;

}

export function multiply( a:number,b:number):number{

    return a\*b;

}

**Main.ts**

import {sum,substract,multiply} from './file1'; (**OR)**

import \* as calc from './file1';

let a=5, b=4;

let total= calc.sum(a,b);

let diff= calc.substract(a,b);

let product= calc.multiply(a,b);

console.log("Add result is "+total);

console.log("difference result is "+diff);

console.log("product result is "+product);

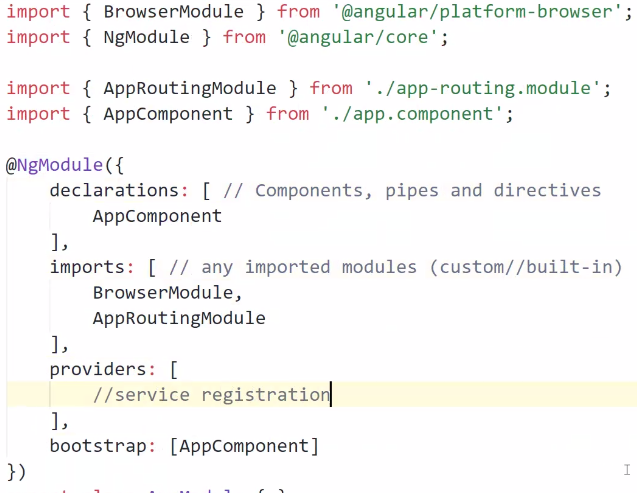
**ANGULAR**

-One html main page, others are partial page

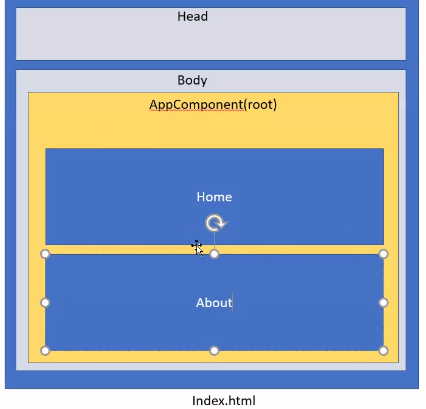
* We will be always on the index page, but the inner content will be changed w.r.t the partial html
* What is diff between Single page application/ multiple page application
* **SPA-** When u make the initial request , it will load the components/ template/html from the index.html from the server to the browser memory (client side) and not the database. It will make the request only through the rest API service.
  + **Lazy Loading-** the process of loading some features of your Angular application only when you navigate to their routes for the first time. This can be useful for increasing your app performance and decreasing the initial size of the bundle transmitted to the user's browser.
  + U can pre-render the angular template from the server side using **angular Universal**
* **MPA-** When u make the initial request , it will load the homepage from the server to the browser memory (client side). Every time when u navigate it will get the page from server.
* **WHY ANGULAR? -**  it is a framework. (React is just a library (.js) that can be downloaded and used in any application. It does not have any design pattern. It uses DOM Manipulation. No rendering, need more third party library for services etc )
* **Angular Cli is used for creating ,testing , building and running and deploying the application.**
* It have dependency injection
* It has different rendering engine (Ivy). Ivy is Angular's upcoming new rendering pipeline and view engine. ... Angular view engine takes the templates and components we've written and translates them into regular HTML and JavaScript that the browser can read and display. Ivy is the next generation of Angular Renderer.

**ANGULAR CLI**

* npm install –g @angular/cli
* ng --version [check the installed version]
* ng new <project name>
* e2e folder-> end to end testing ->protractor – used for testing
* tsconfig.json-> ts configuration
* package.json-> dependencies configuration
* angular.json-> src-source group, prefix:”app” == every component starts with “app”. It can also be empty
* **Angular module**
  + It encapsulates the
    - Components- UI of the application
      * Home.component.ts
      * Home.component.html
      * Home.component.css
      * Home.component.spec.ts ( unit testing purpose like KARMA)
    - Directives
      * Used to add/ remove behavior of a DOM element
      * Used to change the appearance
      * It is the attributres that can be used with the builtin tag/ custom tag
    - Pipes
      * Used to format the data while printing in the templete file. Eg) currency field- $ format.
    - Services
      * These are injectable objects ( angular Dependency Injection creates and manages the object)
      * Reusable function codes
    - Import submodules (builtin/ custom)
  + A module is a typescript class decorated with @NgModule()



* **Run the application**
  + - ng serve –o  **(or )** ng s –o
* **Creating component**
  + - ng generate component <components/ nameOftheComponent>
      * ng generate component components/home
    - ng g c components/home



**Directives**

* Structural directives – adding some additional HTML elements in the DOM
  + Starts with the **\* symbol**
  + \*ngFor, \*ngIf, \*ngSwitch
  + You cannot add 2 structural element in the single tag – This disadvantage can be replaced by “**ng-container**”

<ng-container \*ngFor="let item of products; index as i">

<tr \*ngIf="item.price<50" [ngClass]="{'higher': item.price>50, 'lower': item.price <=10 }">

    <td>{{i}}</td>

    <!-- <td>{{i+1}}</td> -->

                <!-- or -->

    <!-- <td>{{item.id}}</td> -->

    <td>{{item.Fname}}</td>

    <td>{{item.price}}</td>

    <td>{{item.quantity}}</td>

</tr>

</ng-container>

* Attribute directives- change the appearance and the behavior of the element – [ngStyle].
* Components- Directive with the template

**Bindings**

* **Interpolation –** {{variable}}
* **Property Binding –** input binding- [ ] -> ts file to html ---- code file to the template file
  + **[value]=”variable”**
  + **Event Binding** – output binding – ( ) -> html to ts file ----- template file to code file
    - (click)- “show()”
* **Two way Binding**- [()]- ngModel

**Pipe**

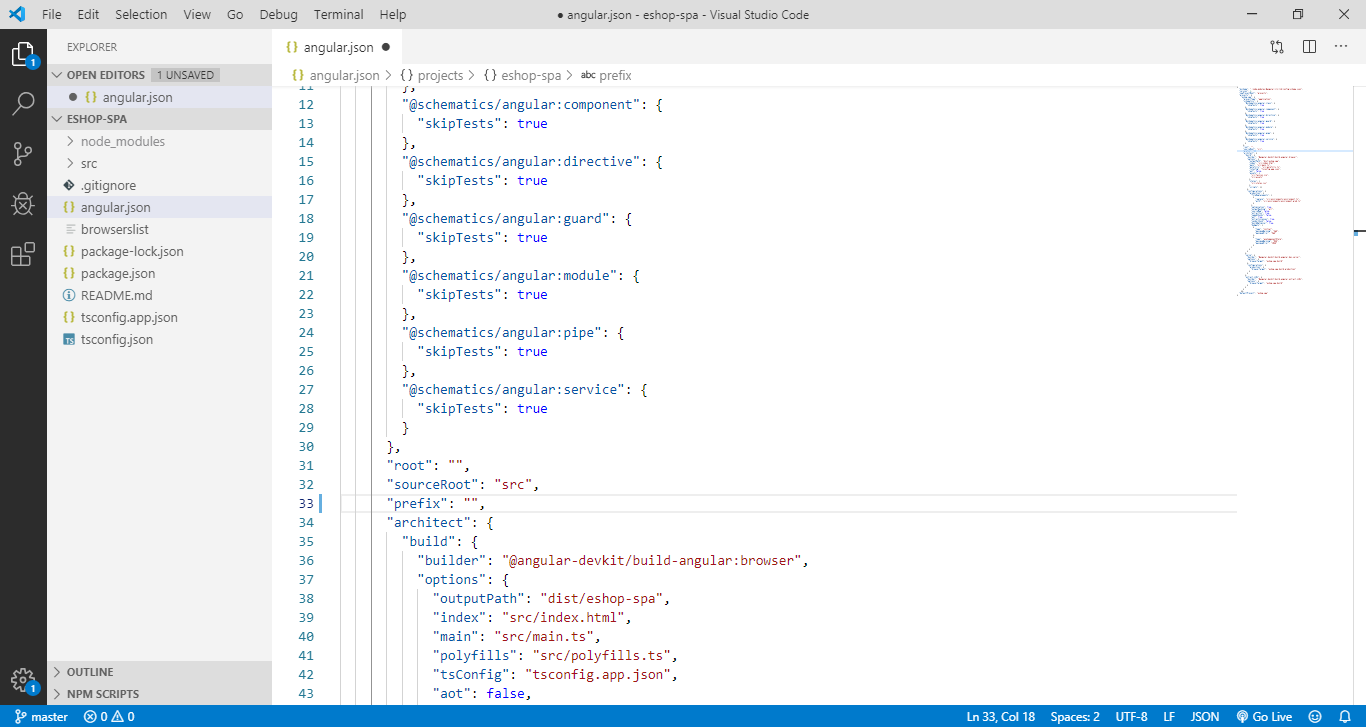
* Used for formatting data in templates
* It is applied using the pipe symbol - **|**
* **Pipe transform interface that takes the argument**
* Left side of the pipe symbol is the input value (arg) for the pipe function
  + Eg) {{ value | pipename [:args] }}
    - {{ name | uppercase}}
    - {{ today | date : ‘dd-MM-yyyy}}
    - {{ salary | currency : ‘INR’ : true : ‘5.2-3’}} - true means **symbol** will display
  + Value is the input value that u need to format,
  + Args are optional
* Builtin pipes
  + Upper case
  + Lower case
  + Title case
  + Json
  + Number
  + Currency
  + Date
  + Async
* Custom pipes
  + ng generate pipe [pipes/pipename]
  + ng g p pipes/pipenam

ANGULAR PROJECT

1. **Create new project** 
   * ng new <projectname> [options]

eg) ng new eshop-spa --minimal --skip-tests

*[we are just Ignoring the test files like e2e , karma*]

1. ****

Delete the prefix:”app” -> app needs to be deleted

1. Install jquery and bootstrap
   * Npm install --save jquery bootstrap
2. Create home and about component
   * ng g c components\about
   * ng g c components\home
3. Create Barrel for all the components, models, services
   * Under components create “index.ts”

export \* from './home/home.component'

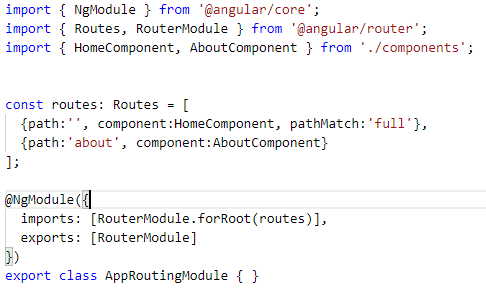
export \* from './about/about.component'

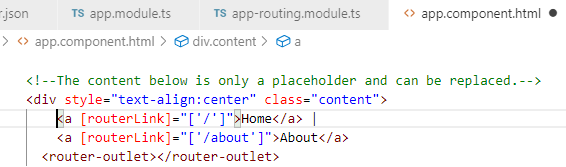
1. Create Routing

Lets see about Routing

It is array of Routes configuration.

Routermodule is the built in module. It provides the functionality of the servives, directives, pipe, components

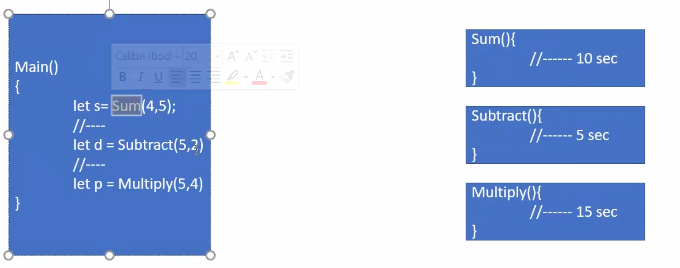




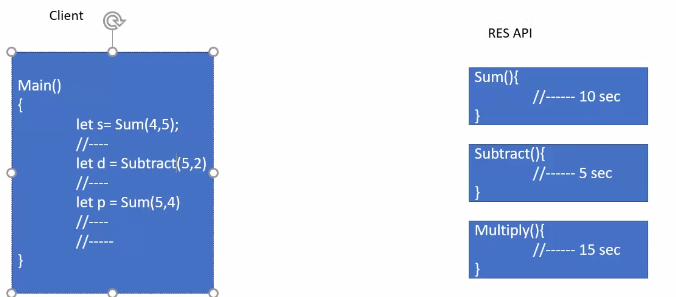
1. Create model
   * ng g interface models/category and
   * ng g interface models/products
2. Create Services
   * Ng generate service services/servicename
   * Eg) ng g s services/product
   * **@Injectable** is not necessary when u do not have dependency on other services
   * Providers are used for registering the services
   * Services can be registered globally and locally
     + Services can be registered globally ,then register in AppModule
     + Services can be registered locally , then register in individual component under **providers** .

**Note:**

**Observable and promises**



* Normally main thread will execute the below code -> one by one
* **Synchronous function – execution of the function one after another**
  + When main executes “Sum” -> “substract” function needs to wait 10 sec and “multiply” function needs to wait for 15 min.
  + When main executes “substract” -> “mutiply” function needs to wait 5 sec
* **ASynchronous function – execution of all function at a time using AJAX**



* + if the functions are not dependent with eachother, then u can execute any number function.
  + \*\* U cannot predict the time of execution
  + If u call a REST API function from the third party server for “sum” function. The execution speed depends on the network speed. U can use
  + **CALL BACK** is the function that execute when the asynchronous operation complete

**Eg) ASync in javascript**

* Async sum(a,b, callback);
  + 2 args- 2 variables
  + 3rd arg- call back – what to do with the result of the function,. When the call back will execute is not predicted
* **In angular , instead of Async we use Promises and observables**
  + Used for Asysnchronous programming
  + Promises
    - What is happening in the async can be tracked using promise
    - It is a built in feature of Javascript
    - **Completed successfully – resole**
    - **If there is any error – reject**
    - Once u make a request , you cannot cancel it .U will get the response
    - It will read the complete data , u cannot filter it

Function sum(a,b)

{

Return new promise((resole,reject)=>{

//total a+b

If(completed)

Resolve(total);

Else

Reject(err);

});

}

Let total=0;

//promise based Async

*Let promis= Sum(1,2);*

*promis.then( successCallback, errorCallback); (OR)*

Sum(1,2).then( successCallback, errorCallback);

Function successCallback(result){

//do something with the sum result

}

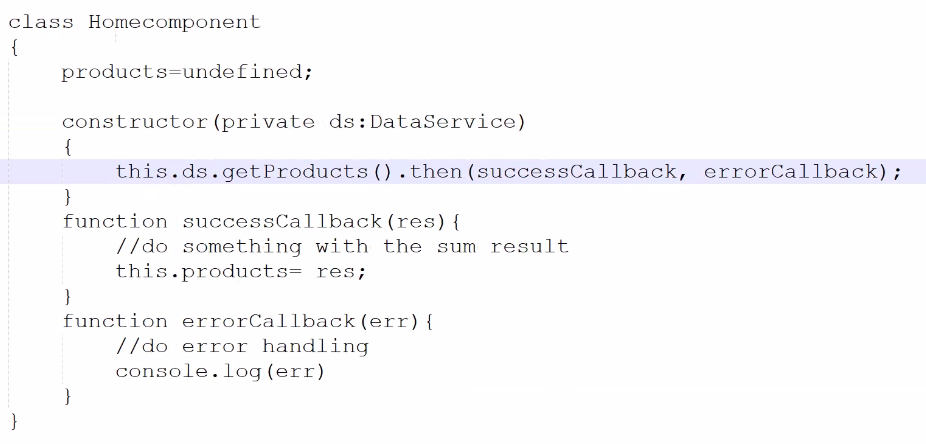
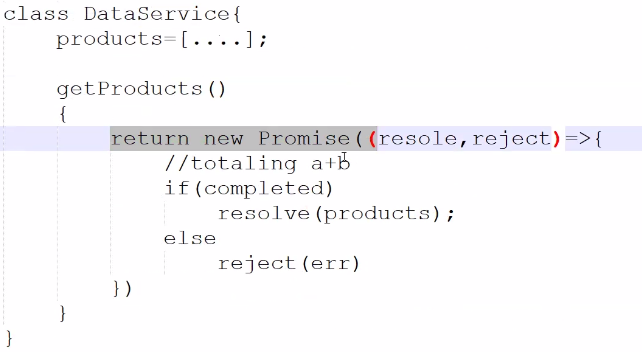
Function errorCallback(err)

{

//do error handling

}

**In real time**



**Disadvantage of Promises:**

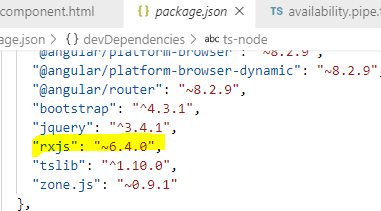
When u call the getproduct() that loads constructor, u get the result of 5 product. But when the product increased from 5 to 10 . Then u will not get any real time updation in the “Promise”. To get the updated product , u need to again call getproduct() because it is the **Request-response Pattern.**

So we go for **observables**

**OBSERVABLES**

It is a real time data updation like using SignalR. It will automatically updated using “**Subscribe**”**. Eg) Youtube Subscription .**Once u subscribe , the data will be updated from the server automatically . This process will continue until unsubscribe.

This is called **Publish Subscribe pattern.**



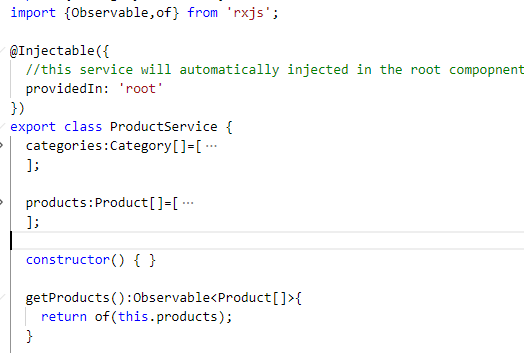
* We need to install **RxJS, Rx.net ,ReactiveX– for using observable – It is a third party lib**
* Once u make a request , you can cancel it .
* We have lot of operators, eg)In dotnet Linq we have “select”, which writes fewer data.
  + Map operator
  + Skip and take
  + Pagination
  + Transforming the data
  + Filtering data
  + Request- server is busy. U can use “retrieve “ operator.
    - Eg) when u send the request with retrieve(3), even though the server is busy . The same request can be sent 3 times to get the data. It only throws the exception when retrieve (3) exceeds 3 request.

Async method either use promise / observable

* Async pipe – it will wait until the observable function complete.It is used to resolve the observable object. ngFor will wait until the **async**  function complete.

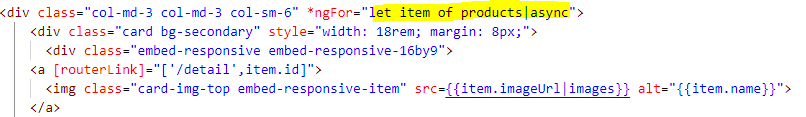
**APPROACH 1 for Observable**

**Product.service.ts**



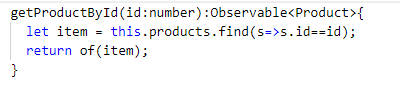
**Home.component.ts**



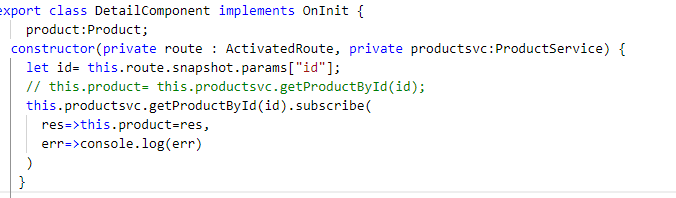
[**Home.component.html**](http://Home.component.html#)

**APPROACH 2 for Observable**

**Product.service.ts**



**Detail.component.ts**



**FORMS in Angular**

Two approaches for the form creation: Both the form will do same operation like validation and submission

* Template driven form-
  + Import FormsModule into AppModule from ‘@angular/forms’ package
  + All validations are done in the HTML file
  + We use HTML 5 validation attributes
  + Need to use ngModel directives for two way binding
  + Every form control must have a name attribute if we use ngModel
  + A reference variable can also be used to access the form control.
  + Every control have some properties that defines the state
    - Valid- invalid
    - Dirty-pristine
      * Dirty -(eg)text box,radiobutton etc – u have typed something
      * pristine (eg)text box – u have not typed anything
    - Touched-untouched
      * Touched -(eg)text box – u have touched the textbox eventhough u didn’t type anything.
      * untouched -(eg)text box – u have not touched the textbox eventhough u didn’t type anything.
* Forms also have valid and invalid properties

 <p>Valid: {{form.valid}}</p>

            <p>InValid: {{form.invalid}}</p>

            <div class="form-group">

                <label for="name">Name</label>

                <input type="text" name="name" class="form-control" [(ngModel)]="product.name" #name="ngModel" required/>

                <div>Prisine: {{name.pristine}}</div>

                <div>dirty: {{name.dirty}}</div>

                <div>Touched: {{name.touched}}</div>

                <div>UnTouched: {{name.untouched}}</div>

                <div>valid: {{name.valid}}</div>

                <div>InValid: {{name.invalid}}</div>

                <div>Errors: {{name.errors|json}}</div>

            </div>

* Reactive Form
  + We use ReactiveFormModule from ‘@angular/forms’
  + We use services like FormBuilder to create FormGroup
  + A formGroup is a class used to create model for the form
  + We need to use the directives like FormGroup and formControlName
  + Validations are done in the model object (FormGroup) using Validation Class
  + This is a **model driven approach**
  + Forms and control properties are applicable here also.