Tata Consultancy Service

International Tech Park,  
Pioneer Building,  
Padandhur Agrahara,  
Whitefield Road,  
Bangalore 560 066

Victor and Pioneer buildings at ITPL

1.Abilash software development centre

2.Vydehi vydehi bus stop to Hope farm

3.Global Axis

4.Victor and Pioneer buildings at ITPL

5.Brigade Bhuwalka Icon

6.G R Techpark (GRT)

International Tech Park commonly called ITPL

and ITPB is a tech park located in Whitefield

I believe on learn and grow.

As a job holder I can learn if there is a purpose. That purpose comes when you are facing new challengs,new things in a job,now in my company I was lagging those, apart from those I don’t have any issue. Good collage. Good relation with manager entire organization .

weeknes

I work for exact results i.e perfect result for that I may take time.

An architect from 3yrs now.

Pressur-I look it into tight time bound notging more than that.

Angular Syntaxes

Interpolation

{{product.name}}

{{Quantity \* price}}

{{Get Total()}}

Src={{product.photo}}

Property binding

[src]=”product.photo”

One-Way binding

<input # productName >

<select #refName>

{{ productName.value}}

Two-Way binding

Two-way binding was the combination of 1.Event Binding+2.Property Binding

ngModel—for id,class—like properties

<input [(ngModel)]=”productName”>

<select [(ngModel)]=”Shipped”>

{{productName}} -> prient the value

{{Shipped}}

[]- Property Binding

()- Event Binding

{{}}- Attribute Binding

# ….{{}}- one way binding

[(ngModel)]- Two way binding

Class-Binding

<h2 [ngClass]=”string”>

<h2 [ngClass]=”[‘array1’,array2’]”>

<h2 [ngClass]=”{‘JSON1’,JSON2’}”>

In CSS

.string{},

.array1{}

.JSON{}

Event-Binding

<button (eventName)=”method($eventArgs)>

<button (click)=”insert()”> public insert()

<button (click)=”update()”> public update()

Handling Object Property (DATA Base)

Public Action(val) case:”insert”

<button #sendInsert value=”insert” (click)=”Action(sendInsert.value)”

Angular Conditions

if,else,switch,case,then

<div \*ngIf=”booleanValue”>

ADD a div to page if “true”

REMOVE a div to page if “false”

Angular Iteration

For Repetation

For,while,do while,of,in etc

Odd,even,first,last,trackby

<li \*ngFor=”let item of collection”>

Data in JSON format Nested

Iteration to generate elements dynamically

Public collection=

[{category: ’Electronics’,

Devices: [‘mobile’,’tv’,’laptop’]}]

[Class.odd]=”odd” [Class.even]=”even”

Condition with Multiple Elements

\*ngIf=”BooleanCondition;

then element;

else element”

\*ngIf=”isError; then isTrue else isFalse”

Will not allow Multiple templates to bind with an element

<ng-template #isTrue>

<ng-template #isFalse>

<ng-template > is used for ngIf, ngFor, ngSwitch

Style Binding

Inline

Dynamically-- [ngStyle]=”getEffects()”

Mouse events Binding

Two methods on one div like mousedown and mouseup

<body (mousemove)=”ApplyEffects($event)”>

<img [src]=”pic” (mouseover)=”Ad1()” (mouseout)=”Ad2()”> Rerliance, pepsi

<img [src]=”bulb” (mousedown)=”down()” (mouseup)=”up()”>

Key events Binding

<input (keypress)=”getkeycode($event)”>

this.keycode=event.keycode; {{keycode}}

this.charcode=event.charcode; {{charcode}}

this.which=event.which; {{which}}

this.shift=event.shift; {{shift}}

Angular pipes

Pies required to transform the data when received from 3rd party-source.

1.Upper &lowercase

Public msg=”welcome to angular”;

<p>{{msg|uppercase}}</p>

2.Number Pipe

num1: number = 12.638467846;

<p> {{num1 | number}} </p>

<p> {{num1 | number:'3.2-5'}} </p>

o/p

12.638

012.63847

Angular supports : upto 29decimals

3.Currency

{{data|currency:’currencySymbol”:booleanValue}}

Public price=450.60;

<p>{{price|currency}}</p> $450.60

<p>{{price|currency:”INR”:true}}</p>

₹450.60 True means globalization format

<p>{{price|currency:”Rs”:false}}</p> Rs:450.60 false- will present the symbol exactly as defined

4.Date

{{date|date:”shortDate”}}

{{date|date:”longTime”}}

{{date|date:”MM/dd/yyyy”}}

5.percent pipe

{{number:percent}}

6.Slice

Angular filter

{{collection | slice: StartIndex: endIndex}}

7.JSON pipe

Convert Data into JSON format

Public products=[{}]

<div>{{products | json}}

8.Custom pipes

Pipe- to configure meta-data for pipe

Pipe Transform-to configure a function for pipe.

Import {pipe,pipe Transform} from “@angular/core”;

String Formate Functions()

String.charAt(),

String.indexOf(),

String.lastIndexOf(),

String.substring(0,4), if(4) //4 to end

String.toLowercase(), .toUppercase(), .length(),.fontsize().

Accesing Data of 1component to Another

@input() public property:type = value;

@input() public electronicsCount = 0;

Import {component, input} from ‘@angular/core’;

Now custom Events

@Output() public selectionChanged:

EventEmitter <string>= new EventEmitter <string>();

onSelectionChanged(){

this.selectionChanged.emit

(this.selectedCategoryValue);

<app-filter (selectionChanged)="onSelectionChanged($event)"

+++==== remaining topics

Angular Services

1.A service is predefined business logic that you can inject and use in any component.

2.Service is a collection of Factories.

3.Factories is a collection of functions()

4.Service reprecent SingleTone design pattern, where an object is created only once and the same object will be used across any number of requests.

5.Angular uses asynchronous mechanisam to access and inject any service into components.

**Mongo**

1.Mongosb is a free open source cross platform document oriented database program.

2.it is a non-RDBMS and Non-SQL database.

3.It is schema-less and support features like

1.Ad-hoc queries,2.high-avialbulity,3.lazy-loding,4.data-Replication.

db.tblproducts.find({“price”:{$gt:40000}})

db.tblproducts.find({“name”:”shoe”}).pretty()

Built in Objects for API

1.Request object: client-server =>to Data Base

2.Response object: Data Base-server=>to Client

3.body-Parser: Data is **encoded** from **URL** and Parsed into **JSON**

app.use (body-paser.urlEncoded({

extended:true;

});

app.use(body-paser.json());

Restful services in Angular

In angular every call is a AJAX call

In Javascript:

Var http = new XmlHttp Request();

http.load(url);

In JQuery:

$get JSON(url);

In Angular:

1.$http.get(“url”).then().success();

**2.http module** upto ANGULAR-5,

**3.httpClientModule** from Angular-6,7

4.httpClientModule uses “RXJS” for making an AjAx call.

5.Angular component uses typescript,which is strictly typed hence component need an observable that can define the type of data used by server..

What type of Data is coming ?

Observable(Iproduct[]) 🡺 rxjs

(Ajax javascript library).

Observable =>identify what data is coming.

Angular Routing

Routing is a technique used in web-application to create User-friendly Url’s and SEO-friendly Url’s.

1. A user-friendly URL’s allows to query-any content directly from URLs

2. A SEO-Friendly URL’s allows to identify the exact location in application and recommended the relative content.

Routes :- It provides a set of properties and methods that are used to CONFIGURE routes for your application.

Routes Module:- It provides a set of properties and methods that are used to ENABLE routes to your application.

3.Properties of angular

1. Path , 2.Component , 3.redirectedTo , 4.data,5. pathmatch, 6.Outlet.

Handling Routes in UI

The configuration for navigating to any specific Route is defined by

<a routlink=”routePath”> Text </a>

<router-outlet> </router-outlet>

Import {NgModule}

Import {routes,routerModule}

Const routes : Routes=[ ];

@NgModule({import:[RouterModule.forRoot(routes)],exports:[RouterModule]})

Route Parameters

http work with mechanism “GO-Get-Forget”

Go--means establish connection with server

Get--get response from server

Forget--Remove the response details from server.

In http by using Query string 1-page information can access to another page.

In routing Techniques use Rout-parameters instead of query string to access 1-page information to another page.

1.Route parameters configured in Route-path

{path: ”products/:id/:name,….”}

2.Route parameters configured in URL

Products/1/mobile

3.Route parameters can access in any component by using ActiveRoute module.

Private routes: ActiveRoutes;

Routes.snapshot.paramMap.get(“id”)

4.we can navigate any routes-manually by using routes module

Private routes:router;

this.routes.navigate( [‘path’], params);

Children attribute, Navigation methods

{ path:’name’, component:name,

**Children**=[path:’name’, component:name]}

Navigate to the child path

this**.router.navigate([‘preview’],**

{relatedTo:this.route});

Test driven Development(TDD)

TDD is a process of 1.developing and 2.testing application simultaneously.

Angular uses MVC-framework that supports TDD.

TDD used only when your application support

1.code reusability

2.code separation

3.Maintainabilaity

It enable unit testing where each and every function is tested.

Angular Material and Animations

Material is open source library for Angular.

Material provides various components for designing UI in angular.

Material uses Browser animations with CSS effects.

Modules use like

1.BrowserAnimationModule

2. MatFormFieldModule

3.MatSelectModule

Angular Component Life Cycle Events

<https://dzone.com/articles/angular-6-part-3-life-cycle-of-a-component#:~:text=In%20Angular%2C%20every%20component%20has,obtain%20control%20of%20the%20components>.

The component phases **use various** **Events** to handle interactions from begin to End.

**ngOnChange**

1.It is the component event that fires-up after initialization.

2.It **identifies and Update** the changes happened in component

3.The base class “SimpleChange” can track the changes Before and After.

**ngOnInit:-**

It is the initialization phase of component that fires Up when a **component is request in application**. It initialize the default value into component.

**ngDOcheck**:-

It is the component event that fires up exactly **when angular is unable to track the changes implicitly.**

ngAfterContentInit.

ngAfterContentChecked.

ngAfterViewInit.

ngAfterViewChecked.

**ngOnDestroy:-**

It is a component event that defines functionality **when the component is closed,** usually **it performs cleanup as a destructors**.

…………………………………………………

onchange

<https://www.concretepage.com/angular-2/angular-2-4-onchanges-simplechanges-example>

Now whenever parent component changes value in any of its property that has been used in child component, then in the child component the ngOnChanges() method runs. It works for any primitive data type such as string, number etc.

Routing:

Routing will provide navigation from one component/module to another

 {

        path:'vendor',

loadChildren:'./customer/customer.module#CustomerModule',

component:RegisterComponent,

canactive:

children:[{components we can declare}]—used for –provides lazy –loading

    },

Activatedroute.params.keyname

Activatedroute.params.map---for array of keys

Async await

async function firstAsync() {  
let promise = new Promise((res, rej) => {  
setTimeout(() => res("Now it's done!"), 1000)  
});  
  
// wait until the promise returns us a value  
let result = await promise;   
  
// "Now it's done!"  
alert(result);   
}  
};

function f1(n){

return new Promise(function(a,b){

if(n%2 == 0){

a('even')

}else{

b('odd');

}

})

}

f1(10)

.then(

function(r){

alert(r);

},

function(r){

alert(r);

})

## Types of Directives

<https://www.codementor.io/christiannwamba/build-custom-directives-in-angular-2-jlqrk7dpw>

Angular 2 categorizes directives into 3 parts:

1. Directives with templates known as **Components**
2. Directives that creates and destroys DOM elements known as **Structural Directives**

\*ngIf \*ngFor and [ngSwitch]

1. Directives that manipulate DOM by changing behavior and appearance known as **Attribute Directives**

**NgStyle, NgClass,**

<p [style.color]="'blue'">Directives are awesome</p>

<p [hidden]="shouldHide">Directives are awesome</p>

Forms

1.Template driven forms:

2.Reactive /model driven forms:

Types of Validation classes

1.Input state validation classes

2.Form state validation classes.

3.custum validations

1.Template driven forms:

Template driven forms are not strongly bound to data hence it recommended to use model driven forms.

1.Input state validation classes

**Identify error type:**

An **input element** can be defined with **multiple** validations to identify specific validation error.we can use **error** **property** it verify the **specified error** type and **return** Boolean true or false.

<input type=text #uname=’ngmodel’>

{{uname.errors.required}}

In reactive forms we do have setValue() and patch values

in this blog, I will explain the difference between setValue() and patchValue() which are frequently used in Angular2 reactive forms.

Updating Form controls from a model is very easy and flexible in Reactive Form API of Angular v2. So, setValue() and patchValue() are the methods by which we can update the Reactive forms controls from our model. Let us see the difference between these two in the following example.

**setValue()**

setValue() will set the value of all form controls of the form group.

1. **this**.profileForm = **new** FormGroup({
3. FirstName: **new** FormControl(''),
4. LastName: **new** FormControl(''),
5. UserName: **new** FormControl(''),
6. Email: **new** FormControl(''),
7. MobileNumber: **new** FormControl('')
9. })

We have the above FormGroup controls. Now, we will use setValue() methods to update all FormControl values.

1. profileForm.setValue({
2. "FirstName":"Ajay",
3. "LastName":"Panda",
4. "UserName":"ajay33",
5. "Email":"ajay.panda@gmail.com",
6. "MobileNumber":"8745212589"
7. });

As you can see from the above code, setValue() method will set all fromcontrol values from model. If you do not mention any of the formcontrol values in model, then it will throw an exception.

**patchValue()**patchValue() method will also set Formcontrol values from a model but only for those which we have mentioned in the model. So,  it is not necessary to pass all the model info in patchValue() method.

1. profileForm.patchValue({
2. "FirstName":"Ajay",
3. "LastName":"Panda"
4. });

As you can see from the above code, I am only passing two properties to my formgroup control, so patchValue() will set these two properties in form. Unlike to setValue(), it is not necessary to pass all models, we can only pass required models to the from group control.

If out of all 6 validations if not satisfied error property will identify.

1.Ngtouched,

2.nguntouched,

3.ngpristine,

4.ngdirty,

5.ngvalid,

6.nginvalid.

**2.Form state validation[entire form]**

ngpristine,ngdirty,ngvalid,nginvalid.

<form #formname=’ngForm’>

{{formname.form.invalid(validation className)}}

2.Reactive /model driven forms:

Import{ formGroup, formControl ,validators} form “@angular/forms”

Reactive form allows dynamically to create a formGroup, formControl.i.e in fom control(validation.required,validation.pattern can be created dynamically(by us).

formGroup is for Form-Element(i.e group of controls) and FormControl is for input element text-box, radio-button,Droupdown etc.

Cors

**CORS** essentially means cross-domain requests. Simply using this line of code to set a header on your response **will** enable **CORS**. res.header("Access-Control-Allow-Origin", "\*"); This code snippet, however, **would** enable **CORS** for all resources on your server.

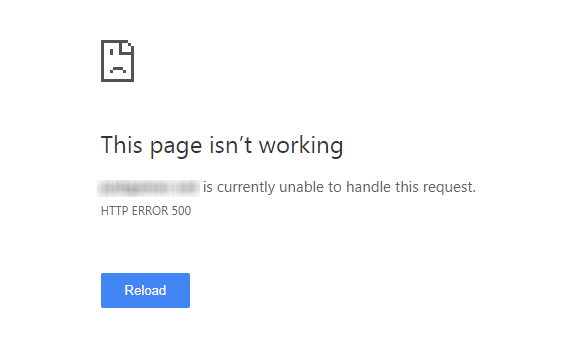
This post shows how to enable Cross Origin Resource Sharing (CORS) in Node. CORS essentially means cross-domain requests.

Simply using this line of code to set a header on your response will enable CORS.

res.header("Access-Control-Allow-Origin", "\*");

This code snippet, however, would enable CORS for all resources on your server.

The **HTTP 404**, **404 Not Found**, **404**, **Page Not Found**, or **Server Not Found** [error message](https://en.wikipedia.org/wiki/Error_message) is a [Hypertext Transfer Protocol](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) (HTTP) [standard response code](https://en.wikipedia.org/wiki/List_of_HTTP_status_codes), in computer network communications, to indicate that the [browser](https://en.wikipedia.org/wiki/Web_browser) was able to communicate with a given [server](https://en.wikipedia.org/wiki/Server_(computing)), but the server could not find what was requested. Further, when the requested information is found but access is not granted, the server may return a 404 error if it wishes to not disclose this information, as well.[[1]](https://en.wikipedia.org/wiki/HTTP_404#cite_note-1)



The 500 Internal Server Error is a very general HTTP status code that means something has gone wrong on the web site's server but the server could not be more specific on what the exact problem is.  
  
**Resolution Possibilities**  
  
The 500 Internal Server Error is a "server-side" error, meaning the problem is not with your PC or Internet connection but instead is a problem with the web site's server.  
Even though the issue is not yours to troubleshoot or resolve, there are a few things you can do:

What is nodemodule

Node module will read what all package.json conrent.

It is confidendial to my company. publically it is not available.

Difference b/w constructor

Non life cycle and ngonit

Routing----------- router-link

Activated routes and route status

Attribute directive

What is dependency injection

Custom pipes exmle

How you will define route.

How to chage image sige according to screen size

How you will use difffernt components in one component

Routes, routerlink, <router-outlet></router-outlet>

**Angularjs vs angular**

<https://gorrion.io/blog/angularjs-vs-angular>

About **project code**

**Command ng new projectName**

import {RouterModule,Routes} from '@angular/router';

import {RegisterComponent} from './shared/components/register/register.component'

const routes:Routes=[

    {

        path:'',

        redirectTo:'login',

        pathMatch:'full'

    },

    {

        path:'login',

        component:LoginComponent

    },

 {

        path:'customer',

        loadChildren:'./customer/customer.module#CustomerModule'

    }

]

export const appRouting=RouterModule.forRoot(routes,{useHash:true})

const routes:Routes=[

    {

        path:'',

        component:AdminComponent,

        children:[

            {path:'', redirectTo:'home',pathMatch:'full'},

            {path:'home',component:HomeComponent},

            {path:'vendor-reg',component:VendorRegComponent},

            {path:'vendors-list',component:VendorsListComponent},

                    ]

    }

]

export const adminRouting=RouterModule.forChild(routes);

this.router.navigateByUrl('admin/vendor-update/'+userObj.\_id)

<router-outlet></router-outlet>

this.router.navigateByUrl(role);

……………………………………………………………………X

How to run single component

Index.html

<body>

  <app-root></app-root>

</body>

App.module.ts

bootstrap: [AppComponent]

1.import {Router} from '@angular/router'

2.constructor(private router:Router)

3.this.router.navigateByUrl('customer/success')

…………………………………………………………..X

**Shared service**

import {SharedService} from '../shared/services/shared.service'

constructor(private shared:SharedService) { }

 fnLogout(){

    this.shared.fnLogout();

  }

……………………………………………………X

<app-table [hasImage]='false' [headers]="headers"  [data]="data" [keys]="keys" (editInfo)="fnEditInfo($event)"

  (deleteInfo)="fnDeleteInfo($event)"></app-table>

…………………………………………………………………X

<div class="message text-center" **[ngStyle]="{'background':clr}"** >

……………………………………………………………X

@Input() msg ;

 @Input() clr;

 @Output() fnCloseMsg= new EventEmitter();

  fnClose(){

    this.fnCloseMsg.emit();

  }

…………………………………………………X

dataObj: any = {

    'uid': ''

  };

[(ngModel)]="dataObj.uid"

[(ngModel)]="dataObj.pwd"

[(ngModel)]="dataObj.email"

[(ngModel)]="dataObj.phone"

(res) => {

        if (insertedCount == 1 && ok == 1 && n == 1)

………………………………………………………………………………………………………X

export class SharedService {

  msgSubject=new Subject();

  maskSubject=new Subject();

}

Other service file………….

this.shared.maskSubject.next(true);

…………………………………………………………………………………………..

Intercept

import {

  HttpRequest,

  HttpHandler,

  HttpEvent,

  HttpInterceptor,

  HttpResponse,

  HttpErrorResponse

} from "@angular/common/http";

intercept(

**request**: HttpRequest<any>,

**next**: HttpHandler

  ): Observable<HttpEvent<any>> {

**request**.clone({

        headers: **request**.headers.set("Authorization", window.sessionStorage.token)

return **next**.handle(updatedRequest).pipe(

error => {

          //logging the httpResponse to browser's console in case of a failuer

          if (event instanceof HttpResponse) {

            console.log("api call error :", event);

…………………………………………………………………………………X

@ViewChildren(Comp4childComponent)

private compfour:QueryList<Comp4childComponent>=new  QueryList();

private usarray:Array<any>=[];

ngAfterViewInit(){

        this.usarray=this.compfour.toArray();

}

<p>

<app-comp4child></app-comp4child></p>

<app-comp4child></app-comp4child>

<app-comp4child></app-comp4child>

<button  (click)="clickMe()">clkviewchildren</button>

……………………………………………………………………………..X

<Url:'users/get-user-det?id='+window.sessionStorage.id>;

………………………………………………………………………….X

<div \*ngIf='isShow'>

    <div **ngSwitch**="{{display}}">

       <div \*n**gSwitchCase**="'true'" [ngStyle]="{'background-color':'green'}">login succesfull</div>

       <div \***ngSwitchCase**="'false'" [ngStyle]="{'background-color':'red'}">

                            Please check your enterd uid or pwd

   <span><button matbutton (click)='fnClose()' [ngStyle]="{'fontsize': '20px', 'fontweight': 'bold'}">X</button> </span>

     </div>

…………………………………………………………………………………..X

constructor(private **http:HttpClient**) {

    debugger

**var dataUrl=this.http**.**get**('https://jsonplaceholder.typicode.com/posts');

        dataUrl.**subscribe((data**)=>{

……………………………………………………………..

Angular 4

* Else block in \*ngIf introduced:  
  — Instead of writing 2 ngIf for else , simply add below code in component template:

\*ngIf=”yourCondition; **else myFalsyTemplate**”  
“<ng-template #myFalsyTemplate>Else Html</ng-template>”

Angular 5

* Preserve White space: To remove unnecessary new lines, tabs and white spaces we can add below code(decrease bundle size)

// in component decorator you can now add:  
“preserveWhitespaces: false”  
// or in tsconfig.json:  
“angularCompilerOptions”: { “preserveWhitespaces”: false}`

* HttpClient: until Angualar 4.3 [@angular/HTTP](http://twitter.com/angular/HTTP) was been used which is now depreciated and in Angular 5 a new module called HttpClientModule is introduced which comes under [@angular/common](http://twitter.com/angular/common)/http package.
* Few new Router Life-cycle Events being added in Angular 5: In Angular 5 few new life cycle events being added to the router and those are:

ActivationStart, ActivationEnd, ChildActivationStart, ChildActivationEnd, GuardsCheckStart, GuardsCheckEnd, ResolveStart and ResolveEnd.

* Angular 5 supports TypeScript 2.3 version.
* Improved in faster Compiler support:  
  A huge improvement made in an Angular compiler to make the development build faster. We can now take advantage of by running the below command in our development terminal window to make the build faster.  
  ng serve/s — aot

…………………………………………………………………………………….X

Angular 6

* Remove support for *<template>* tag and “*<ng-template>*” should be used.
* Registering provider: To register new service/provider, we import Service into module and then inject in provider array. e.g:

// app.module.ts **import** {MyService} **from** './my-service';  
...  
providers: [...MyService]  
...

But after this upgrade you will be able to add providedIn property in injectable decorator. e.g:

// MyService.ts@Injectable({ providedIn: 'root'})  
export class MyService{}

CLI Changes: Two new commands have been introduced  
— **ng update** <package>  
\* Analyse package.json and recommend updates to your application  
\* 3rd parties can provide update scripts using schematics  
\* automatically update code for breaking changes  
\* staying update and low maintenance  
**— ng add**  
\* add new capablities to your applicaiton  
\* e.g ng add [@angular/material](http://twitter.com/angular/material) : behind the scene it add bit of necessary code and changes project where needed to add it the thing we just told it to add.  
\* Now adding things like angular material, progressive web app, service workers & angular elements to your existing ng application will be easy.

Difference B/w Constructor and NgOnInit

<https://medium.com/javascript-in-plain-english/difference-between-constructor-and-ngoninit-in-angular-537ecfa6ce1e>

Constructor

* The **Constructor** is a default method of the class that is executed when the class is instantiated.
* Constructor ensures proper initialization of fields (class members) in the class and its subclasses.
* Angular Dependency Injector (DI) analyse the constructor parameters.
* When we call new MyClass() it creates a new instance of the class.
* While calling new MyClass() we must have to pass the exact match of the parameters type to the constructor of the class, example:  
  new MyClass(arg1:number, arg2:string, argN:any)
* These arg1:number, arg2:string, argN:any, must be of same type defined in constructor of class MyClass.

# ngOnInit

* **ngOnInit** is a life cycle hook called by Angular to indicate that the Angular is done creating the component.
* In order to use OnInit we have to import it in the component class like this:  
  import {Component, OnInit} from '@angular/core';
* Actually implementing OnInit in every component is not mandatory. But considered good practice.
* A class implements OnInit like this:  
  export class AppComponent implements OnInit { }

# Using ngOnInit

**Angular** calls its **[ngOnChanges](https://medium.com/@AnkitMaheshwariIn/angular-project-with-lifecycle-hooks-understand-ngoninit-in-depth-b9919ad09e6" \t "_blank)**[()](https://medium.com/@AnkitMaheshwariIn/angular-project-with-lifecycle-hooks-understand-ngoninit-in-depth-b9919ad09e6" \t "_blank) method whenever it detects changes to the value of input properties of the component (or directive).

Learn more about **[ngOnChanges](https://medium.com/@AnkitMaheshwariIn/angular-project-with-lifecycle-hooks-understand-ngoninit-in-depth-b9919ad09e6" \t "_blank)**[()](https://medium.com/@AnkitMaheshwariIn/angular-project-with-lifecycle-hooks-understand-ngoninit-in-depth-b9919ad09e6" \t "_blank) in [**Angular Project with Lifecycle Hooks**](https://medium.com/@AnkitMaheshwariIn/angular-project-with-lifecycle-hooks-understand-ngoninit-in-depth-b9919ad09e6)**.**

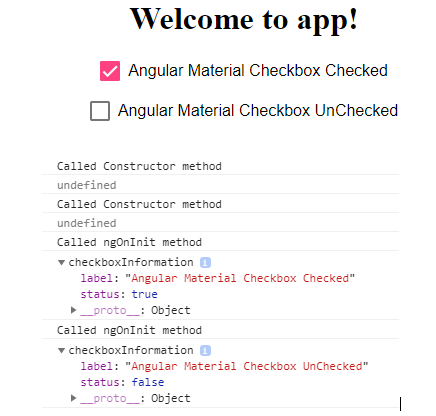
# Difference between ngOnInit() and constructor()

* We use constructor() for all the initialization/declaration.
* It’s better to avoid writing actual work in the constructor.
* The constructor() should only be used to initialize class members but shouldn't do actual "work".
* So we should use constructor() to setup Dependency Injection, Initialization of class fields etc.
* ngOnInit() is a better place to write "actual work code" that we need to execute as soon as the class is instantiated.
* Like**loading data**from Database — to show the user in your HTML template view. **Such code should be written in ngOnInit()**.

# Conclusion

* **Constructor** initialize class members.
* **ngOnInit()**is a place to put the code that we need to execute at very first as soon as the class is instantiated.

<https://www.angularjswiki.com/angular/what-is-the-difference-between-constructor-and-ngoninit-in-angular/>



ngSwitch case implement

ngSwith **object** can added

<span [ngSwitch]="event.format">

      <span \*ngSwitchCase="'InPerson'" class="label label-warning">In-Person</span>

      <span \*ngSwitchCase="'Online'" class="label label-warning">Online</span>

      <span \*ngSwitchDefault class="label label-warning">TBD</span>

    </span>

ngClass case implement

**event**() can added to

1.HTML

<h2 [ngClass]="getTitleClass(event.format)">{{event.name}}</h2>

2.ts.file

 getTitleClass(eventFormat) {

    if (eventFormat === 'InPerson')

      return ['in-person']

    if (eventFormat === 'Online')

      return ['online']

    return ['tbd']

}

3.css file

.in-person { color: green; }

    .online { color: red; }

    .tbd { color: #aaa; }

How to call a method from one component to another

<https://www.itsolutionstuff.com/post/angular-call-component-method-from-another-component-exampleexample.html>

@Component({

selector: 'app-comp-two',

**template**: `<div>

<p>Call Component Two</p>

<button (click)="one.myFunctionOne()" >Call Component One Method</button>

<app-comp-one #one></app-comp-one>

</div>`,

})

2nd method

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

**import** { CompOneComponent } **from** './compOne.component';

@Component({

selector: 'app-comp-two',

**template**: `<div>

<p>Call Component Two</p>

</div>`,

})

**export** **class** CompTwoComponent **implements** OnInit {

constructor() { }

ngOnInit() {

**let** myCompOneObj = **new** CompOneComponent();

myCompOneObj.myFunctionOne();

}

}

……………………………………………………………

npm uninstall -g @angular-cli

npm install -g @angular/cli@latest

package.json

"scripts": {

    "ng": "ng",

    "start": "ng serve",

    "build": "ng build",

    "test": "ng test",

    "lint": "ng lint",

    "e2e": "ng e2e"

  },

Angular.json

"styles": [

              "./node\_modules/@angular/material/prebuilt-themes/deeppurple-amber.css",

              "src/styles.css"

            ],

"configurations": {

            "production": {

              "fileReplacements": [

                {

 "**replace": "src/environments/environment.ts",**

**"with": "src/environments/environment.prod.ts"**

                }

              ],

<https://angular.io/api/core/ChangeDetectorRef>

# ChangeDetectorRef[link](https://angular.io/api/core/ChangeDetectorRef#changedetectorref)

CLASS

Base class that provides change detection functionality. A change-detection tree collects all views that are to be checked for changes. Use the methods to add and remove views from the tree, initiate change-detection, and explicitly mark views as dirty, meaning that they have changed and need to be re-rendered.

abstract class [ChangeDetectorRef](https://angular.io/api/core/ChangeDetectorRef) {

[abstract **markForCheck**(): void](https://angular.io/api/core/ChangeDetectorRef#markForCheck)

[abstract **detach**(): void](https://angular.io/api/core/ChangeDetectorRef#detach)

[abstract **detectChanges**(): void](https://angular.io/api/core/ChangeDetectorRef#detectChanges)

[abstract **checkNoChanges**(): void](https://angular.io/api/core/ChangeDetectorRef#checkNoChanges)

[abstract **reattach**(): void](https://angular.io/api/core/ChangeDetectorRef#reattach)

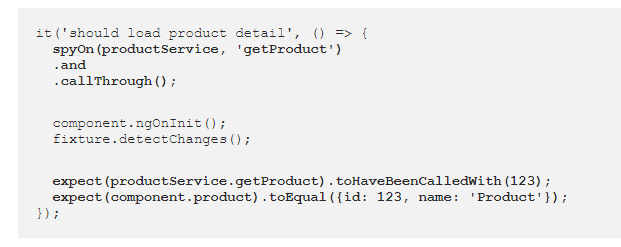
}

<https://hackr.io/blog/angular-interview-questions>

<https://dzone.com/articles/50-top-angular-interview-questions-amp-answers>

<https://www.interviewbit.com/angular-interview-questions/>

<https://medium.com/@ranostaj/unit-testing-angular-8-components-with-services-2863ed2b5fed>



he three status codes that felt the most appropriate are:

* **401** - Unauthorized
* **403** - Forbidden
* **404** - Not Found

The three status codes that felt the most appropriate are:

* **401** - Unauthorized
* **403** - Forbidden
* **404** - Not Found

In my mind, the use of each of these three HTTP status codes could be justified. Sarah is not authorized to view Tricia's profile (401); Sarah is forbidden from viewing someone else's profile (403); and, Sarah simply cannot see resources that she's not allowed to view (404).

The initial problem that I had with using either of the HTTP status codes, 401 or 403, was that I felt like it was exposing secure information. Both of those responses sort of say, "Yeah, that resource exists, but you can't see it." My problem with this is that it confirms that those resources exist.

When you ask a Doctor if he treats a particular patient (at least in Law & Order - wicked awesome show!), he will often say something to the effect of, "*Officer, you know I can neither confirm nor deny having a patient as it would be a breach of doctor-patient confidentiality*." This is how I feel about 401 and 403 in this particular type of resource request - I don't want to confirm or deny its existence.

# Difference Between setValue() And patchValue() In Angular 2

in this blog, I will explain the difference between setValue() and patchValue() which are frequently used in Angular2 reactive forms.

Updating Form controls from a model is very easy and flexible in Reactive Form API of Angular v2. So, setValue() and patchValue() are the methods by which we can update the Reactive forms controls from our model. Let us see the difference between these two in the following example.

**setValue()**

setValue() will set the value of all form controls of the form group.

1. **this**.profileForm = **new** FormGroup({
3. FirstName: **new** FormControl(''),
4. LastName: **new** FormControl(''),
5. UserName: **new** FormControl(''),
6. Email: **new** FormControl(''),
7. MobileNumber: **new** FormControl('')
9. })

We have the above FormGroup controls. Now, we will use setValue() methods to update all FormControl values.

1. profileForm.setValue({
2. "FirstName":"Ajay",
3. "LastName":"Panda",
4. "UserName":"ajay33",
5. "Email":"ajay.panda@gmail.com",
6. "MobileNumber":"8745212589"
7. });

As you can see from the above code, setValue() method will set all fromcontrol values from model. If you do not mention any of the formcontrol values in model, then it will throw an exception.

**patchValue()**patchValue() method will also set Formcontrol values from a model but only for those which we have mentioned in the model. So,  it is not necessary to pass all the model info in patchValue() method.

1. profileForm.patchValue({
2. "FirstName":"Ajay",
3. "LastName":"Panda"
4. });

As you can see from the above code, I am only passing two properties to my formgroup control, so patchValue() will set these two properties in form. Unlike to setValue(), it is not necessary to pass all models, we can only pass required models to the from group control.

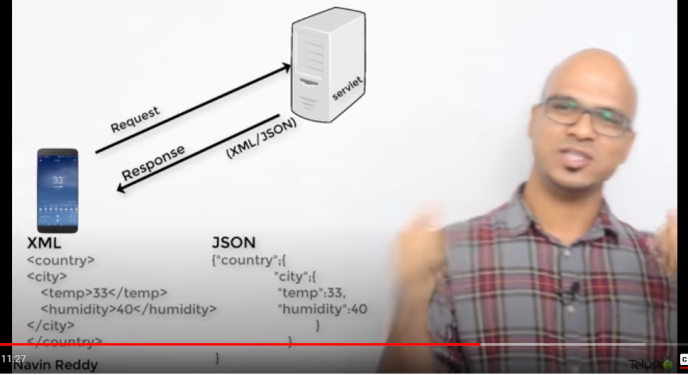
<https://dzone.com/articles/what-is-viewencapsulation-in-angular#:~:text=%7D-,In%20the%20browser%2C%20when%20you%20examine%20source%20code%2C%20you%20will,Next%2C%20let%20us%20explore%20ViewEncapsulation>.

**ViewEncapsulation**

**ViewEncapsulation.None,**

**ViewEncapsulation.Native,**

<https://stackblitz.com/edit/angular7-viewencapsulation?file=src%2Fapp%2Fcomponents%2Fshadowdom.component.ts>



New in angular 8