**Angular Docs:**

**Introduction:**

**What angular**

1.Angular is a framework which is use to build client side application

2.Great for single page application without reloading the entire page so it is a great choice to if my application is based on JavaScript

**Why angular:**

1.by design angular promotes modular approach and hence the application we build have clear structure.

2.by making use component so this component can be reusable.

3.Angular have inbuilt capability for validation and routing hence development quiker and easier.

4.unit testable

5.google team+micrsoft(Type script)

**Angular history:**

2010 angular js

2016 – angular version 2

2016 -4

2017 -5

2018 – april 6

2018 – oct 7

**Perquisites**

HTML,css,js

Basic TypeScript

**Environment Ready**

Tesxt editor -VS code

Install node js latest one

Node –v

Npm –v(it is making compatible to use Angular library)

Install Angulat CLI:

Visual studio Code as aEditor

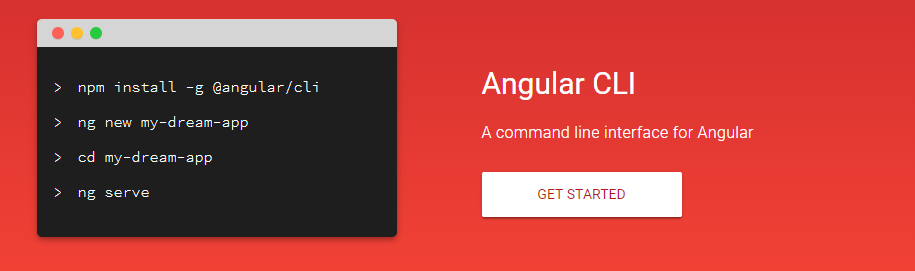
Angular CLI:

Command line interface :

The [Angular CLI](https://angular.io/cli) is a command-line tool for managing the Angular development cycle .it allows us to generate basic building block for our angular application.

Installation command is there

<https://cli.angular.io/>



**@component:**

A class with the @[Component](https://angular.io/api/core/Component)() [decorator](https://angular.io/guide/glossary#decorator) that associates it with a companion [template](https://angular.io/guide/glossary#template). Together, the component and template define a [view](https://angular.io/guide/glossary#view). A component is a special type of [directive](https://angular.io/guide/glossary#directive). The @[Component](https://angular.io/api/core/Component)() decorator extends the @[Directive](https://angular.io/api/core/Directive)() decorator with template-oriented features.

An Angular component class is responsible for exposing data and handling most of the view's display and user-interaction logic through [data binding](https://angular.io/guide/glossary#data-binding).

Read more about components, templates, and views.

1.first open the folder in VS code where we wanted to create angular project

In vs code terminal

If ng command is not recognized(

Solution:

1. Right Clicked on My Computer (windows)
2. Selected Advanced System Settings
3. Clicked "Environment Variables"
4. Under "Path" variable, made the FIRST value listed %AppData%\npm

Once I did that I was able to close powershell and reopen and all worked.

Or else use

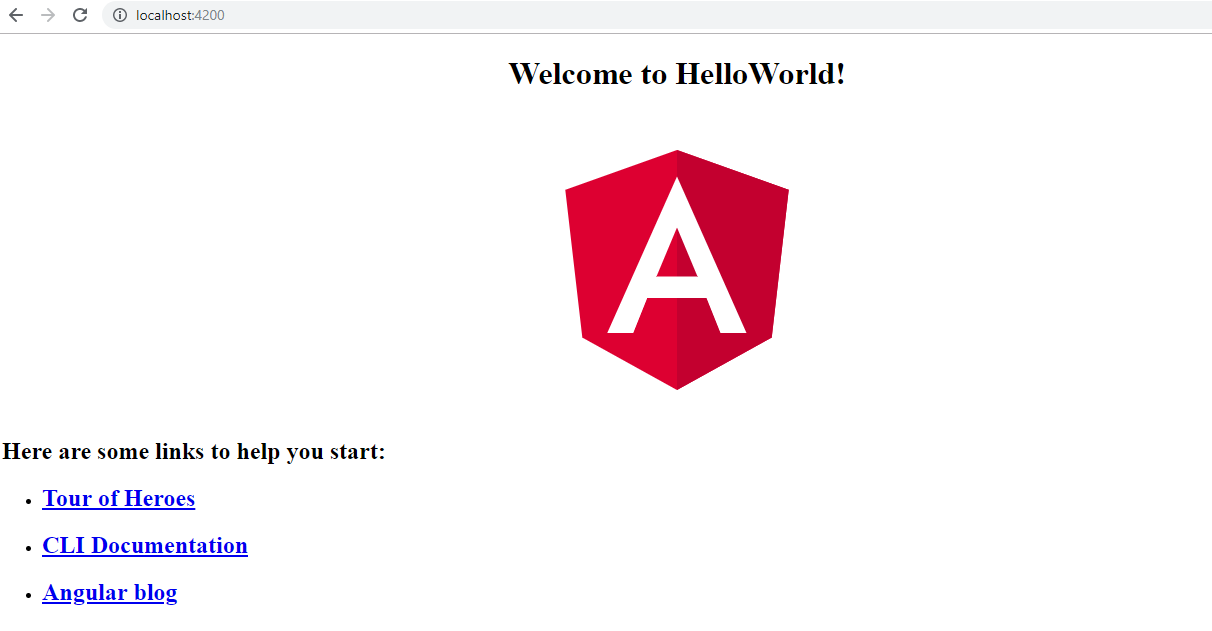
Use node command line and go specific folder

1.**ng new helloworld**

Go to project i.e Helloworld

**2.ng serve** (for running application) get succefully compiled

**Hit --🡪 localhost:4200**



**Architecture:**

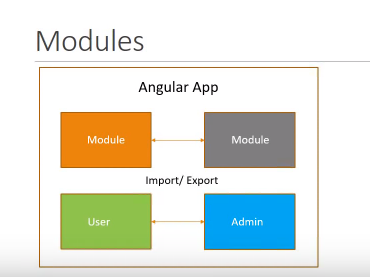
**Modules:**

Angular apps are modular and Angular has its own modularity system called *NgModules*. NgModules are containers for a cohesive block of code dedicated to an application domain, a workflow, or a closely related set of capabilities. They can contain components, service providers, and other code files whose scope is defined by the containing NgModule. They can import functionality that is exported from other NgModules, and export selected functionality for use by other NgModules.

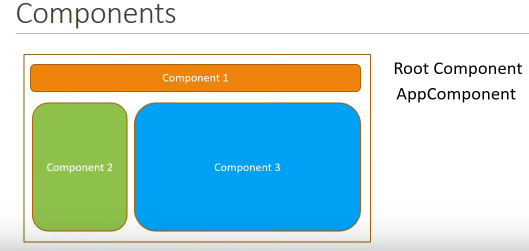
Every Angular app has at least one NgModule class, [the *root module*](https://angular.io/guide/bootstrapping), which is conventionally named AppModule and resides in a file named app.module.ts. You launch your app by *bootstrapping* the root NgModule.

While a small application might have only one NgModule, most apps have many more *feature modules*. The *root*NgModule for an app is so named because it can include child NgModules in a hierarchy of any depth.

An NgModule is defined by a class decorated with @[NgModule](https://angular.io/api/core/NgModule)(). The @[NgModule](https://angular.io/api/core/NgModule)() decorator is a function that takes a single metadata object, whose properties describe the module.



**Components:**

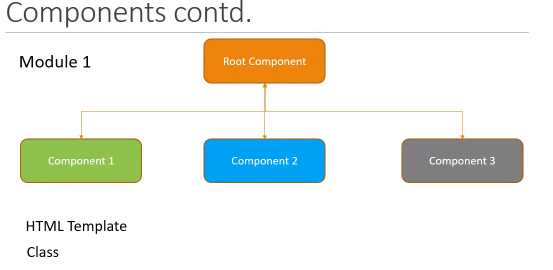


A component controls a patch of screen called a view. For example, individual components define and control each of the following views from the [Tutorial](https://angular.io/tutorial):

* The app root with the navigation links.
* The list of heroes.
* The hero editor.

Each module made up of component and services control a portion of view

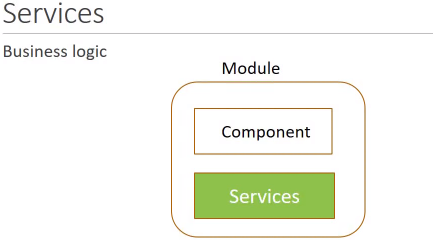
For ex:

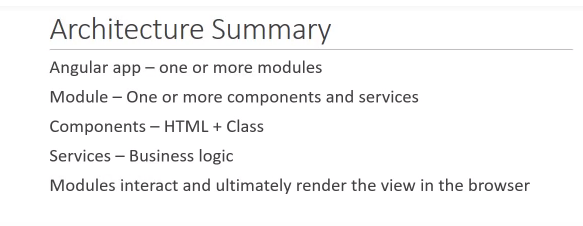


Component 1 :header, Component 2. side bar

**Services:**

**It is a class which contain business logic of our application**





**IN our project**

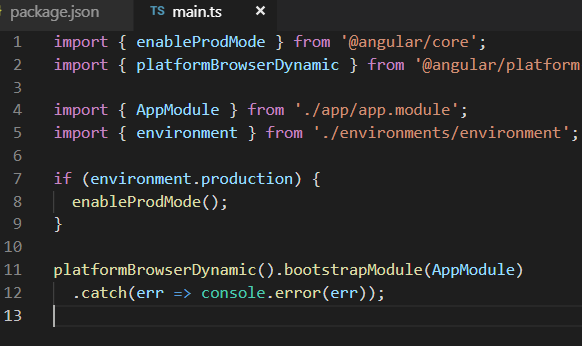
1.Package.json:

Contain all dependency information and script command like ng servewhich we have use or using

2.Src/app

Main.ts file : it is the entry point of our angular application.

When we run ng server main.ts file get execute and from this file it is bootstrapping the root module i.e AppModule



App.module.ts : it is the root module of our application from this file we are kickstart or bootstrap the AppComponent.

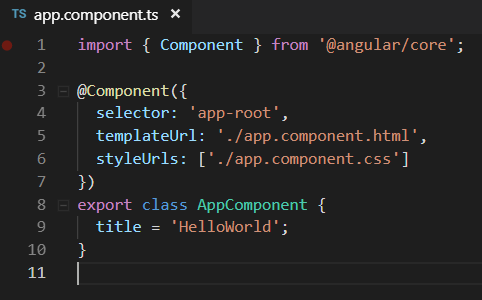


App.component.ts : it is the root component of our application

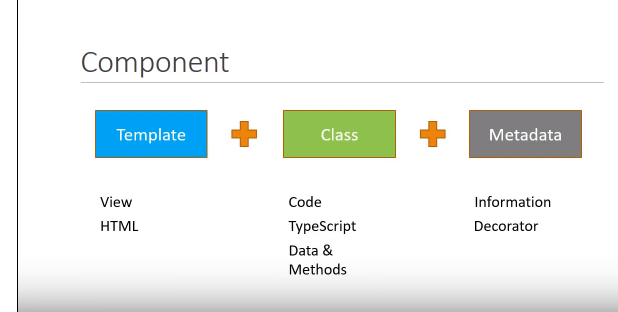
It contain two file

App.component.css,app.component.html (which control the view logic)

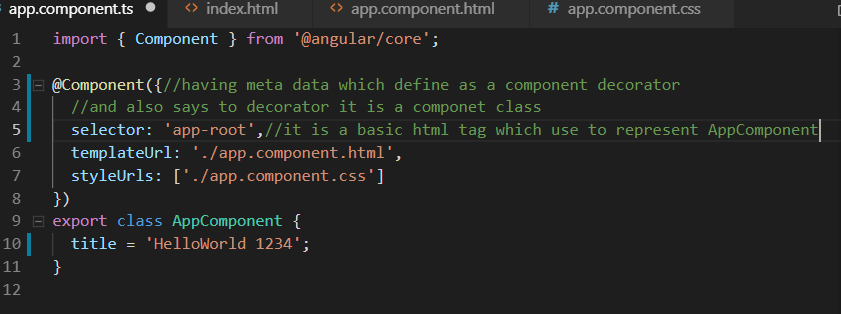
App.compontent.ts (type script contain class which contain logic of view data)



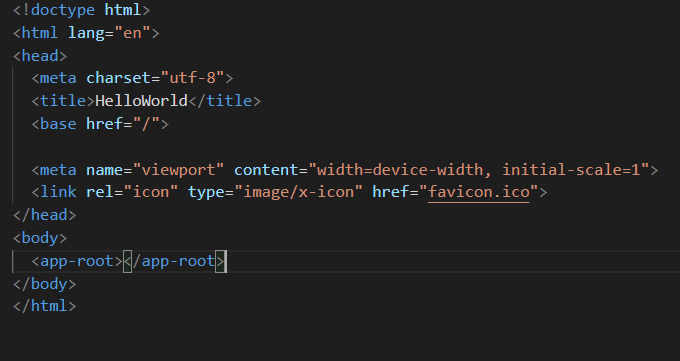
Lecture: 4



.



In Index.html we have to give <app-root></app-root> to render root html i.e. app.componenet.html.

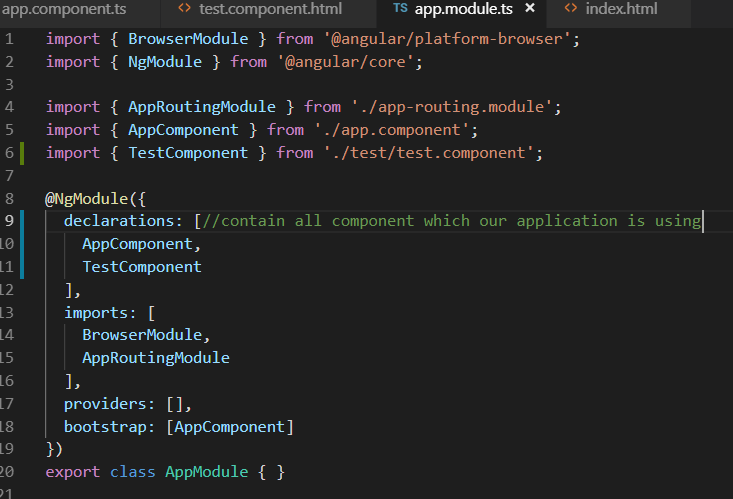


**Create a new component**

**ng g c test**



**Updated to app.module.ts**



App.componet.html- it is a root component html all html will render from here It self here we have to give that particular tag which is define in any component selector

<app-test></app-test>(inside App.componet.html file)

**Selector also we can define like**

**2.Selector : 🡪 .app-test**

**In html we have to include like that way:**

**<div class=”app-test”>**

**</div>**

**3.selector:[app-test]**

**<div app-test>**

**</div>**

**Template define in following way:**

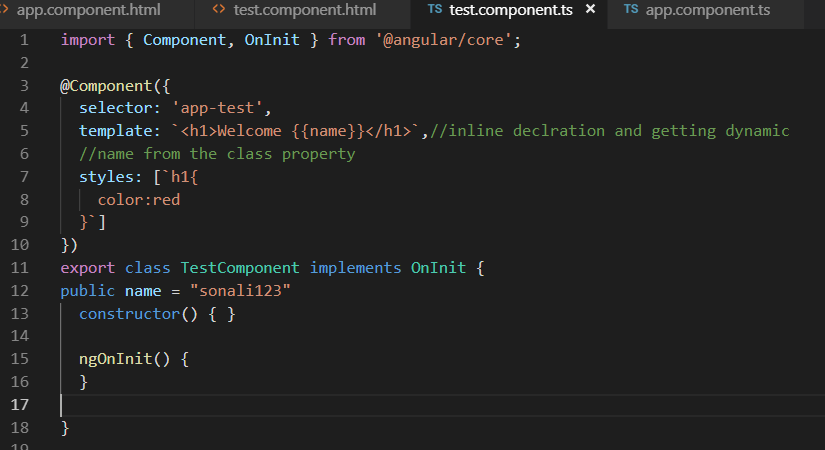
Template: `<div>whwtever we want show ehis is inline template</div>,<div></div>`

Styles:[`div{ color:red}

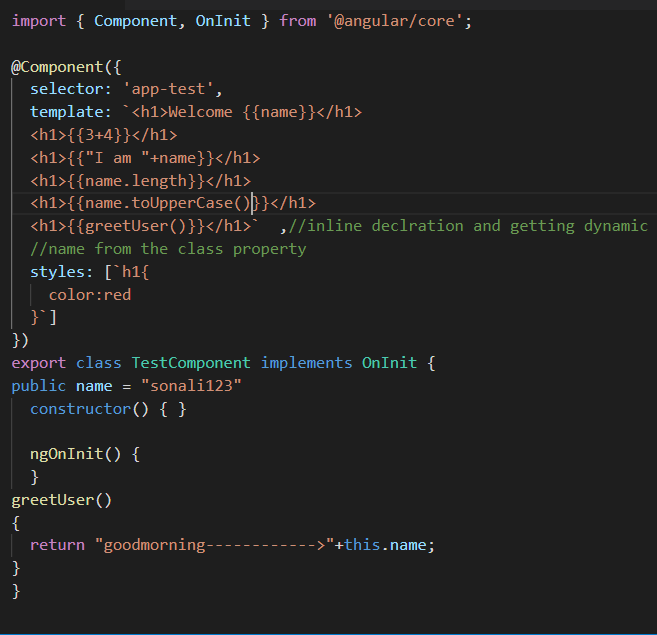


**Lecture -5 : Interpolation**

**Interpolation is binding the class variable to the template**



In interpolation in the {{}} rendering the class property so this is call interpolation so it is binding the data also bind JavaScript methods, and custom method as well which is define below.



Limitation of binding notation i.e. {{}}

1.direct assignment is not possible In {{}}

2.we can’t access global access variable i.e window,console,screen in {{}} beacause it is not aware about this variable

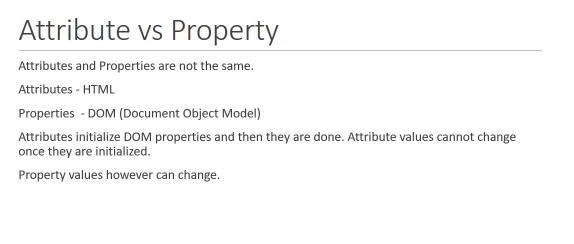
For that we have to perform in the class and bind them to that particular variable

**Lecture:6**

**Property Binding in angular:**

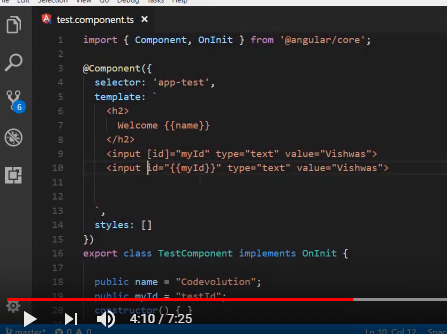
**Property binding Is required because interpolation is not support to bind Boolean value**

**Difference between Html attribute and DOM property:**

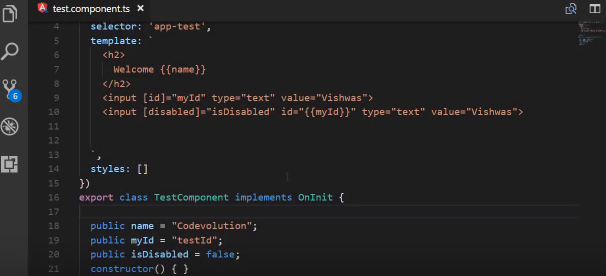


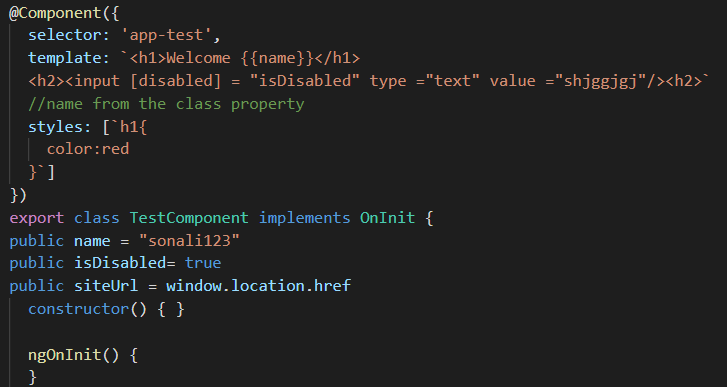
**$0.getAtrribute(“value”)**

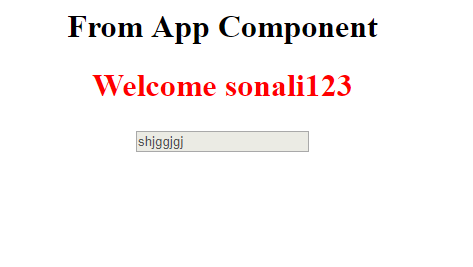
**$0.value = gghg**



**Here we are binding myId property to input html tag**

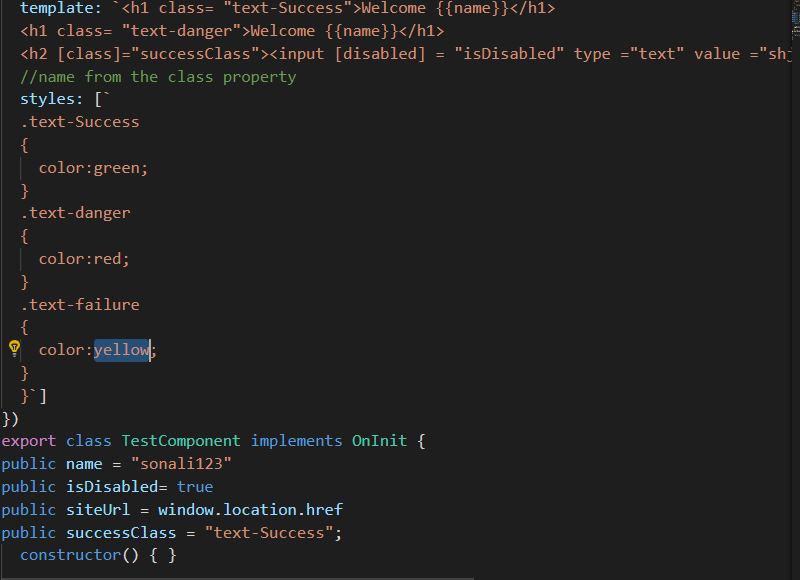






**Lecture 7: Class Binding**

**Apply css by class**



**If we wanted to provide multiple style on a single tag than we have to define attribute like abov is**

**i** public hasError = true;

public messageClass=

{

"text-Success":this.hasError,

"text-special" :this.hasError

}

**To use this In our html tag we have to use [ngClass]=”messageClass”**



**Lecture 8 : Style Binding**

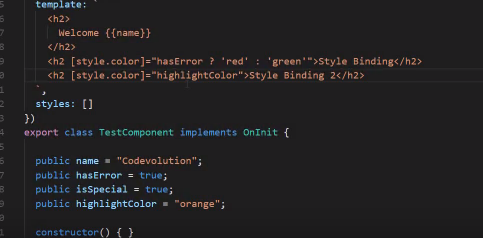
Inline style Binding:



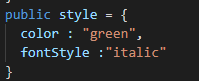
Conditionally apply style:



Binding property define style in html tag:

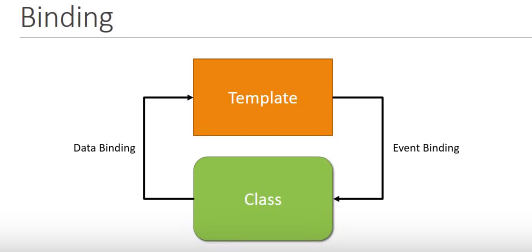


By using ngStyle tag:



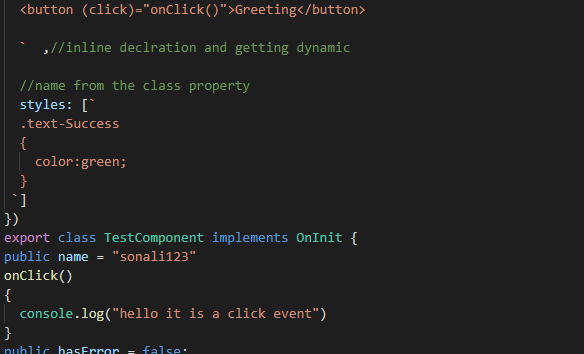


**Lecture 9: Event Binding**

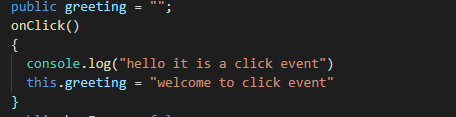


**1.On keyboard event Dom element view is changing**

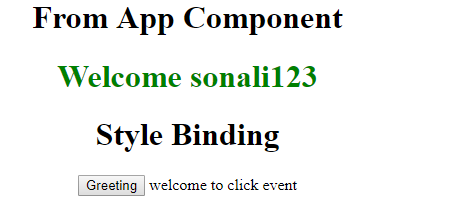
**Click is a event function**



**2.set property on mouse click event**

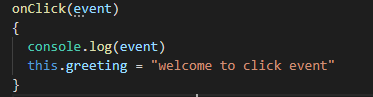


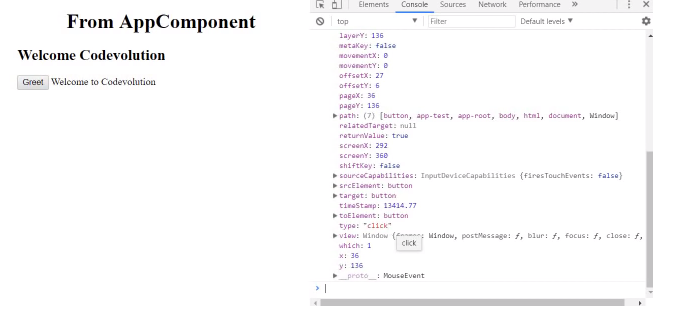




**Some times we required event information with event so have to pass like this way this $ event give us DOM information**





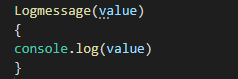


**Lecture - 10 - Template Reference Variables**

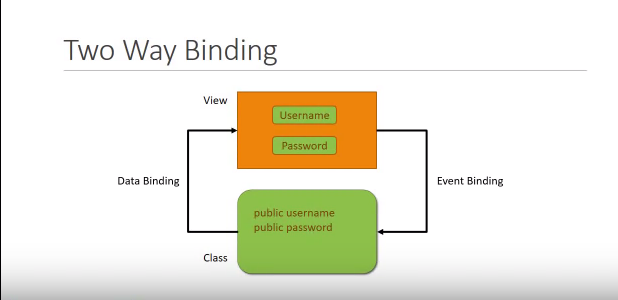
Some times we require to change in our data from view i.e from view to the class by user interaction

For example: some time we want input tag value to the console by clicking on any button here #myinput is define like an id and through myinput.value we are passing the reference variable value



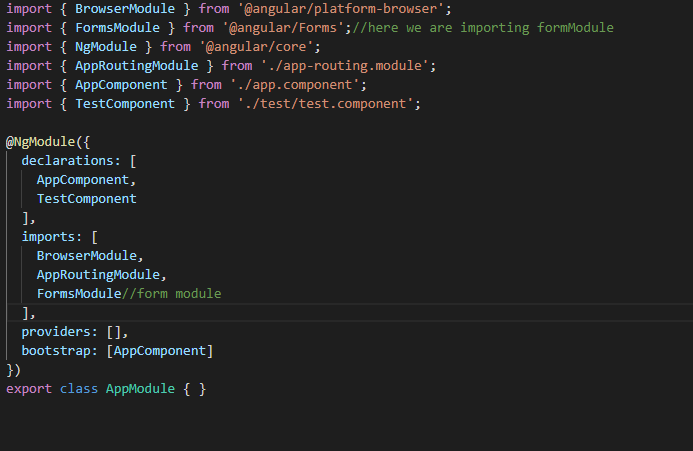


**Lecture - 11 – Two way binding**

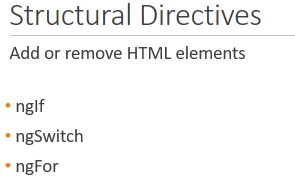


For performing two way binding angular had provide us **ngMode**l directive which is define like this [()] through this directive as we are inserting inside the input box property get update in the model class and through thr interpolation i.e {{}} it is update on view means in the template as well.

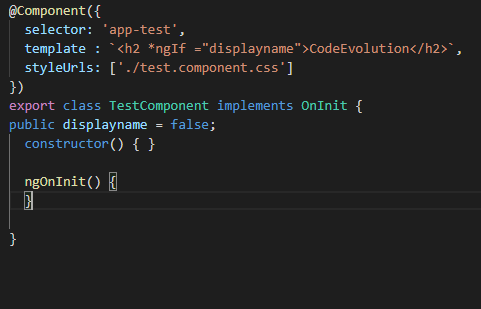
Ngmodel is not directly recognized to the class because it is belong from other model i.e. from FormModel have to import it in app.module.ts

.

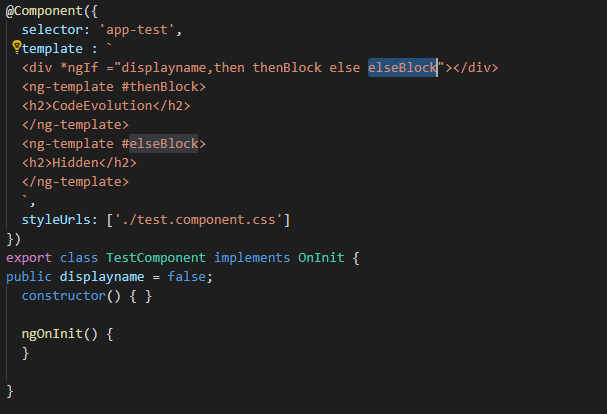
**Lecture - 12 – Ng if Directive**



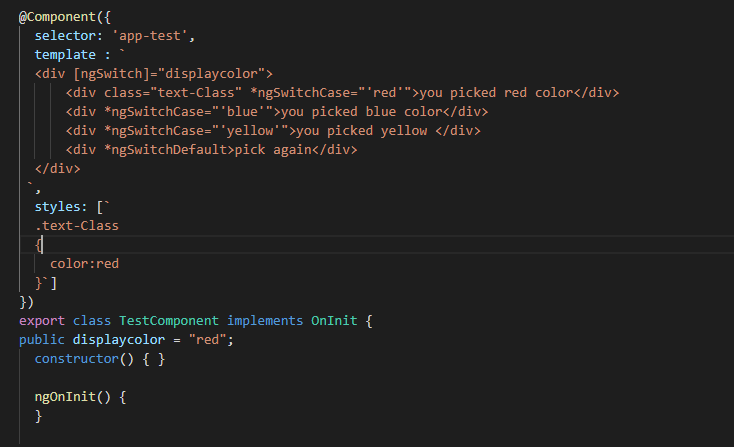
**ngIf : it is a conditional directive**



**ngIf directive for then and else block for performing conditional operation**

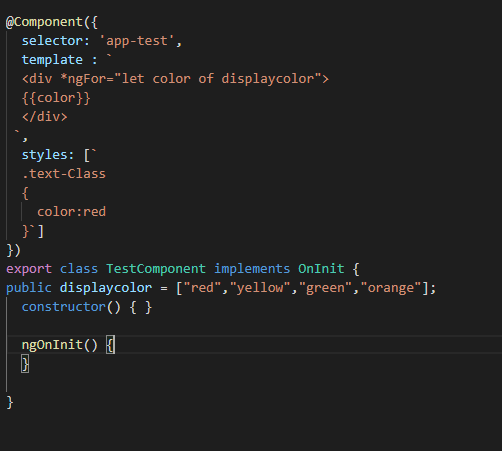


**Lecture - 13 – Ng Swich**



# Lecture - 14 - ngFor Directive

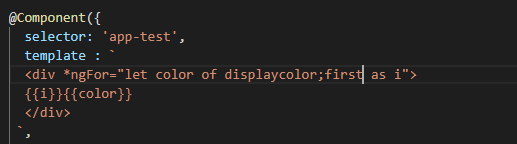
# ngFor Directive we are using for iterating the array elememt



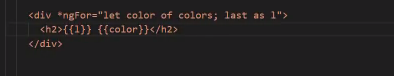
**To Provide indexing we can write like this way**



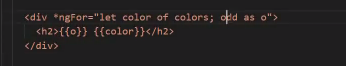
**For checking is this elements is first or not put the condition like this**



**Similarly we can check last**



**For odd or even order**



# Lecture - 15 Component Interaction

# How Component are interact with each other by using @output() and @input() decorator

# 

# How we can use parent component i.e app component property to child component i.e in test component

# 1.App.componenet.ts take one property

# 

# 2.in the test component selector i.e in app.template.html we will bind parent component property

# 

# 3.we have to receive this perent data property in test component by @input() to inform that it is not a normal property

# 

# For changing property name in child provide aliases as follows

# 

# For sending data from child to parent we have to use events

# Because child component don’t have parent component selector so for that we have to use eventEmitter and @output() as follows.

# Test.componenet.ts

# 

# App.template.html

# 

# Here $event gives us the access of child component event message

# App.component.ts

# Getting access and display in parent componenet

# 

# Lecture - 16 : Pipes in angular

# Pipes perform some inbulid functionality on html element for string property type

# Pipe transform the data only for the view it doesn’t change data for class property

# CODE:

# 

# OutPut:

# 

# Json Conversion:

# 

# Number Pipes

# 

# 

# Lecture - 17 : Services in angular

# 

# 

# 1.Shring data in multiple component class.

# 2.specific operation logic

# 3.database interaction logic

# Lecture - 18 : Dependency injection

# 

# 

# 

# 

# In above code with DI we are passing dependent class as a constructor

# So in case if any thing is changes in dependent class like passing parameter there is no need to change in parent class.

# 

# 

# 

# 

# 

# 1.Create Employee Service

# Ng g s employee

# 

# And have to two componenet

# Ng g c employee

# Ng g c employeeDetails

# 2.register the service in app module so scope will be over the application whatever componenet is register in appModule.ts

# 

# App.module.js

# 

# employeeDetailsComponent.js:

# 

# @injectable : when we require to inject one service to another service then we required @injectable and service class don’t have any decorator as component class have @component so that’s y we are keeping @injectable decorator with service class

# Lecture 20 : Http and Observable

# 

# Http : make a request to database or a web server and api to get or receive data .

# Observable: it is a response from the web server or an api.

# 

# 

# Lecture 21: Http and Observable (practical Implementation fetching data using Http)

# For make a http request first we have to introduce HttpClient in app.module.ts

# 

# 1.employee.service.ts

# 

# Make a assets/data folder in src folder and make a json file to get employee details.

# 

# Employee.json file

# 

# 2.Http.get method return observable response. So cast the response from observable to employee arraylist

# (a)create a interface

# employee.ts in app folder

# 

# after creating interface employee.service.ts

# 

# EmployeeDetail.componenet.ts

# 

this.\_empoyeeServeice.getEmployee().subscribe(data =>this.employees = data);

# in subscribe method the left hand side data is the argument of the function and right hand side data is the body of the function

# Lecture 22 : Http Error Handling

# For error handling first we have to install

# npm install rxjs-compat –save

# in the service class call catch method as follows:

# employeeService.ts

# 

# In employeeDetailComponent.ts

# 

# If I had given a wrong file name from where we are fetching the data then we find the error as follows.

# 

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Lecture 23 : Routing and navigation :

# 

# 

# 1.ng g routing-demo –routing

# (a).include base tag in index.html

# 

# (b).app.routing.module.ts

# Contain routing model for our application and configure routing modules

# (c).import app rounting module in app.module.ts

# 2.generate departmentList and employeeList componenet

# ng g c departmentList –it –is (for inline template and inline templates in component

# 3.Configure Routes

# We will do that in app.routingmodule.ts

# 

# Module.ts file

# 

# Now where this routing component is to display the ans is in router-Outlet directive

# App.componenet.html

# 

# 

# Lecture:24 - Wildcard Route and Redirecting Routes

# Wild card route is usinf if any thing we are passing which routing componenet is not existing then we use wildcard route

# (a)create page-not found component

# 

# Add this component to app-routingModule

# Note : put always at the last in const route :Routes[]

# 

# For empty path url we can make default redirect for any of the component

# As follows:

# But this is not correct way

# 

# This is always go to deparment on any of click

# 

# This will work perfectly:

# 

# Lecture 25 - Route Parameters

# 

# <https://github.com/gopinav/Angular-Tutorials/tree/master/routing-demo/src/app>

# we have added style to our existing project

# 1.here existing departmetList.ts

# 

# And view is like this

# 

# 3. how we will pass id parameter to department details for more information about department for that first create department Detail component

# ng g c department-details

# departmentlistcomponent.ts

# 

# Router is using for using route service and passing parameter

# departmentDetailscomponent.ts

# 

# Lecture 26 : ParamMap observable

# Note:

# ActivateRouter is using for reading parameter from url

# Through snapshot method routing is not happening with previous component call and because the ngOninit method Is not calling hence no component initialization is happening only it is trying to reuse the previous componenet

# below is the code for testing

# 

# 

# 

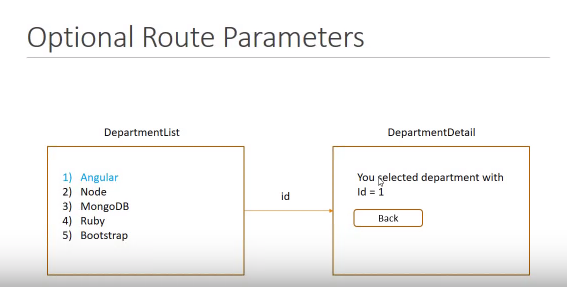
# Note: So with the param observable we can easily navigate previous and next page her we are using paramMap api to read the routing parameter

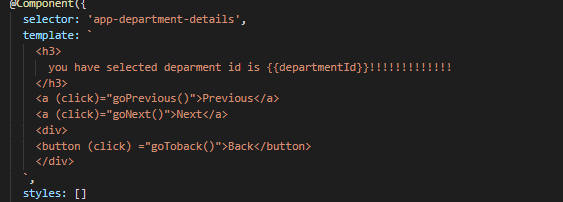
# Lecture 27 :Optional Route Parameter

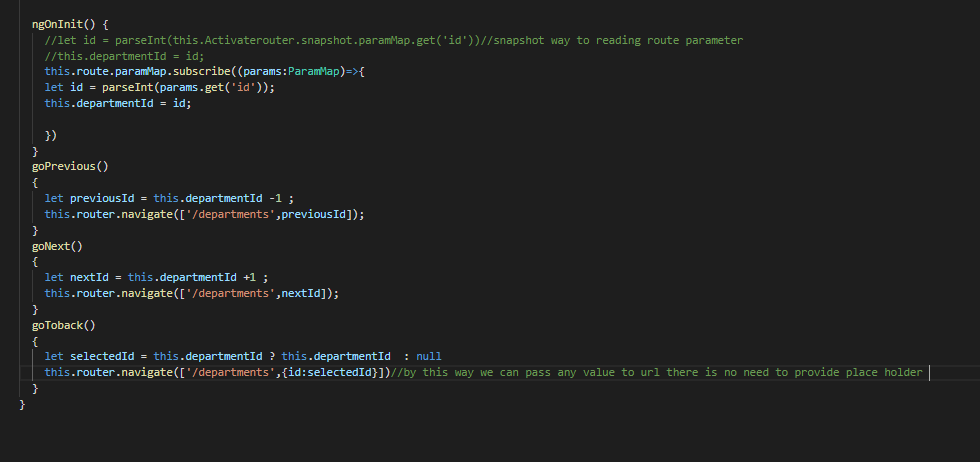
The Angular Router allows you to easily retrieve parameters from the URL which is an essential functionality that's required by most web applications. You can use both ways: the paramMap observable or the snapshot way but the latter requires you to be careful when re-using components.

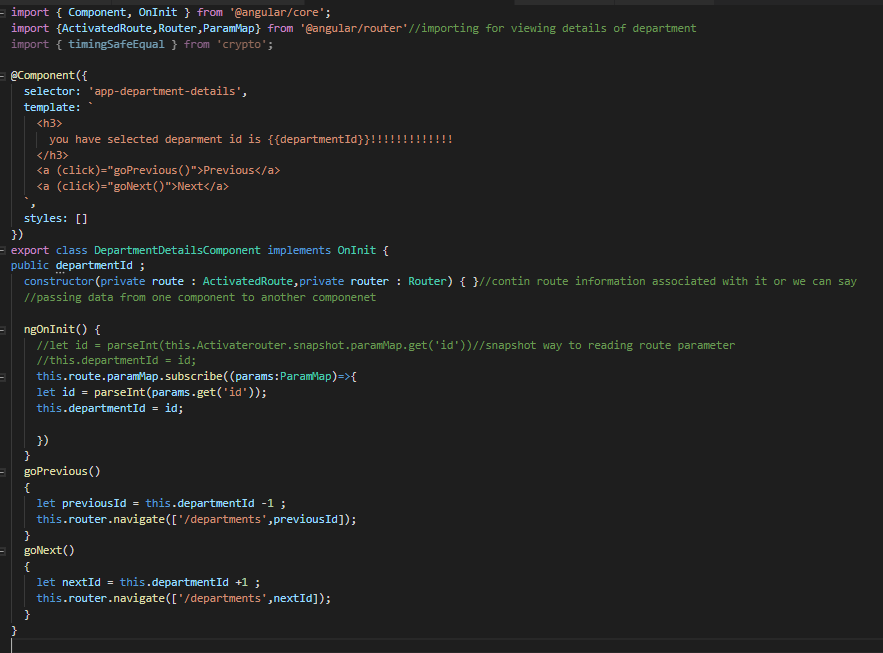
**Department-details-component.ts :**

Here we are implementing back button functionality for navigating list component and which department is already navigated highlate that which we perform by comparing id parameter which is presented in the url



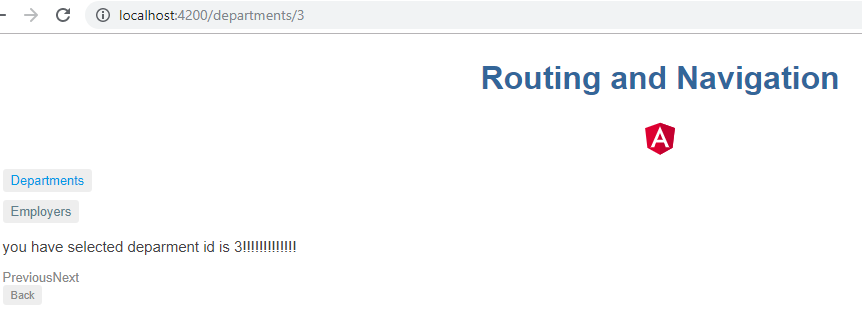








Out put: here if previous tag have selected 3 id and click on back to navigating list of department so we are getting as follows



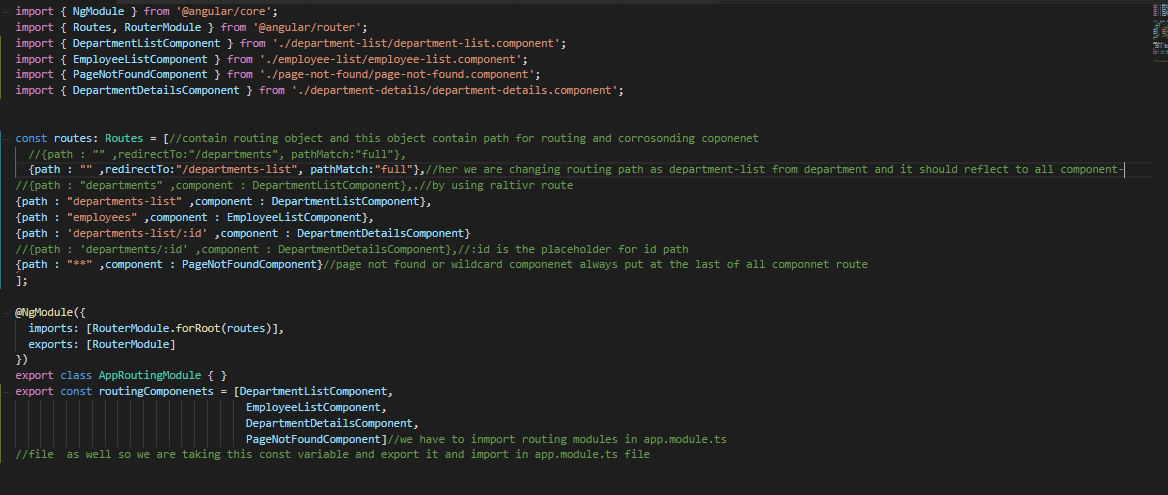


**Lecture 28: Relative Navigation**

If we wanted to change in routing url so we have to change in every component where we are using that routing path and that component so this way of routing is not flexible so for that we are using relative Navigation

1.departmrnt.routing.ts class

2.app.component.html class change path according to requirement



# At line no 34 we added relatviePath for getting current routing path and added id parameter in

# department-listcomponent.ts

# 

# Departmrnt-detailcomponrnt.ts

# 

# Lecture 29: Child Routes

# Sometimes some routing is calling inside the route or we can say some route is view inside the other route that route is calling child route

# 1.ng g c department-overview –it –is

# 2.ng g c department-contactinfo –it –is

# Above both component will call inside the department-detail componenet hence we have to reformate the department-detail component routing path in app.routing.ts for that We have to take children property in department-detail object all three images for department-setail-componenet

# As follows

# 

# This two router only view inside the department-detail component so we have to give router directive inside this component

# 

# 

# Lecture 30: Angular 6 new feature and changes:

# 

# 

# 

# 

# Previously we have to provide in root model as follow nad now as follows

# 

# 

# This above all changes in angular framework

# 

# 

# 

# 

# 

# 

# 

# Lecture 31: Angular 6 new feature and changes:

# For updating from 5 to 6 follow following link:

# <https://update.angular.io/>

# 

# 

# For uninstalling current cli

# npm uninstall -g @angular/cli

# npm cache verify

# npm install –g @angular/cli@latest

# if we are getting any problem for installing then first enter

# ng cache clean –force

# locally as mentioned in above screen shot this command update the angulal version check in

# package.json

# npm install --save-dev @angular/cli@latest

# 

# 

# 

# 

# 

# Throw this command we are updated rxjs version and use latest rxjs import packages

# mpm uninstall rxjs compat

# Lecture 32 : Angular 8 Features and changes

# 

# 

# 

# npm update lazy route

# It will update automatically

# 

# 

# 

# Lecture 33 :Angular Forms

# 

# 

# 

# 

# Lecture 34 :Template Driven Angular Forms

# 

# 

# 1.genrate the new project with latest cli so before creating make sure u have fired following commands

# 

# 

# Create a folder

# Angualr-form and go till there in vsula studio code

# ng new tdf,

# ng serve

# 2.add bootstrap to our application(it is css frame work which make better style for our html)

# <https://getbootstrap.com/docs/4.3/getting-started/introduction/>

# 

# Copy this css link in index.html

# 

# Bootstrap added tag

# 

# Lecture 36 :Adding form html

# Creating form html

# app.componenet.html

# 

# App.componenet.ts

# 

# Lecture 37 :Binding data with ngForm(i.e from user to model class)

# To use angular form we have too import formsmodule in app.module.ts

# Now we have created a basic html but angular form tag bind its form control value by template reference variable and

# Angular attaches its form value by ngForm directive with form tag which give valuable information about that form

# 

# 

# ngGroup directive we are using filds inside the fild and create object inside object

# 

# 

# Note: for data binding we are having three directive for template driven and form value binding

# ngForm

# ngModel

# ngGroup

# Lecture 38 :Tracking state and validity

# Here we are getting data from the user and send to the model class and later we will send it to the server for that create class

# ng generate class User

# 

# 

# Now we have to bind userModel data to our enrolment form by using this user model class as follows

# 

# For binding model class data to html template we have to use [ngModel] for one way property binding i.e from model to view and use [(ngMode)] two way property binding i.e from view to model and from model to view as follows

# And this concept is also use for updating any existing data

# ngMode is for html attribute binding only which have performed previously

# 

# Lecture 39 :Validation on tdf with visual feedback

# 

# 

# #nam22 is the template reference variable and we can check applied class on particular filed by {{nam22.className}}

# 

# Ng-touched : value Is visited or not

# Ng-pristin :value is not change and if change ng-dirty is applied

# Ng-vailid : if field contain correct value if we applied required attribute in the particular file and keep as blank ng-invailid class will apply

# Alternatively we can use for each class it provide associated property as follows

# 

# For getting access of model attribute we can get as follows:

# 

# Hence this will give us Boolean value then a class value which is bettet approach hence

# 

# Apply conditional validation on each field by duing bootstrap class-is-valid class and apply on the field

# 

# This class will provide style on field is condition is not satisfied

# For phone with pattern

# 

# Hence we can see visually something is wrong

# 

# Lecture 40 :provide visual feedback that is error message to User

# 

# 

# We can provide conditionally and specific error message as follows

# 

# Validation for Select tag :

# For select tag we have to use like this way:

# 

# 

# Lecture 41 :Form Validation

# For checking form is valid or not

# 

# 

# For custom form tag validation:

# 

# Lecture 42 :Submit Form data

# How to post the data to the server

# Bind ngSubmit directive which emitted on submit event to the form and novalidate attribute is using for avoiding browser validation.

# 

# 

# For sending data to the server we have to use service

# ng g s enrolment

# 

# Import httpClientModule in module.ts

# So now we are in a position to make http request

# 

# 

# Lecture 42 : Expresss server to receive form data

# 1.create a server folder in Angularform

# 2.crete a new package.json open server folder in terminal

# npm init –yes

# Package.json file will create in server folder

# 

# 3.now add dependency

# npm install –save express body-parser cors

# express : it is a web server

# body-parser : middle ware to handle form data

# cors : it is a package to make request on different server port

# crete a file i.e

# server.js

# 

# Now we are posting data from the form by service hence

# 

# And to receive post request on express server

# 

# After once submition form disappear

# 

# 

# Lecture 43 : Error Handling from the server if something is going wrong

# 

# 

# 

# 

# Lecture 44 : explaine d how TDF is use formGroup and form control

# Lecture 45 : reactive Form:

# 

# 

## High-level Differences between Template-driven and Reactive Forms

Below are some of the high-level differences between the two types:

1. Template-driven forms make use of the "FormsModule", while reactive forms are based on "ReactiveFormsModule".
2. Template-driven forms are asynchronous in nature, whereas Reactive forms are mostly synchronous.
3. In a template-driven approach, most of the logic is driven from the template, whereas in reactive-driven approach, the logic resides mainly in the component or typescript code. Let us get started by generating a component and then we'll update our form code.

# <https://www.pluralsight.com/guides/difference-between-template-driven-and-reactive-forms-angular>

# Lecture 46-17 Adding Html form in reactive form group

# Crate a new project

# ng new reactive-form1

# app.component.html

# 

# Lecture 47 -18 Angular Forms tutorial crating the reactive FormModule:

# App.module.ts

# Import reactiveFormModule

# 

# In app.component.ts

# In reactive form model is using following two class class i.e

# 

# app.component.ts

# 

# Binding formGrop module with Html

# 

# Lecture 51-19 Nesting Form group

# FormGroup clss is also be use to group togethere different form control

# Hence we have create a address formGrop class which contain other formControl file as follow and make communicate in html and formmodel class i.e component class

# 

# 

# Lecture 52-20 Managing Control Values

# How control values set without user interation i.e programitaically

# Example: if we are retriving formControl values from a backend api and update formContol values with in values

# We can perform that by using setValue method for populating all form value and use patchValue method for populating only required value both this methodr provided by ReactiveForm

# 

# 

# Lecture 53-21 Form BuilderService

# Form builder is altetnative way of crating formGroup which give better way for crating multiple formGroup or manully rpeatative formCreation can be void by this approch

# 

# Lecture 54-22 Simple Vailidation

# For red box for invalid field

# 

# 

# For showing single message

# 

# For multiple vaildation rule on one field make validator attribute as an arry and mention more vailidation rules

# 

# Conditional message displaying :

# 

# For more cleaner code in html we can show define like this

# 

# 

# Lecture 55-23 Custom Validation

# 

# 

# 

# 

# Lecture 56-24 Cross Field Validation

# We are vaildtaing a field to match with anothor field ex.confirmPassword is matching with passwod

# Create a shared folder in app folder and crete a vailidator class

# 

# Passwors-validator.ts

# 

# App.component.ts

# 

# App.component.html :

# 

# Out Put:

# 

# Lecture 57-25 Conditional Validation

# means a particular validation is applied for a certain condition

# here we will use ngOnit life cycle method which is calling duration on compaonent get created for more details go throgh on following link:

# <https://angular.io/guide/lifecycle-hooks>

|  |  |
| --- | --- |
| ngOnInit() | Initialize the directive/component after Angular first displays the data-bound properties and sets the directive/component's input properties.  Called *once*, after the *first* ngOnChanges(). |

# We want to subscribe on ngOninit lifecycle hook

# 

# 

get email()//to use this refernce we can make html code cleaner by direct calling this method

{

return this.registrationForms.get('email');

}

# 

# Lecture 58-25 Dynamtic form controls

# On run time if we wanted to add a field and at that time form control will be generate for that we are using formArray Directive

# 1.add formArray to makes us possible to maintain dynamic control

# 

# 2.add alternateEmail field with creation of formBuilder form array

# 

# 3.create getter to call or get aceess of this formArray field on html as well on component as well

# 

# 4.to create a method which call dynamically call form controls into the formcontrol array

# 

# 5.add button to html for trigger this function and on that button we awill call athat particualt method

# 

# 6 iterate over the form array and display on the form fields on html

# 

# Complete form:

# App.component.ts

# 

# App.componet.html

# 

# 

# Lecture 59-26 Submitting Or Posting Reactive Form On The server

# 1.First bind (ngSubmit) event which is trigger on on submit event and call onSubmit() handalor in componenet class method.

# 

# 

# 2.to make this data availble on the server crete a service class

# Ng g s registration

# Import httpClient and pass into the construtor and also add in app.module.ts

# 

# 

# Make register method for make post request

# 

# Subcribe to the observable to get data from the server api on app.component.ts

# 

# Disable register button if any form value is invalid

# 