Stand alone app  
Web based app  
-- Client Server architecture (3 layers)  
1. Presentation layer - client side - to present the info to the client  
1. HTML - to design static web pages  
2. CSS - to apply styles  
3. Javascript - to develop dynamic web pages  
4. AJAX - Asynchorous Javascript and XML - not a prg lang it is just a technique to create faster and better performance web pages - made of JS - indicates client interaction must be continuos/fluid - always response is XML  
5. jquery - contains javascript library function - show()  
6. Microsoft silverlight - UI in .net platform  
7. Adobe Flex - UI in Java  
8. Angular JS (1.0)- used to develop client part based on MVC architecture made up of Javascript  
9. Angular 2/Angular 4/Angular 5/Angular 6/Angular 7/Angular 8 - used to develop client part made up of typescript  
10. Node JS -used to develop server part using Javascript  
11. React JS - used to develop client part using typescript  
12. Knockout JS  
2. Business logic layer - server side - to write logic  
3. Data access layer - db side - to store any info in db  
  
Typescript 3.7 (latest version)  
1. open source, object oriented programming maintained by Microsoft  
2. strongly typed (contains data type) - superset of Javascript - internally all typescript compiles as Javascript  
3. Typescript does not directly on browser, it needs some compiler to convert typescript to javascript - Babel transpiler  
4. Typescript is ES6 version of JS  
5. Typescript program is saved as .ts extension  
6. Typescript contains class, interface,inheritance

Software requirements  
1. Nodejs - contains npm(node package manager) - all packages to develop angular app  
2. Visual studio code editor

> node --version

>npm --version

1. we need a package called typescript  
>npm install -g typescript

2. Compile typescript programs  
>tsc first.ts

3. Execute javascript  
>node first.js

npm config rm proxy  
npm config rm https-proxy

npm config set registry https://registry.npmjs.org/

Typescript data types

1. Build in/primitive data type  
Number,string,boolean,void,null,undefined,any

2. User defined datatype  
Array, tuple, interface,class,enums,functions

Array - collection of similar data elts

Tuple - includes set of values of different data type

Interface  
1. contains properties and method declarations  
2. interface does not convert into javascript, it is used only for type checking  
3. Optional property - ?  
4. readonly property  
5. Interface can also be extended  
interface extends interface  
class implements interface  
interface extends class

Class  
- contains properties, methods, constructor  
- constructor can be created using "constructor"  
- no constructor overloading  
- An object for the class created using "new" operator where it invokes constructor  
- In the constructor the members of the class can be accessed only using "this" keyword

>npm install -g @angular/cli

Angular 7

1.Angular is a framework used to build client side app  
2. great to develop SPA (Single Page application)

Why?  
1. Angular by default promotes modular approach, hence appl we build will have a clear structure  
2. use of Components- we can have lot of reuseable code  
3. provides in built features like validation, routing,forms which makes development  
easier and quick  
4. Angular codes can be unit testable  
Jasmine - unit testing framework  
karma - tool to execute unit testing  
Protractor - end to end testing  
5. Angular is product of Google team and make use of typescript from microsoft

Angular CLI (Command Line Interface)  
-- created a base project which is required for developing angular appl

>npm install -g @angular/cli - to install angular cli

>ng new project -- creates new angular project

e2e - used to do protractor related testing  
node\_modules - contains all packages related to angular appl  
src - contains the angular project source code like components,services,pipes,images,templates,styles

package.json - contains the packages to build and run angular appl  
1. dependencies - contains packages essential for running angular app  
2. devdependencies - packages only required to develop the angular app

to start angular project in devepolemnt -- > ng serve or npm start  
now angular appl by default start in port no 2000 -- http://localhost:2000/  
> ng serve --open  
--- automatically open the browser and run in 4200  
>ng serve --open --port 3000

To start angular in prod env  
> ng build /npm build

To run unit testing  
> ng test / npm test --------- will start karma tool to run unit test written by Jasmine

To run linting tools (verify code quality)  
>ng lint / npm lint

To run end to end testing  
>ng e2e / npm e2e

In angular, always first file excueted is index.html

main.ts - specify the name of entry point file which is main.ts contains the code to bootstrap our root appl module is appModule

polyfills.ts -- specify the angular appl to exxecute on wide range of latest browsers

app.module.ts - contains all the components created for angular appl

main.ts - AppModule (app.module.ts) - AppComponent (app.component.ts) - app.component.html

Component - 4 files  
app.component.ts -- we have write all logic inside component  
app.component.html -- for the related component we execute html file  
app.component.css - styles for HTML page  
app.component.spec.ts -- write unit test case using Jasmine

Component - it is a class with template and decorator  
-- @Component - write business logic

selector - contains custom tag to run component

templateUrl - if html file contains many code in that case we create separate html file   
template - if html code is very less, we write the code directly

stylesUrl/styles

Interpolation/data binding  
-- used to display the data from component to browser  
-- using {{ expr }}

2 types  
1.one way binding - from component to view or from view to component  
2. two way binding - from component to view and from view to component

Property binding  
-- moves component class property to html elt   
-- using []

Event binding  
-- flow from HTML page to component class  
-- using ( )

create component inside src/app  
> ng g component employee

>ng g component sample --prefix tech

Structural directive  
- changes the structure of DOM and display  
3 types  
1. \*ngIf - used to check single condition  
2. \*ngFor - used for iteration  
3. \*ngSwitch,\*ngSwitchCase,\*ngSwitchDefault - check multiple condition

Pipes  
-- at the time of displaying, if we want to format in different manner in that case we use pipes -- using | symbol  
-- {{ expr | pipes }}

Types of pipes  
1. currency pipe  
{{ 1000 | currency : 'USD' }} - $1000  
{{ 1000 | currency : 'INR' }} - Rs1000  
{{ 1000 | currency : 'INR':'code' }} - INR 1000  
{{ 1000 | currency : 'USD':'symbol' }} - $ 1000

2. uppercase  
3. lowercase  
{{ name | uppercase }}  
4. date - short,long,medium,full - both date and time  
shortDate,longDate,mediumDate,fullDate - only date  
shortTime,longTime,mediumTime,fullTime - only time

{{dob | date:'dd-MM-yyyy'}}  
{{dob | date:'fullDate'}}

5. percent  
{{0.5757 | percent }} - 57%

6. json - if we want to display in json format  
7. number  
{{567.78967 | number:'3.3-3'}}  
first 3- min integer digit  
second 3 - min fraction digit  
third 3 - max fraction digit

8. slice - delete elements from array

Custom pipe/user defined pipes

>ng g pipe "pipename"

Two types of pipe  
1. Pure pipe -- is only called when detects a change in the value or parameter passed to the pipe  
-- by default all pipes are pure pipe  
-- pure:true  
2. Impure pipe -- is called for every change detection cycle no matter whether the value or parameter changes  
-- pure:false

Nested components   
- one compoennt into another component

Nested Components

    -- one component inside another component

ng serve --open --port 2000

Parent component - Employeelist component

child component - EmployeeCount component

   based on the values of employees in employeelist component, the radio button value should be automatically changed

how to pass data from parent component to child component - @Input with property binding

how to pass data from child component to parent component - @Output

let a: number=25;

let b: string='25';

console.log(a==b); //true- check only value

console.log(a===b); //false- check both value and datatype

@Output

1. Create a property called selectedRadioButtonValue to keep track of the value of radio button

2. create a custom event with @Output which use EventEmitter class, so whenever radio button value changes it invokes the event handler method

3. create a function which raise our custom event which uses emit() to raise the event which keep track of the value selected

4. now we use two way binding and binds the selectedRadioButtonValue and pass the value to the custom event

to do two way binding we use [(ngModel)] present inside FormsModule and this FormsModules is imported in app.module.ts

5. Now we pass the selected radio button value from child component(employeecount) to parent component(employeelist), for that we create a property which keeps track of selected radio button value

6. create a function that will be called when custom event within child component is raised

7. we are going bind the custom event to the <app-employeecount> selector

$event is used to receive selected radio button value

Life cycle hooks

1. ngOnChanges() -- executes everytime when a value of input control within the component has been changed

2. ngOnInit() - called after ngOnChanges(), used to initialize the data in the component

3. ngDoCheck() - This event is triggered every time for input properties of the component is checked

4. ngAfterContentInit() - executes just after ngDoCheck() to link with chcild component initialization

5. ngAfterContentChecked() - called after ngAfterContentInit()and on every subsequent execution of ngDoCheck()

6. ngAfterViewInit() - called after ngAfterContentChecked()

7. ngAfterViewChecked() - called after ngAfterViewInit()

8. ngOnDestroy() - just before the anguar destroys the component

To install bootstrap

>npm install bootstrap@4 --save

--save - install in dependencies

--save --dev - install in dev dependencies

2. configure in angular.json

"scripts": [

              "./node\_modules/bootstrap/dist/js/bootstrap.min.js"

            ]

3. configure in src/styles.css

@import "~bootstrap/dist/css/bootstrap.css";

-----------------------------------------------------------------=----------------------------------------------

Routing

1. used to develop Single page application

Steps:

1. app-routing.module.ts -- we are configure all our routing

We configure the routes in a constant called appRoutes imported from @angular/Router. This constant hold array or JS object which contains path and component which should be loaded

Normally if we use href , it will redirect to each link, but it will be refreshed each time (ie) each time it contacts the server to do processing which is not good way so angular provides a directive called

routerLink

<router-outlet> where we want to load our component

>npm install jquery --save

>npm install bootstrap@3 --save

in angular.json

"scripts": [

              "node\_modules/jquery/dist/jquery.min.js",

              "node\_modules/bootstrap/dist/js/bootstrap.min.js"

            ]

Styles.css

@import"~bootstrap/dist/css/bootstrap.css";

routerLinkActive directive which has bootstrap class called active, but whenever we use routerLinkActive  only home and other component also active

  For '/' case, it always presents between /users as well as /servers, so empty '/' is part of all path, so we dont want home to marked active all time.

   So we use "routerLinkActiveOptions" which is property binding contains JS objects and one configuration called "exact:true", which tells angular if exact full path is matching it makes that component to active

1. Routing Programmatically (inside the component)

     -- Normally all routing is configured in app-routing.module.ts

     -- But now we are going to do routing from the component

     -- navigate() inside Router class

2. Passing params with routes

        -- with :parameter

3. Fetching Route params

      -- We need to inject ActivatedRoute to access the route param inside ngOnInit()

4. Fetching route param reactively

/users/10/Anna

whenever we executed it updates the url, but it is not reflected in ur web page, for that purpose we have to go for 3rd party feature called Observable

  Observable use subscribe(), will update the user object only if the parameter changes. This code will not be executed when ngOnInit() runs, if the parameter changes and Observable will fire and then we receive the updated parameter assigns to the user object

5.Whenever we enter some wrong url, we get error

   We want to redirect to some specific page when there are no routes are available, for that we use a property called "redirectTo" and path="\*\*" (wild card routes) and make sure this path should be in last one in the array

6. Query param - passed between ? and &

    fragment - #

7. child routes

     -- with help of "children" property

8. Location strategy Hash mode

      Here we have routes like /servers,/users etc, it works fine in local setup but we host the appl in web server, in that case it will be a problem becuase url is always parsed and handled by server

     So whenever we run index.html in our server it leads to 404 error

          Because all url are parsed by server not by angular

For eg: we have url called "/servers", it will look for /servers route on real time server , but we dont have that route because we have only one file called index.html, it leads to 404 error

we want to make webserver to handle the request for that we use useHash:true in app-routing.module.ts

This will create "#" which called hash mode routing, which informs the web server only care about the path in url before # and after # whatever is there that will be taken care by the server

<http://localhost:4200/servers>

Guards

  --  in order to protect the component

1. canActivate

      -- we want to run any code before the component is loaded

2. canActivateChild - if we want to guard the child component

ng g component items

ng g component login

ng g component admin

ng serve

Service - actually component is used to get the data from html pages and it send to service (where we write all logic inside service )

>ng g service authservice

Angular Forms

   Html template is used to collect data and component class handles that data and sent the data to the server using service

2 types forms

1. Template driven forms  - form code are written in HTML

2. Reactive/ model driven forms - form codes are written in component class

Template driven forms

1. import a module FormsModule

2. Much of code and logic are reside in the HTML template, we have bulky html code and minimal component code

3. we heavily relay on two way binding using [(ngModel)], we dont have to keep track of input field values and react to change to input field values, angular takes care of that using ngModel directive

4.Angular also provides ngForm directive along with ngModel directive which automatucally tracks the form and form elements

Drawbacks

1. when it comes to unit testing, the form validation logic cannot be unit tested, only way to test the logic in html is used to run e2e testing

2. when it comes to handle complex forms

when?

   whenever we want to develop simple forms

Steps

1. Adding HTML form

     1. form-group and form-control are bootstrap classes

     2.  create name, email,phone text box

     3. Drop down menu called topics

     4. create radio button and check box and submit button

2. Binding data with ngForm

    1. Now we use <form> tag attaches with ngForm directive to track entire form controls

    2. To track each form control we use ngModel

    3. Along with ngModel we use name attribute, so that we can able to retrieve different form control values

    4. ngModelGroup directive is used the group the input controls, for eg address

3. Bind all data to model class

       - we have to bind all the html data to the model class, so that we can pass as a single object

       1. create a model class

             >ng g class User

       2. apply two way binding using [(ngModel)]

4. Form validation

      - Angular provides 3 classes to do validation based on the state

1. When u load the class for 1st time, u have not yet visited the form fields, so angular applies ng-untouched. If we visit the form control either by clicking or navigating angular applies ng-touched.

2. Angular also tracks if the value of form control changes. When u load the form for 1st time, the value is not changed so angular applies ng-pristine. If we change the value of form control, then angular applies ng-dirty

3. If the form control value is valid then angular applies ng-valid and if  it is invalid then angular applies ng-invalid

In order to apply all these classes, ngModel provides properties like untouched, touched, pristine,dirty,valid and invalid

How to access the validation properties?

      By creating reference to ngModel directive

when ur form is invalid we use property invalid. Bootstrap provides a class called "is-invalid" to handle invalid state

[class.is-invalid]="name.invalid && name.touched"

[class.d-none]="name.valid || name.untouched"

d-none- do  not show error message if form-control is valid and untouched

5. Form submission

Reactive Form /Model driven form

1. Much code and logic resides in the component class

2. No two way binding instead we need to react to user inputs to update the value, for that angular provides methods to update the form control values from component class

3. Reactive forms are suited for complex forms

4. since we write code in component class we can do unit testing using Jasmine and Karma

5. Custom validation for example, we want to validate password and confirm password field are same

6. Conditonal validation, based on some condition we want to do validation

7. we can create dynamic form fields, eg: create email fields dynamically

8. we have to import "ReactiveFormsModule"

Steps

1. create basic html form with username, password, confirmpassword and submit button

2. Creating the Form model

      1. Import ReactiveFormsModule inside app.module.ts - which gives access to classes to built reactive forms like FormGroup, FormControl

      2. In Reactive form, the form is represent as model in component class using FormGroup(ngForm)and FormControl(ngModel) class

#userForm="ngForm" -------- registrationForm=new FormGroup()

ngModel --------- FormControl

     3. Associate this model with the Html page using formGroup and formControlName directive

3. Nesting form groups (like ngModelGroup) - formGroupName

4. Setting default values programmtically

     using setValue() - load default values for all controls

     using patchValue()- load default values only for particular controls

5. We created Form model using FormGroup and FormControl classes. But by creating multiple FormControl instances manually becomes repeative, to avoid this Angular creates "FormBuilder" services to handle multiple form controls

     1.Inject FormBuilder inside the constructor and import it

     2. use FormBuilder instance to generate form controls using group()

6. Simple validation

     -- unlike template driven forms, in reactive forms we do validation in component class

     1. Use Validators class

7. Cross field validation (ie) validate between password and confirm password

AbstractControl - used to refer registrationForm(ie) html form

1. we create custom validator called Password.validator.ts

export function PasswordValidator(control: AbstractControl) :{

      [key:string]:boolean} | null {

      const password=control.get('password');

      const confirmPassword=control.get('confirmPassword');

if(password.pristine || confirmPassword.pristine){

    return null;

}

return password && confirmPassword && password.value!=confirmPassword.value ? {'misMatch':true}: null;

}

2.Add the custom validator to registrationForm

3. Apply error messages in HTML page

8. Conditional validation - we do validation based on some condition

  1. create email text and subscribe check box

<div class="form-group">

            <label>Email</label>

            <input type="text"  formControlName="email" class="form-control">

        </div>

        <div class="form-check mb-3">

            <input class="form-check-input" formControlName="subscribe" type="checkbox">

            <label class="form-check-label">

                Send me promotional offers

            </label>

        </div>

  2. we configure email and subscribe in component class

  3. We want when the checkbox is clicked then only email has to be validated

   To track the value of checkbox we use valueChanges(). Every formcontrol provides its current value as Observalble through valueChanges property. We subscribe to the observable and do conditional validation based on the form control value

  4. We want to subscribe in ngOnInit(), we get the value of subscribe checkbox using valueChanges which returns Observable

9. Dynamic Form fields

     We want to generate different text box for email based on  the number of email available

    1. FormArray class from ReactiveFormsModule to create dynamic list of controls

    2. Define alternateEmail as a FormArray in Component class

    3. Now we are going to create a button, whenever u clcik this button it wiil push the controls in to alternateEmails

    4. create addAlternateEmails() which push the controls into alternateEmails FormArray

    5. Iterate over FormArray and display form fields

<div formArrayName="alternateEmails"

\*ngFor="let email of alternateEmails.controls; let i=index">

    <input type="text" class="form-control my-1" [formControlName]="i">

</div>

10. Custom validation

      In case we enter the username as admin, it should display error msg

import { AbstractControl } from '@angular/forms';

export function ForbiddenNameValidator(control: AbstractControl):{[key: string]:any | null}

{

    const forbidden=/admin/.test(control.value);

    return forbidden ? {'forbiddenName':{value:control.value}}:null;

}

(blur)="validateTopic(topic.value)"

        (change)="validateTopic(topic.value)" #topic="ngModel"

          [class.is-invalid]="topicHasError && topic.touched"

<small class="text-danger" [class.d-none]="!topicHasError || topic.untouched">Please choose a topic</small>

Select control validation

    Topic control should be either angular or react or vue, if the user has "I am interested in" then we should display the error msg

    1. Add required attribute

Html page - collect all input to component class - send to backend (Spring/Spring Boot) through service- will connect to db

HttpClient from @angular/common/http - we can able to connect from angular to backend services like Spring or Spring Boot appl