**Trainer: tkmagesh77@gmail.com**

**Trainer mobile no:99019-11221**

what is angular?  
framework for building rich internet applications with better user experience.

earlier controller is on server side now the decision about interaction is happening on client side

earlier view is on server side ,now the html is generated on client side

model(Application state(data) ) is handled on client side

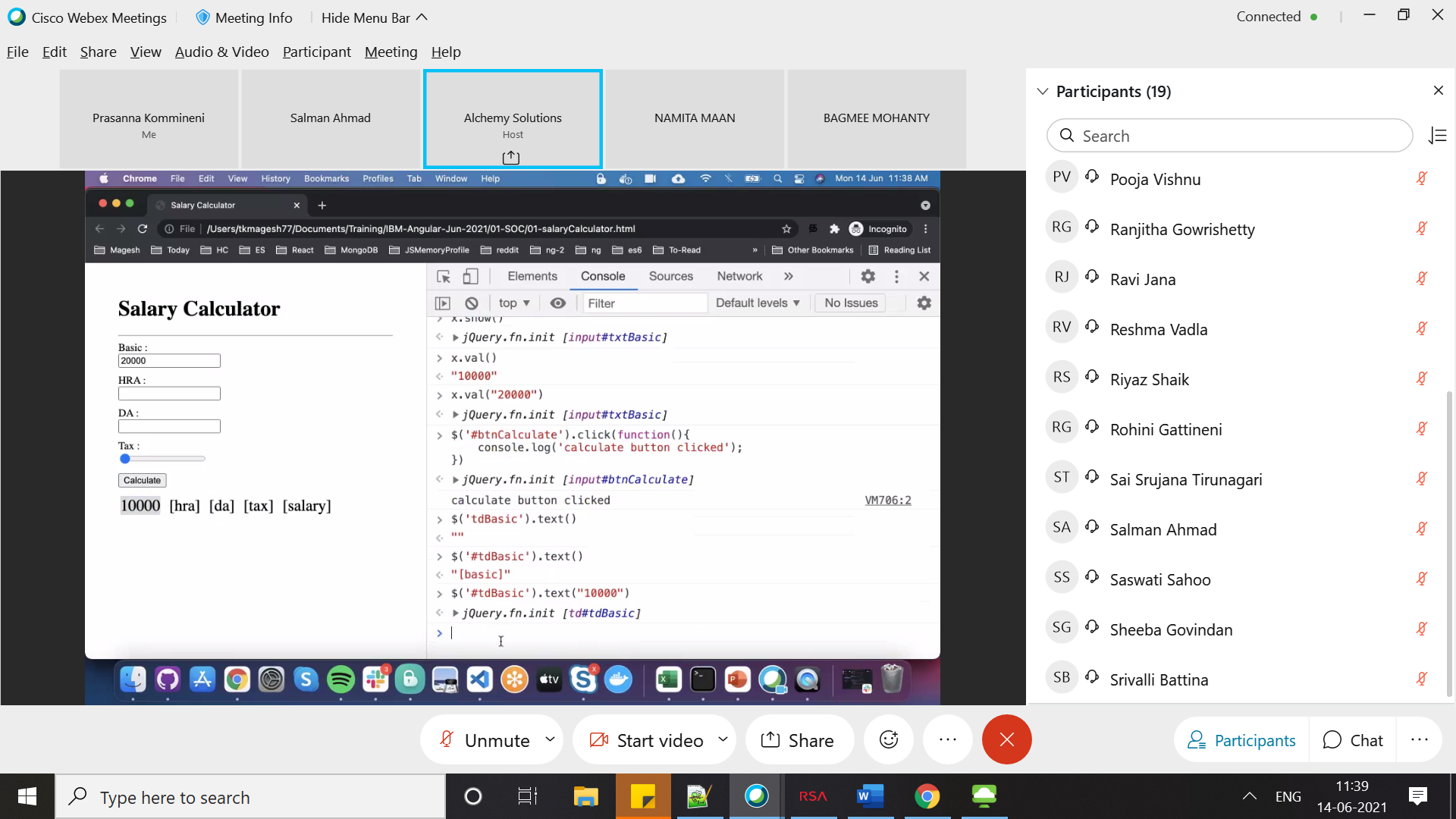
web applications are also able to work as desktop application

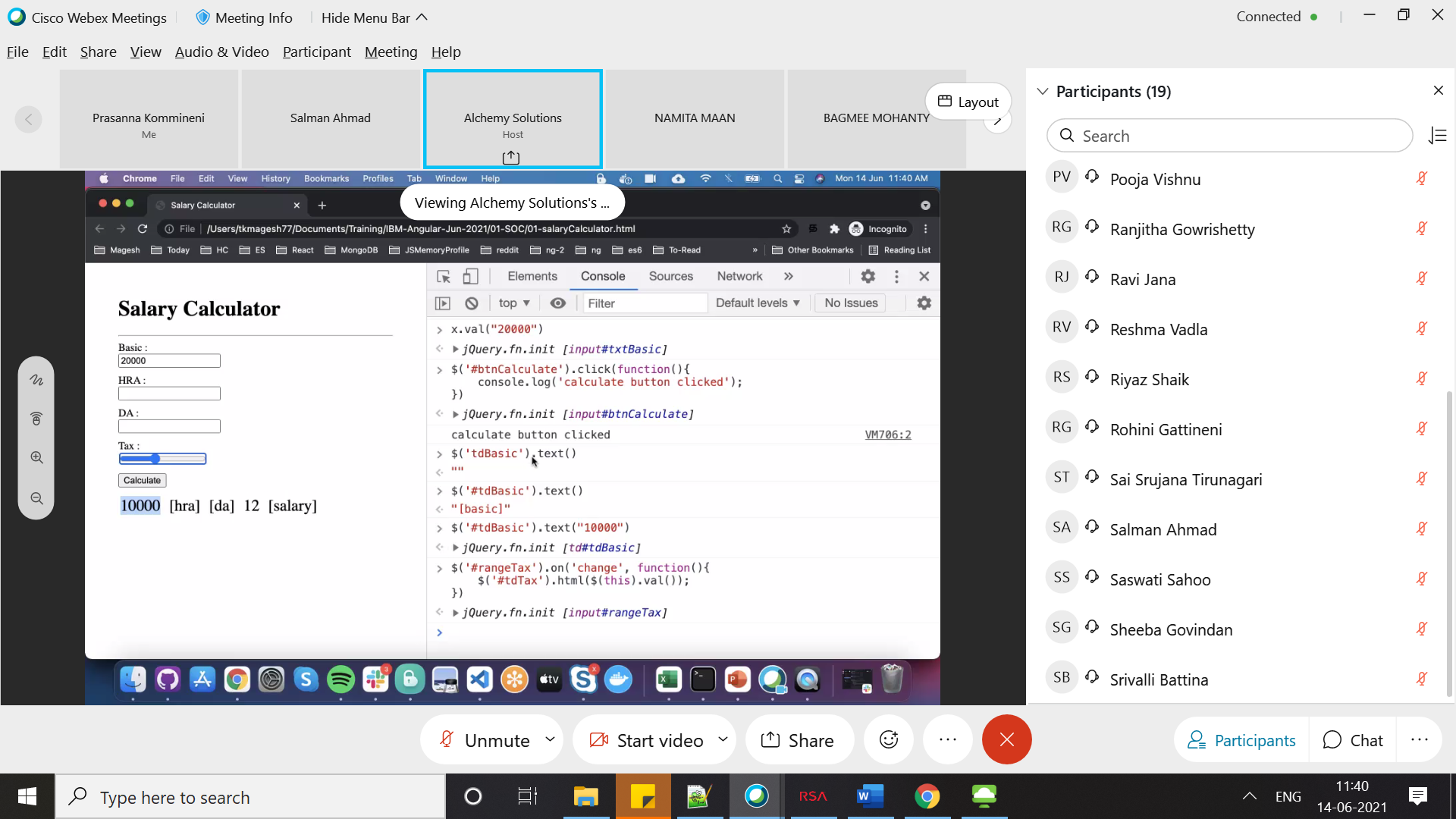
Build a layer of the application that run in the browser

As we are not reloading the page (it maintains the state of app) these are called single page applications

Jquery

Is dom manipulation library





<https://github.com/tkmagesh>

Steps to make applications maintainable

1. Separation Of concers: separate the functionality of uiand non ui
2. If A depends on B (only public interfaes) those are loosely coupled ,if it is also depending on private interfaces those are tightly coupled

Variable declaration

var (variables declared using are not block scoped)

let (variables declared with let are block scoped)

const (variables which value will not change)

Array Destructuring

Rest operator

let emp = {id:11,salary:1231}

let [id,salary] = emp;

<http://es6-features.org/>

**Angular Building Blocks**

**1)Module**

Directory of application entities (Directives ,Pipes ,Services ,components)

Helps in dependency Injection

Starting point of application(we cannot initialize the app if we don’t have module)

App Module is a class

**2)Components**

1)Used to build the views

2)Component encapsulates presentation(HTML,CSS) ,UI behavior (logic to execute when usen user interacts with the app) , UI state (data)

3) State and view synchronization is build in

4) components can be composed to create new components

**3)Directives**

Low level DOM manipulation logic (to add a style , remove a style from DOM node)

**4)Pipes**

Data transformation for presentation

**5)Service**

All non UI logic will be written in service

**Application setup**

Set up Typescript compiler

Set up Web server

Install the Angular libraries

Set up Build steps

**Angular cli**

helps in installing and set up the application(above steps)

**npm install @angular/cli -g**

**package.json**

contains metadata info of the application

dependencies of the application

**DevDependencies**

Which are used during development like typescript compiler and testing frameworks ,test runner (karma)

**Scripts**

Command which we can run

We have config file for each tool we are using in our app (karma , angular ,typescript)

Application code is in source folder

**Strt point of app is main.ts**

Ng serve command will include (poltfill.js , runtime.js)

Ng serve will use web server(internally)

Build will bundle (all the dependencies ) and load the respective js files in browser when we load

Bootstrap means starting something

**Decorator** is concept of typescript (used to add metadata to your app)

**@NgModule({})** to identify a class as a module

Ngmodule input contains different sections

Declarations (to register UI entities (components ,Directives ,Pipes))

Providers( Non UI Entities (Service))

Imports( module level dependencies(other modules this module depends on))

Bootstrap (top level component in the hierarchy(application))

When doing import and export we don’t metion file extension

Component class is devoreated with @Component decorator

Metadata input for @components decorator is

templateUrl:html content to be displayed for this component[]

stylesUrl :styeles to apply [‘./app.css’]

selector : tag with which we can refer this application

interpolation syntax

{{title }} 🡪 title is attribute in component

When data is changing angular will take care to change the UI

ES6 Modules

According to ES everything defined in JS/TS file is a private ,

to export the contents to outside we have to explicitly export

any public entity has to be exported explicitly

to use any public entity we need to import explicitly

we can have only one default export entity in Js/Ts

export default function a(){

methody body;

}

All exports in a file are (combined) and exported as module

(webpack development )Server uses the wbsockets to inform the chrome that there are changes in the application so it will reload the application

Angular searches with selector of Bootstrap component mentioned app module ,if it does not found the selector of bootstrap component, it will throw the error .

We can give multiple components in bootstrap but we have to place the respective selectors in index.html otherwise we will get the error

Using one component inside other component is alled as component composition

Types of directives

**Attribute Directives**(to change attribute of existing dom node)

ngClass ,ngStyle

(used with in []) [ngClsss]

**Structural Directive**(to change the structure of dom Tree by adding or removing dom nodes)

ngIf ,ngFor,ngSwitch (prefixed with \*)

**Pipe**

to transform the data to be displayed in html

|currency:param1:param2

**dependencyInjection**

Constructor(service:ServiceType)(this will create the class variable for us)

Above step is called dependency injection.

We have to provide the service in module to be able to inject in class(constructor).

Object.toString() will give [object Object]

Object is instance type ,object is the instance

**LocalStorage**

Key /value store

Key should be unique

Don’t persist sensitive data

Same origin policy

Eah origin will have 10MB storage

JSON.stringify(input) 🡪 to serialize the data

JSON.parse(data)-> to deserialize the data

setItem(key,value) getItem(key);

localstorage.clear()->to clear the storage

localstorage.length to get no of items

**if we don’t add injectable() decorator and we are trying to inject services in constructor we will get following error**

**Error: src/app/app.module.ts:19:15 - error NG2005: The class 'BugService' cannot be created via dependency injection, as it does not have an Angular decorator. This will result in an error at runtime.**

**Either add the @Injectable() decorator to 'BugService', or configure a different provider (such as a provider with 'useFactory').**

**If we are not using @injectable , we can use following syntax**

const bugServiceFactory = (bsn:bugServiceNew) => {

  return new BugService(bsn);

};

  {provide:BugService,useFactory:bugServiceFactory,deps:[bugServiceNew]}

Pipes are triggered only when state is changed.(if modify existing instance pipes will not be triggered).we have to update the instance to trigger the pipe.

Moment to display different data

Async Operation

We don’t wait for the application to complete

Asynch operations are stored in queue

Promise is an object to hold the result of async operation ang give it to subscriber when they are ready

**Async await**

a = ()=>{

let p = new Promise((resolveFn,rejectFn)=>

{

return resolveFn(1000);

}); return p;}

async function b(){ let result = await a(); console.log(result);}

()=>{

let p = new Promise((resolveFn,rejectFn)=>

{

return resolveFn(1000);

}); return p;}

**Generators in typescript**

**function \* oddNumGen(value){ for(let i of value){ if(i%2==1){yield i}}};**

**undefined**

**let o = [1,2,3,4,5,6,78,8];let j=oddNumGen(o);**

**undefined**

**j.next()**

**{value: 1, done: false}**

**Obesrvables are streams of events**

Operators for manipulating observable

Filter ,merge,pipe ,debounce,mergeMap,scan

Debounce it adds the delay (it releses the events in fixed intervals)

In angular we are handling aynchronousa activities using observable

A promise can handle only one result, observables canhandle stream of events

Npx json-server filename

Constructor should be responsible for injection logic only.

We have to write the logic in ngOninit method only.

**Test Cases**

**fDescribe will execute only the focused test(tests which are named as fdescribe).**

Xdescribe to ignore the tests

Describe(string,function);🡪like creating test class

It(string,function) 🡪 to write testing logic

Expect() 🡪Assertions

We use an API called Testbed provided by angular , we can create testing module for our app

TestBed.configureTestingModule to setup the components and providers required for the component

**Testcase for apiservice file**

import { TestBed } from "@angular/core/testing"

import { HttpClientTestingModule, HttpTestingController } from '@angular/common/http/testing';

import { BugApiService } from "./bugApi.service"

import { bugs as mockBugs } from '../mock-data/bugs';

describe("BugApi Service", () => {

let httpTestingController : HttpTestingController,

bugApi : BugApiService;

beforeEach(() => {

TestBed.configureTestingModule({

providers : [BugApiService],

imports : [ HttpClientTestingModule]

});

httpTestingController = TestBed.inject(HttpTestingController)

bugApi = TestBed.inject(BugApiService)

});

afterEach(() => {

httpTestingController.verify();

});

it("Should get all the bugs", () => {

bugApi

.getAll()

.subscribe(bugs => {

expect(bugs).toBeTruthy("No bugs are returned")

expect(bugs.length).toBe(3, "Not all the bugs are returned")

const bug = bugs.find(bug => bug.id == 1)

expect(bug).toBeDefined();

expect(bug?.name).toBe('Data integrity checks failed')

//verify only open bugs are returned by the getAll() call

});

const req = httpTestingController.expectOne('http://localhost:3000/bugs');

expect(req.request.method).toBe('GET');

req.flush(mockBugs);

});

it("Should make POST requests when saving a bug for bugs with id 0", () => {

const testBugData = {

id : 0,

name : 'Dummy Bug',

isClosed : false,

createdAt : new Date()

};

bugApi

.save(testBugData)

.subscribe(newBug => {

expect(newBug).toBeTruthy();

expect(newBug.id).toBe(100);

});

var req = httpTestingController.expectOne('http://localhost:3000/bugs');

expect(req.request.method).toBe('POST');

req.flush({

id : 100,

name : 'Dummy Bug',

isClosed : false,

createdAt : new Date()

});

})

it("Should make PUT requests when saving a bug for bugs with non zero id", () => {

const testBugData = {

id : 100,

name : 'Dummy Bug',

isClosed : false,

createdAt : new Date()

};

bugApi

.save(testBugData)

.subscribe(newBug => {

expect(newBug).toBeTruthy();

expect(newBug.id).toBe(100);

});

var req = httpTestingController.expectOne('http://localhost:3000/bugs/100');

expect(req.request.method).toBe('PUT');

req.flush({

id : 100,

name : 'Dummy Bug',

isClosed : false,

createdAt : new Date()

});

});

it("Should throw an error when deleting a bug that doesn't exist", () => {

const bugToRemove = {

id : 300,

name : 'Dummy Bug',

isClosed : false,

createdAt : new Date()

};

bugApi

.remove(bugToRemove)

.subscribe( bug => {

//verify the success criteria

}, err => {

expect(err).toBeTruthy();

expect(err.error.type).toBe('Bug does not exist');

});

const req = httpTestingController.expectOne('http://localhost:3000/bugs/300')

expect(req.request.method).toBe('DELETE');

req.error(new ErrorEvent("Bug does not exist"));

})

})