**Day 1**

UI ( User Interface )

https://[www.google.com](http://www.google.com) : URL

req(http/https)------🡪

Client Server

🡨-----res(http/https)------ HTML/HTML5

CSS/CSS3

JavaScript

JavaScript was object based interpreter scripting language.

Object based Vs Object Oriented

Compiler vs Interpreter

ES5 : ECMA Script (European Computer Manufacture Association

ES6 Features Using TypeScript :

npm install –g typescript (typescript)

tsc

Angular 1.x

HTML/CSS/JavaScript

Angular 2 to 8

HTML/CSS/TypeScript

jQuery

Node JS :

Node JS is a run time environment for the JavaScript files (library or framework).

MEAN Stack : MongoDB/MySQL Express Module Angular Node JS

MERN Stack MongoDB/MySQL Express Module React JS Node JS

In ES6 :

Declare the variable we can use var, let and const.

TypeScript : Typescript is a super set of JavaScript which support all features of ES6 or ES2015, E  
ES7 or ES8.

But browser doesn’t support typescript it require transpiler which help to convert typescript to JavaScript

ie typescript or babel

npm (node package manager) : it help to download the external modules.

Syntax

npm install –g moduleName

or

npm install modulename

npm install –g typescript

tsc

//var a=10;

//var a =20;

//let b =20;

//let b =40;

for(var i=0;i<100;i++){}

console.log(i);

for(let j=0;j<100;j++){}

console.log(j);

const c=100;

c=200;

DataTypes :

TypesScript support data types concept.

1. number
2. string
3. boolean
4. object
5. any

syntax

let variableName:dataType;

let a:number=100;

let a;

let b:number;

let c:number=30;

let d:any;

var a=[10,20,30,40,50,”Raj”,true];

for in loop

for(let variableName in arrayName) {

}

for(let variableName of arrayName) {

}

Function types

1. Normal function which takes more than one parameter

function empInfo(id,name,salary) {

return name;

}

function empINfo(id:number, name:string, salary:number):void {

}

Rest Operator : Rest Operator or parameter variable help to receive zero or 1 or more than one parameter as a arguments. Like a var-arg in Java.

function functionName(…variableName:dataType[]) {

}

Arrow function:

Syntax :

let functionName = (parameterList )=> function body

var functionName = (parameterList) => function body

let functionName = (parameterList )=> {function body}

var functionName = (parameterList) => {function body }

object : person, employee, bank

class :

function Person() {

this.pId=100;

this.pName=”Ravi”;

this.disInfo=function(){

console.log(“Pid is “+this.pId+” PName is ”+this.pName);

}

}

//Person();

let p1 = new Person();

p1.disInfo();

tsconfig.json : TypeScript configuration file.

**Inheritance** : Inheritance is use to inherits or acquire the properties and behavior of old class to

new class.

1. Single
2. Multilevel
3. Hierarchical

Typescript doesn’t support multiple inheritance using class. It support using interfaces.

class Employee {

}

class Manager extends Employee {

}

**String template :**

let info:string =”sfsafafasfas

afafsfas

afsafas

asfasf”

TypesScript modules :

Modules is a collection of variable, function, classes as well as interfaces.

Like package in java or namespace in C++.

Decorator : Decorator is a special function which

provide meta-data about the class, property or constructor.

Decorator is like a annotation in Java.

Meta-data means data about data.

­­Syntax

@DecoratorName({

Set of property

})

**Day 2**

**Angular 2-8 : (2,4,5,6,8)**

Angular is a **component based** **architecture** open source framework.

Angular is use to create SPA (Single Page Application). Angular 2-8 which help to create desktop, web as well as mobile application.

Library Vs Framework.

jQuery is a pre-defined library function.

Angular 1.x / 2-8 : Framework.

Library : Library provided lot of pre-defined function which internally connected to each other. So the developer or programmer has to call those library function to do the specific task. Library is not a standard doesn’t follow any design pattern.

$(“selector”).dotask()

**Architecture :** Architecture always recommended that divided our component or code base upon their functionality .

Client – Server

RMI Architecture

EJB Architecture

MVC Architecture

**Framework** : Framework contains set of pre-defined API(classes, interfaces or functions)

which internally connected to each other to perform specific task. Framework also known as

protocol or template but not final product. Framework internally follow standard rules.

Implementation of Design pattern is taken care by Framework.

Framework will call the code (Inject the Dependency injection of resources) in our application.

When we develop any application using framework 70 to 80 % task taken care by framework. So developer or programmer has to write hardly 20 to 30 code to make final product.

MVC : Model View Controller

Model --🡪 Normal Java classes or EJB or Spring Framework (Business Logic and DAO)

Controller -🡪 Servlet or Rest API (RestController)

View -🡪 HTML /JSP

RESt API

Angular 1.x : It is MVC based framework.

Angular 2 to 8 Controller is replaced by Component.

TypeScript /HTML/CSS

Google

Angular CLI (Command Line Interface).

npm install –g @angular/cli

ng (Next Generation for HTML) : A**ng**ular

ng new project-name

**ng new demo-app**

cd demo-app

ng serve or npm start

<http://localhost:4200>

ng generate component component-name

or

ng g c component-name

Data- binding : Data binding providing bridge between template and component or it is use

To share data between component to view and vice-versa.

2 types

1. One – way data binding
   1. String interpolation : Component --------🡪 View (Template)

Syntax

{{variableName}}

* 1. Property binding : Component ----🡪View (Template)

[propertyName]=”variableName”

* 1. Event biding : View ------------------🡪 Component

1. Two – way data binding

In Angular we use ngModel attribute to achieve two – way data binding

Component 🡨----------------------------🡪 View (HTML)

Syntax

[(ngModel)] = “VariableName”

[()] -🡪Banana bracket

ngModel is a pre-defined attribute part of FormsModules. So we have to import FormsModule in app.modules.ts file.

**Template Reference:**

**View 🡨------------------🡪 View**

<input type=”text” #nameRef (keyUp)=”0”>

{{nameRef.value}}

Directive : Directive is use to create the custom tags(user-defined) which help to add extra functionality to DOM(View/HTML).

Types of Directives

1. Component Directive

Component directive is use to control the view or part of the View in HTML page.

@Component({

selector:”tag-name”

template/templateUrl=””

})

export class Demo {

}

1. Structure Directive : Structure directive is use to add or remote DOM elements

(pre-defined tags or user-defined tags).

* 1. \*ngIf
  2. \*ngFor

1. Attribute Directive :Attribute directive is use to apply styling to template

ngStyle and ngClass

**Day 3 : Angular Forms and Services**

**ng new angular-forms-service**

**Angular Forms :** Angular provided totally two types of forms

1. Template Driven Forms
2. Model Driven Form or Reactive Forms

Template Driven Form : The flow of the form is view to component

View -----------🡪Component.

This type of form is simple to develop it easy for those people working on

html/html5. In this type of form we have to create the form reference using

ngForm attribute which is a part of FormsModule and bind the property with

ngModel attribute.

**Model Driven Form :** The flow the form is component to view. Angular provided

Pre-defined API (FormGroup and FormControl) **which help to bind the form and**

**form property** from component to view. This type of form is good the people from

Typescript background. This is good for complex form.

**Login Page** -🡪Form Group

TextField -🡪 form control

PasswordField -🡪Form control

Address-🡪Form Group

TextField -🡪Form control

TextField 🡪Form control

TextField 🡪Form control

**Template Driven Form Validation :**

**HTML5**

**<input type=”text” name=”user” required>**

**Angular Service :** If we write any business logic in component that code

Become local to that component means we can access that code within that

Component template. But if we can to achieve separation of concern then

We have to use model layer(service).

**Types of Service**

1. User defined service
   1. Creating object of service class using new keyword.
   2. Creating object of service class using DI (Dependency Injection)
2. Pre-defined service (Http Service)

IOC : Inversion of Control : It is concept or design pattern in place of creating

The object or any resource explicitly allow to create to container.

Only pull from a container whenever you want.

DI : Dependency injection :It is a implementation of IOC.

3 types of DI

1. Setter base DI
2. Constructor base DI
3. Interface base DI

**Pre-Define Service** : Angular provided pre-defined Http service which help to

Call http methods get(), post(), put() and delete()

Get() : Get the Resources

Post() : Create the Resources

Put() : update the Resources

Delete() : Delete the Resources

Angular provide **HttpClient** pre-defined API to call http methods.

So in Service layer we have to do the DI for HttpClient. HttpClient is a part of

**HttpClientModule** we have to import in app.modules.ts file.

**Day 4 :**

**Component Communication**

1. Passing the value from parent to child component

**@Input** decorator is responsible to receive the value from parent to

child Component.

1. Passing the value from child to parent component

**@ViewChild()** decorator is responsible to access the child component

Property and behavior

Or

**@Output** decorator is responsible to pass value from child to parent using

EventEmitter() API.

**Angular Routing:**

Angular Routing is use to navigate from one component’s template to another

Component’s template with or without conditions using path specified in

Routing modules.

JWT : Json Web Token :

**Auth Guard** : Auth Guard provided set of interfaces which help to validate JWT

Or unique ID before navigating to dashboard or home page of the application.

**Pipe :**

Angular provided set of pre-defined pipes(filter) which help to filter the

Data on view using String interpolation.