**HTTP Headers:**

Content-Type - Request

The type of data you are sending to server.

Content-Type - Response

Data sent by the server.

- MIME format,

text

plain- text/plain

xml- text/xml

json- text/json

html- text/html

css- text/css

image

image/jpg

image/png

audio

audio/mp3

audio/ogg

audio/wav

video

video/mp4

video/ogg

video/wmv

application

Does not belongs to all

Encoding data

Accept - Requesting to server to send the response in specific type.

Authorization - : "Scheme Value"

Basic Auth = > "Basic tokenValue"

Token Auth = > "Bearer tokenValue" (JWT)

Http Status Codes :

- 2xx

- 200 OK

- 201 Created (Location Header)

- 204 No Content (Ex. Delete/PUT record)

- 3xx (Redirection (Permanent/Temporary/Cashed (304)))

- 301

- 302

- 304

- 305

- 307

- 4xx (Client Error)

- 400 Bad Request

- 401 Unauthorized

- 403 Access Denied

- 404 Not Found

- 5xx (Server Error)

- 500 Internal Server Error

- Timeout error

Day 2

Bootstrap

Download from: getbootstrap.com

- Get Bootstrap using:

Source files

CDN urls

Package managers - npm, nuget, yarn, bower etc.

- Dependencies and DevDependencies

npm install --save <package1@version> <package2> <package3>

npm i -S package[@version]

npm i -S jQuery bootstrap

npm install --save -dev <package1@version> <package2> <package3>

npm i -D package[@version]

Install Both:

npm I bootstrap jquery --save

TypeScript

What is Typescript?

It is superset of JavaScript

It is compiled language

It is type safe

Fully compatible to ES6 features

Compiled by 'TSC' compiler

Control the compiler configurations using tsconfig.json

Set the Target version of JS in tsconfig.json

Set weather source map file is required or not (.map files)

Install typescript globally

step 1 npm install -g typescript

strp 2 tsc --init ( To create a tsconfig.json file)

step 3 tsc (To compile ts file)

Module

A typescript file that contains a set of exported members

An exported member can be class, interface, function and constant.

Export the members using **export** keyword and import them into another module using **import** keyword.

**Barrel** is a way of importing modules into another files.

A Barrel use an index.ts file that **exports a set of modules from a directory** and it can be imported using a single line.

Angular (SPA)

lazy loading

universal

**Angular React**

Framework Library

MVC pattern No pattern

Built in CLI No

Two way Binding No two way binding

IV Rendering engine DOM manipulation lib

Complete framework not - use third party library

Angular CLI:

step 1 npm install -g @angular/cli

ng --version

To Create the Project:

step 2 ng new projectName

Angular module

It encapsulates the

components - UI for the application

**.ts .html .css** .spec.ts(For testing)

directives

- used to add or remove behaviour of DOM element.

- Change the appearance.

pipes

- Used to format the data while printing in template file.

services

- Injectable objects (Angular DI) / reusable fuction codes.

import submodule(built in / custom)

A module is a typescript class decorated with @NgModule()

To run the application

compile ng serve -o

ng s -o

To Create new components

ng generate component <name>

ng g c <name>

**Directives:**

Structural directives

Starts with a \* symbol

\*ngFor, \*ngIf, \*ngSwitch

Attribute directives

Components directives

**Bindings:**

Interpolation – {{variable}}

Property binding – input binding [ ]

e.g.: [value] =”variable/value”

Event binding - output binding ( )

e.g.: (click) =”show()”

Two way binding – [( )]

**Pipes: S**ymbol is **-** |

Pipes are used for formatting data in templates.

Left side is the input value (Argument)

Syntax: {{input value | pipename[:args] }}

e.g.: {{name | uppercase}}

{{today | date:’dd-MM-yyyy’}}

{{salary | currency:’INR’}}

Build-in pipes

UpperCase

LowerCase

TitleCase

Json

Number

Currency

Date

Async – observable object

Custom pipes

To Create Custom pipes:

Ng generate pipe [pipes/pipename]

Ng g p [pipes/pipename]

To New Create Project

ng new <projectname> [option]

ng new eshop-spa --minimal --skip-tests --skip-install

-- minimal = this option can be minimal the files

Go on project directory and try to open VS Code

code . => press enter

To create modal/interface

ng g interface models/interfacename

ng g interface models/product

To create service

ng g service services/servicename

ng g service services/product

Services can be registered globally and locally.

Register globally in AppModule provider section.

**Observables and Promises:**

User for Asynchronous programming

**Promises:**

It is a built in feature of JS

Uses Request-response pattern

**Observables:**

Observable = Publish subscribe Pattern – hot star, youtube, facebook, realtime data processing.

Used for real-time data processing

Follows publish subscribe pattern

Subscribe for the data once and it updates the changes untill you unscbscribe.

Need to use RxJs library (ReactiveX)

**Forms In Angular**

Two approaches for Form creation:

Template driven Forms

Import FormsModule into AppModule from ‘@angular/forms’ package

We use HTML 5 validation attribute.

Need to use ngModel directive for two way data binding.

Every form control must have a **Name** attribute if we use ngModel.

A reference variable can also be used to access the form control (#)

Every control have some properties that defines the state-

Valid – invalid

Dirty – pristine (nothing typed/cleaned)

Touched – untouched

Form also have valid and invalid properties.

Reactive Forms

We use ReactiveFormModule from ‘@angular/forms’ package

We use services like FormBuilder to create a FormGroup.

FormGroup is a class used to create model for the form.

We need to use the directives like FormGroup and FormControlName

Validations are done in the model object (FormGroup) using Validations class.

Every control have some properties that defines the state-

Valid – invalid

Dirty – pristine (nothing typed/cleaned)

Touched – untouched

Both we can do Form validation and submission.

**JSON Server – For dummy server like api for data sharing**

To install server npm install -g json-server

To run the server Json-server --port 3000 --watch eShopBD.json

HTTP services

We need to use the HttpClientModule that provides the HttpClient service.

HttpClientModule is provided by ‘@angular/common/http’

It allows you to connect and fetch data from backend API server.

It provides GET, POST, PUT, DELETE and PATCH methods.

These methods are generic methods and it returns observable objects.

**Interceptors:**

**Resolver Service:**

**Life Cycle hooks:**

Constructer() executes before binding data to properties

ngOnInit() – OnInit interface

ngOnChanges() – OnChanges interface

ngOnDestroy() – onDestroy() interface

**Custom directive:**

ng g directive directives/

**Guard:**

Permission – giving the permission

**Lazy Loading: similar like MVC area**

ng generate module <areaName> --route <customerRouteName> --module <mainModuleName>

ng generate module customer --route customer-route --module app.module

To generate the components inside the area:

ng g c customers/components/profile

ng g c customers/components/orders

<http://localhost:4200/customers> Home page of customer

<http://localhost:4200/customers/profile> profile page of customer

<http://localhost:4200/customers/order> page of order

-------------------------------------------------------------\*\*\*\*\*\*\*\*\*--------------------------------------------------------------

**Mongo DB – 21st Oct 2019**

---------------------------------------------------------

It is unstructured, NoSQL database.

Stores the data in JSON format.

To store related info togetherly.

Unstructured Databases-

Documents (JSON) - **MongoDB**, SQL API(documentDB)

Key-Value/Tuple Pair - Azure table API, Redis Cash Database

Graph - Gremlin, Neo4j, (Adjest and vertises (Linked In))

Column family - Cassandra, Hadoop, Cloud data

It is open source driven by MongoDB community.

Onpremise NoSQL database that follows documents data storage.

Supports data indexing and partitioning (Sharding) for better performance.

Highly secured (multiple level authentication support)

It use BSON (Binary JSON) as the data storage format and JS as procedure and functions.

Installing MongoDB:

Install binary file .msi (installable), Run as a windows service.

Portable executable is available, download zip and extract. Need to run explicitly.

In the bin folder we can see mongodb server and client. (“MongoDB\Server\4.0\bin\”)

Server file: mongod.exe

Client Tool: mongo.exe (CUI)

GUI Client: VS code CosmosDB plugin, NoSQL booster for MongoDB, MongoCompass.

Migration Tools:

monogexport.exe

mongoimport.exe

mongodump.exe

mongorestore.exe

Installing BY ZIP:

Type below text in notepad:

"C:\Program Files\MongoDB\Server\4.0\bin\mongod.exe" --dbpath C:\Program Files\MongoDB\Server\4.0\data

Save this as

Mongodb://[hostname/ipaddress]:27017

mongodb://127.0.0.1:27017

Open cmd from mongo bin directory

And run

mongo.exe

use <dbname>

use eshop

* switched to db eshop done
* To check the db is created or not
  + db.stats();
* To get db collection names
  + db.getCollectionNames();

**Relational DB vs NoSQL:**

Database – database

Table – Collection/Container

Row – Document

Columns – Attributes

Every mongo dB document have a unique identifier with the name \_id.

A unique identifier

For Insert record:

insertOne()

insertMany()

**How to querying Data?**

We use find() for fetching data.

Conditions to fetch data:

Find(filter);

Find({id:3});

Find(filter, projection);

1. exclude
2. include

Sorting item

db.products.find({},{\_id:0, name:1, "availability": 1, "price": 1})

.sort({price:-1})

For ASC order use: 1

For DESC order use: -1

Query a fixed number of document

db.products.find({}, {\_id:0}).limit(4);

db.products.find().skip(4).limit(4);

Update documents/records:

db.colletion.update(filter, updateData, options)

db.products.update({availability:"Out of Stock"},

    {

        $set:{availability:"Available to Sale", quntity:10 }

    },

    { multi:true }

);

Add new attribute in existing record

db.products.update({quntity:10},

    {

        $set:{ expiryDate:"2019-09-21" }

    },

    { multi:true }

);

If update query not able to find this same record then insert new one.

db.products.update({name:"Pepsi"},

{

    $set:{id:12, quntity:4}

},

{upsert:true}

)

**How to use relational operators:**

Relational Operator: $le, $lte, $ge, $gte, $ne, $eq

Conditional Operator: $and, $or

Syntax:

(condition) AND (condition)

And(condition, condition)

Mongo $and(condition, condition)

**Array Operations:**

Iterate through an array of elements and push to array when use with $push

To avoid duplication we can use $addToSet

**db.students.update(**

**{name:"Arti"},**

**{$push:{marks:{$each:[25,30,35]}}}**

**)**

**Mongo DB Aggregation:**

Combine data from multiple documents.

3 types of aggregation:

Single purpose aggregation functions.

Count()

Distinct()

Aggregation pipeline.

An array of operation that executes one after another.

Result of one operation is the input of next operation.

We use operator like $match, $project, $group, $sort, $unwind etc.

Map-Reduce function (java script code)