**What is Type Script TS**

<https://www.typescriptlang.org/>

Advance version of JavaScript with some new functions and features

TS cannot execute directly to the browser a compiler used to convert ts to js first then executes

npm install -g typescript

tsc –v

create a new folder anywhere create some files

tsc index.ts

function sum(a:number, b){

    return a + b;

}

sum('10', '32');

**Core data types**

String, number, Boolean

let num:number = 12;

let str:string = "string";

let bool:boolean = false;

let define:any = 12;

**Objects**

let student:object = {

    id: 4,

    name: "John"

}

let student1:{id:number, name:string} = {

    id: 4,

    name: "John"

}

**Arrays**

let arr:number[] = [34,56, 90];

let arr1:string[] = ['john', 'tom', 'david'];

let arr2:any[] = [12, 'john', true];

**enum**

enum Role {

    ADMIN, EDITOR, SUBSCRIBER

}

**Union type**

let data: number | string = "12";

console.log(data);

**Type Alias in TypeScript**

type myType = number | string | undefined | boolean;

let a: myType  = 10;

let b: myType = '10';

let c: myType = undefined;

let d: myType = true;

**Function Type**

function sum():number {

    return 9 + 8;

}

console.log(sum());

**Void Functions**

function showMsg() {

    alert();

}

console.log(showMsg());

**Include and Exclude file folder**

npm tsc –init

tsconfig.ts

  "exclude": [ ],

  "include": [ ]

**Classes**

class Person {

    name:'';

    email:'';

    setName(name:string) {

        this.name = name;

        console.log(`Name is ${this.name}`);

    }

    getName(){

        return this.name;

    }

}

let obj = new Person();

obj.setName("John");

console.log(obj.getName());

**What is Angular**

<https://angular.io/cli>

* Angular front-end development framework
* Used to develop single page application (SAP)
* Developed and managed by Google
* Model View Controller (MVC) Framework
* Open Source
* NPM and CLI command line interface

**What we will learn**

* Installation
* Structure
* Basics
* MVC
* Advance
* Routing
* API
* Forms
* Including CSS

**Set up and Installation**

Node and NPM 🡺 <https://nodejs.org/en>

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

First Angular App

npm install -g @angular/cli

ng new my-app --no-standalone --routing --ssr=false

ng serve

**Note: (**in case if you faced execution policy issue run these commands**)**

get-ExecutionPolicy –list

set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy Unrestricted

**Files and Folder Structure**

Package.json

* Contains information about npm packages that we used or include

node\_modules

* All the dependencies that forced by the cli

Src folder

* app
* assets
* index.html
* main.js
* style.css
* package-lock.json
* angular.json
* tsconfig files

**Components**

<https://angular.io/guide/component-overview>

ng generate component navbar

ng generate component myList --inline-style

ng generate component myList1 --inline-template

ng generate component myList2 --inline-template --inline-style

**Modules**

<https://angular.io/guide/feature-modules>

ng generate module user-auth

ng generate component user-auth/Login

ng generate component user-auth/Register

Module contains more than one components

Import your module in app module to access it publically

Also you need to export your component from custom module

**Events**

<https://angular.io/guide/event-binding>

<input type="text" #inputVal (keyup)="inputKeyUp(inputVal.value)"  value="{{inputData}}"/>

<input type="text" #inputVal (keydown)="inputKeyUp(inputVal.value)"  value="{{inputData}}"/>

<button (click)="inputKeyUp(inputVal.value)">Click Here</button>

**Counter Example Exercise**

**Angular ngIf: Complete Guide**

<https://blog.angular-university.io/angular-ngif/>

<ng-container \*ngIf="userLoggedIn">

    .... visible only to authenticated users

</ng-container>

<div class="container" \*ngIf="courses.length; else noCourses">

    <h1>All Courses</h1>

</div>

<ng-template #noCourses>

    <h1>No courses available.</h1>

</ng-template>

**Toggle an items using if else exercise**

**Switch Case**

<https://angular.io/api/common/NgSwitchCase>

<container-element [ngSwitch]="color">

    <h1 \*ngSwitchCase="'red'">Red</h1>

    <h1 \*ngSwitchCase="'green'">green</h1>

    <h1 \*ngSwitchDefault>Default</h1>

</container-element>

**For Loop / Nested For Loop**

<https://blog.angular-university.io/angular-2-ngfor/>

  HEROES = [

    {id: 1, name:'Superman'},

    {id: 2, name:'Batman'},

    {id: 5, name:'BatGirl'},

    {id: 3, name:'Robin'},

    {id: 4, name:'Flash'}

  ];

<table>

    <tr \*ngFor="let hero of HEROES">

        <td> {{hero.id}} </td>

        <td> {{hero.name}} </td>

    </tr>

</table>

**Forms**

<https://blog.angular-university.io/introduction-to-angular-2-forms-template-driven-vs-model-driven/>

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

Also need to register it

<form #userForm="ngForm" (ngSubmit)="postData(userForm.value)">

   <div class="form-group">

       <label for="">Name</label>

       <input ngModel type="text" name="full\_name" class="form-control">

   </div>

   <input type="submit" name="" value="Submit" class="btn btn-success mt-3">

</form>

**Reactive form example**

import { NgForm, FormControl, FormGroup, Validators } from '@angular/forms';

  form = new FormGroup({

      "firstName": new FormControl("", Validators.required),

      "password": new FormControl("", Validators.required),

  });

  onSubmitModelBased() {

  console.log("reactive form submitted");

  console.log(this.form.value);

  }

  get firstName(){

    return this.form.get('firstName');

  }

<section class="sample-app-content">

    <h1>Reactive Form Example:</h1>

    <form [formGroup]="form" (ngSubmit)="onSubmitModelBased()">

        <p>

            <label>First Name:</label>

            <input type="text" formControlName="firstName">

            <span style="color: red;" \*ngIf="firstName && firstName.invalid && firstName.touched">Name Field required</span>

        </p>

        <p>

            <label>Password:</label>

            <input type="password" formControlName="password">

        </p>

        <p>

            <button type="submit" [disabled]="!form.valid">Submit</button>

        </p>

    </form>

</section>

**File Uploading**

  ifile:any;

  getFile(event:any){

    console.log(event.target.files[0]);

    this.ifile = event.target.files[0];

  }

  onSubmitModelBased() {

    console.log(this.form.value);

    this.newname = this.form.value;

    const forme = new FormData();

    forme.set('sfile', this.ifile);

    forme.set('formvalue', this.newname);

    this.http.post('http://localhost/angular-api/index.php', forme).subscribe( (data) => {

      console.log(data);

    } );

**Add Style**

<https://ng-bootstrap.github.io/#/home>

ng add @ng-bootstrap/ng-bootstrap

Adding Header Navigation

**Angular Material UI**

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

Much similar as bootstrap

**Daily task list using Angular exercise**

Passing data from parent component to child

Define data in parent component

data = 10;

Call it with child tags

<app-navbar [items]="data"></app-navbar>

Get this items array as input in child component

import { Component, Input } from '@angular/core';

@Component({

  selector: 'app-navbar',

  templateUrl: './navbar.component.html',

  styleUrl: './navbar.component.css'

})

export class NavbarComponent {

  @ Input() items = 0;

}

Use this items array in template file

{{items}}

Some more examples with loop and if else

Sending data from child to parent component

Define a function in parent component

accessChildData(item:string){

    console.log(item);

}

Call child template in parent

<app-navbar (fetchFromChild)="accessChildData($event)"></app-navbar>

Import output and event emitter in child component

import { Component, Input, Output, EventEmitter } from '@angular/core';

use output and emitter

@ Output() fetchFromChild = new EventEmitter<string>;

Call fetchFromChild function with event emitter method

(click)="fetchFromChild.emit(search.value)"

**Property Binding**

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

<input type="text" [(ngModel)]="name">

<h1>{{name}}</h1>

**Angular Pipes**

<https://angular.io/guide/pipes>

Some basic pipes

|  |  |
| --- | --- |
| **PIPES** | **DETAILS** |
| DatePipe | Formats a date value according to locale rules. |
| UpperCasePipe | Transforms text to all upper case. |
| LowerCasePipe | Transforms text to all lower case. |
| CurrencyPipe | Transforms a number to a currency string, formatted according to locale rules. |
| DecimalPipe | Transforms a number into a string with a decimal point, formatted according to locale rules. |
| PercentPipe | Transforms a number to a percentage string, formatted according to locale rules. |

<h1>{{'Text to UpperCase' | uppercase}}</h1>

<h1>{{'Text to LowerCase' | lowercase}}</h1>

<h1>{{ datee | date: 'short'}} </h1>

**Custom Pipes**

<https://angular.io/guide/pipes-custom-data-trans>

ng generate pipe pipes/myPipe

Call created pipe where you want

{{ 12 | myPipe }}

**Angular Routing**

<https://angular.io/guide/router>

<ul>

        <li><a href="/about">About</a></li>

        <li><a href="/contact">Contact</a></li>

</ul>

import { RouterModule, Routes } from '@angular/router';

const routes: Routes = [];

const routes: Routes = [

  {

    path: 'about',

    component: AboutComponent

  },

  {

    path: 'contact',

    component: ContactComponent

  }

];

<ul>

        <li><a routerLink="/about">About</a></li>

        <li><a routerLink="/contact">Contact</a></li>

</ul>

<router-outlet></router-outlet>

**Dynamic Routing**

<ul>

        <li><a routerLink="/user/1">John</a></li>

        <li><a routerLink="/user/2">Smith</a></li>

        <li><a routerLink="/user/3">Patric</a></li>

</ul>

const routes: Routes = [

  {

    path: 'user/:id',

    component: ContactComponent

  }

];

import { ActivatedRoute } from '@angular/router';

constructor(private route: ActivatedRoute){

      console.log(this.route.snapshot.paramMap.get('id'));

}

Wildcard routing

const routes: Routes = [

  {

    path: '\*\*', component: PageNotFoundComponentComponent

  }

];

Child Routing

<ul>

    <li><a routerLink="me">About Me</a></li>

    <li><a routerLink="us">About Us</a></li>

</ul>

<router-outlet></router-outlet>

const routes: Routes = [

  {

    path: 'about',

    component: AboutComponent,

    children: [

      {

        path: 'me',

        component: MeComponent

      },

      {

        path: 'us',

        component: UsComponent

      }

    ]

  }

];

**Services**

<https://angular.io/tutorial/tour-of-heroes/toh-pt4>

export class MyServiceService {

  constructor() { }

  users() {

    return [

      {

        id: 1,

        name: 'john',

        email: 'email@m.com'

      },

      {

        id: 2,

        name: 'john2',

        email: 'email2@m.com'

      }

    ];

  }

}

import { MyServiceService } from './my-service.service';

constructor (private userdata:MyServiceService) {

    console.log(this.userdata.users());

}

**Http Services**

<https://angular.io/tutorial/tour-of-heroes/toh-pt6>

import this module to app module file

import { HttpClientModule } from '@angular/common/http';

import { HttpClient} from '@angular/common/http';

constructor(private http: HttpClient) { }

httpUsers() {

    return this.http.get('https://jsonplaceholder.typicode.com/todos/');

  }

userhttp:any;

  constructor (private userdata:MyServiceService) {

    console.log(userdata.httpUsers().subscribe( (data) => {

      console.log(data);

    } ));

  }

**Save data to database using service**

**Interfaces**

<https://angular.io/tutorial/first-app/first-app-lesson-04>

interface datatype{

  id:number,

  name:string,

  email:string,

  address:any

}

saveData() {

    const datap:datatype = {

      id: 2,

      name: 'john',

      email: 'email@m.com',

      address: 'address'

    }

    return datap;

  }

**Lazy Loading**

<https://angular.io/guide/lazy-loading-ngmodules>

Child Router

const routes: Routes = [

  { path: 'login',component: LoginComponent },

  { path: 'list', component: ListComponent }

];

**Parent Router**

const routes: Routes = [

  {

    path: 'admin',

    loadChildren: () => import('./admin/admin.module').then(m => m.AdminModule)

  }

];

**Guard/Middleware in Angular**

ng generate guard auth

const routes: Routes = [

  { path: 'login', component: LoginComponent },

  { path: 'register', component: RegisterComponent },

  {

    path: 'dashboard',

    component: DashboardComponent,

    canActivate: [authGuard]

  }

];

**Pagination**

<https://www.npmjs.com/package/ngx-pagination>

import {NgxPaginationModule} from 'ngx-pagination';

**Client Side Pagination**

<li \*ngFor=" let rec of users | paginate : { itemsPerPage: 10, currentPage:} "></li>

**Server Side Pagination**

<li \*ngFor=" let rec of users | paginate : { itemsPerPage: 10, currentPage: page, totalItems: 40 } "></li>

**CRUD Using Pagination**

**Build & Deploy to Github**

* ng build
* create a git repository
* git init 🡺 initialize git