## Create First Angular 4 application

**Getting Typescript**

First install node.js from the website and get **TypeScript** using the following command

>npm install –g typescript

Install **Angular cli**

>npm install –g angular-cli

$ ng new angular4\_hello\_world

You get the message “Installing..” Wait to see the message “Installed

To check if it is angular 4 type

$ng –v

@angular/cli: 1.0.1

node: 6.10.2

os: darwin x64

@angular/common: 4.1.0

@angular/compiler: 4.1.0

@angular/core: 4.1.0

@angular/forms: 4.1.0

@angular/http: 4.1.0

@angular/platform-browser: 4.1.0

@angular/platform-browser-dynamic: 4.1.0

@angular/router: 4.1.0

@angular/cli: 1.0.1

@angular/compiler-cli: 4.1.0

Open using VS Code –see only the needed one

$ ng serve

<http://localhost:4200>

### Adding a component

$ ng generate component hello-world

Add the following line to app.component.html within the h1 tag.

<app-hello-world></app-hello-world>

### Adding Data to the Component

$ ng generate user-item

export class UserItemComponent implements OnInit {

name:String; //to be added

constructor() {

this.name = "John Smith"; //to be added

}

ngOnInit() {

}

}

In the user-item.component.html add {{name}}.

### Working with Arrays

Create a new component user-list

$ ng generate user-list

names:String[];

constructor() {

this.names = ["John Smith","William Beck","Joe Mathew"];

}

Add the template code to user-list.component

<ul>

<li \*ngFor="let name of names">{{name}}Test</li>

</ul>

### Accept Inputs

Update User Item class to

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

and within in the constructor class

@Input() name: String; //added

Update User List component

<ul>

<app-user-item \*ngFor="let name of names"

[name]="name"></app-user-item>

</ul>

## Reddit application

Create a new application

$ng new anuglar2\_reddit

Copy the images folder to src/assets

Overwrite index.html and styles.css

Copy vendor under app folder

* Analyse the code.
* Copy the code segment from snippet 1 to app.component.html
* Add the following line before the last line

<button (click)="addArticle(newtitle, newlink)"

class="ui positive right floated button">

Submit link

</button>

Add #newtitle and #newlink to the input components.

Add the addArticle() to the app.component.ts class within export

export class AppComponent {

addArticle(title: HTMLInputElement, link: HTMLInputElement): boolean {

console.log(`Adding article title: ${title.value} and link: ${link.value}`);

return false;

}}

### Create Article Component

$ng generate component article

1. Copy the snippet2 to article.component.html
2. Add the host to article.component.ts . row is the css class

host: {

class: 'row'

}

1. Add the following lines to Add.component.html to include the article component.

<div class="ui grid posts">

<app-article>

</app-article>

</div>

1. Check the output to see the article been added below the form. On clicking the votes the page refreshes. Because event bubbles up. So add “return false; “ in the votesup and votesdown function and change the return type to Boolean.

### Move the model to Article.model.ts

export class Article{

votes: number;

title: string;

link: string;

constructor( title: string,link: string, votes?:number)

{

this.votes=votes || 0;

this.title=title;

this.link=link;

}

votesup():void{

this.votes+=1;

}

votesdown():void{

this.votes-=1;

}

}

1. Change the component to include Article

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

**import {Article} from './article.model';**

@Component({

selector: 'app-article',

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

styleUrls: ['./article.component.css'],

host: {

class: 'row'

}})

**export class ArticleComponent implements OnInit {**

**article: Article;**

**constructor() {**

**this.article = new Article("Angular 2",**[**http://angular.io**](http://angular.io)**,10);**

**}**

**voteup():boolean{**

**this.article.votesup();**

**return false;**

**}**

**votedown():boolean{**

**this.article.votesdown();**

**return false;**

**}**

ngOnInit() {

}}

Reload and check the browser.

### Update the article to Article[] and add three articles in the constructor in the App.component.ts

And update the addArticle to push the values to this array

export class AppComponent {

**articles: Article[];**

**constructor() {**

**this.articles =[ new Article("Angular 2",**[**http://angular.io**](http://angular.io)**,10),**

**new Article("Angular - Up and Running",**[**http://angular.io**](http://angular.io)**,6),**

**new Article("Angular 2 Training",**[**http://angular.io**](http://angular.io)**,8)];**

**}**

**addArticle(title: HTMLInputElement, link: HTMLInputElement): boolean**

**{**

**this.articles.push(new Article(0, title.value,link.));**

**title.value ='';**

**link.value ='';**

**console.log(`Adding article title: ${title.value} and link: ${link.value}`);**

**return false;**

**}}**

### Update the Article component to take the Input

**@Input() article: Article; //(do not forget to add the import under core)**

Remove the constructor entry.

Update the App.Component.html

<app-article \*ngFor="let myArticle of articles" [article]="myArticle">

</app-article>

Reload and check the output. We get the desired output.

## Create a Product Application

### Create an Inventory App like earlier

$ng new InventoryApp

$ng generate ProductList

### Create Product.model.ts

class Product {

// public means there is a public variable in this class

constructor(

public sku: string,

public name: string,

public imageUrl: string,

public department: string[],

public price: number) {

}

}

Update the App.component.ts by copy and pasting Snippet 1

### Update the ProductList , add instance variables

@Input() productList: Product[];

@Output() OnProductSelected: EventEmitter<Product>;

currentProduct: Product;

Instantiate the OnProductSelected in the constructor

this.OnProductSelected = new EventEmitter<Product>();

### Add the following to Product-list-component.html

### <div class="product-lists">

### <app-product-row \*ngFor="let myproduct of productList"

### [product]="myproduct"

### (click)="clicked(myproduct)"

### [class.selected]="isSelected(myproduct)" class="product">

### </app-product-row >

### </div>

### Now define the two methods click and isSelected in the ProductList Component

clicked(myProduct):void{

/\*this.currentProduct = myProduct;

this.OnProductSelected.emit(myProduct); \*/ //add after running the code at the end.

console.log("Product is selected"+myProduct);

}

isSelected(product: Product): boolean {

if (!product || !this.currentProduct) {

return false;

}

return product.sku === this.currentProduct.sku;

}

### Create component ProductRow

$ng generate ProductRow

Copy Snippet 3 to the css.

Update the following line

@Input() product:Product; to the class

### Generate ProductImage, ProductDisplay and ProductDepartment

**ProductDepartment**

inputs: ['product'],

In the Controller

product: Product;

**html**

<span \*ngFor="let name of product.department; let i=index">

<a href="#">{{ name }}</a>

<span>{{i < (product.department.length-1) ? '>' : ''}}</span>

</span>

**ProductDisplay**

@Input() price:number;

**html**

<div class="price-display">${{ price }}</div>

**ProductImage**

@Input() product: Product;

**html**

<img class="product-image" [src]="product.imageUrl">

## Http Application

New-http.component.ts

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

import { Http, Response, URLSearchParams } from '@angular/http';

@Component({

selector: 'app-new-http',

templateUrl: './new-http.component.html',

styleUrls: ['./new-http.component.css']

})

export class NewHttpComponent implements OnInit {

ngOnInit() {

}

data: Object;

loading: boolean;

constructor(private http: Http) {

}

makeRequest(): void {

this.loading = true;

//Angular 2 way

const params = new URLSearchParams();

params.append('postId','1');

this.loading = true;

this.http

.request('https://jsonplaceholder.typicode.com/comments',

{search: params})

.subscribe((res: Response) => {

this.data = res.json();

this.loading = false;

});

// Angular 4 way

/\* this.http

.request('https://jsonplaceholder.typicode.com/comments',

{params:{postId: 1}})

.subscribe((res: Response) => {

this.data = res.json();

this.loading = false;

});

\*/}}

New-http.component.html

<h2>Basic Request</h2>

<button type="button" (click)="makeRequest()">Make Request</button>

<div \*ngIf="loading">loading...</div>

<pre>{{data | json}}</pre>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Create a Route Application

Create a project using Angular cli

$ng new RoutingApp

**Include the necessary import**

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

**create a file constants.ts**

const routes: Routes = [

{ path: '', redirectTo: 'home', pathMatch: 'full' },

{ path: 'home', component: HomeComponent },

{ path: 'about', component: AboutComponent },

{ path: 'contact', component: ContactComponent },

{ path: 'contactus', redirectTo: 'contact' },

];

Copy the Components folder from the participant assets and import all the components to remove the errors and add in the declaration for the NgModule

**Install the Routes**

Add the following line in the ngModule imports

RouterModule.forRoot(routes) *// <-- routes*

Copy Snippets to app.component.html

Run and you should be able to see the application.

On navigating there is no hashbang in the url and that is the default Angular 2 behavior. To change to #.

Add the imports and use it in the providers

**import** {LocationStrategy, HashLocationStrategy} from '@angular/common';

providers: [

{ provide: LocationStrategy, useClass: HashLocationStrategy }

]

## Login Protected Router App

In this we will use Bootstrap

npm install --save @ng-bootstrap/ng-bootstrap

npm install ng2-bootstrap bootstrap --save

In module.ts add

import {NgbModule} from '@ng-bootstrap/ng-bootstrap';

and add the following lines in the imports

NgbModule.forRoot()

Let us write a service responsible for authentication

### Create AuthService

app/ts/services/AuthService.ts

import { Injectable } from '@angular/core';

@Injectable()

export class AuthService {

login(user: string, password: string): boolean {

if (user === 'user' && password === 'password') {

localStorage.setItem('username', user);

return true;

}

return false;

}

}

Add the logout, getUser, isLoggedIn function

logout(): any {

localStorage.removeItem('username');

}

getUser(): any {

return localStorage.getItem('username');

}

isLoggedIn(): boolean {

return this.getUser() !== null;

}

Copy the components folder and place it in under ts folder

Create a folder guards under ts and create a file **loggedin.guard.ts**

import { Injectable } from '@angular/core';

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

import { AuthService } from '../services/AuthService';

@Injectable()

export class LoggedInGuard implements CanActivate {

constructor(private authService: AuthService) {}

canActivate(): boolean {

return this.authService.isLoggedIn();

}

}

Add the following lines in app.module.ts

const routes: Routes = [

{ path: '', redirectTo: 'home', pathMatch: 'full' },

{ path: 'home', component: HomeComponent },

{ path: 'about', component: AboutComponent },

{ path: 'contact', component: ContactComponent },

{ path: 'protected', component: ProtectedComponent,

canActivate: [LoggedInGuard]}

];

* Exercise for participants. See the errors and add the necessary import statements and style the bootstrap components.
* Refer the presentation and add cities in the ContactComponent, On clicking a city, it should navigate to protected content with the id displayed.

**Nested Routes**

Create a new angular2-nested-routing.

Copy and show the demo

Redux and Angular (If this is included in the course)

**app-state.ts**

export interface AppState{

counter: number;

}

**app-store.ts**

import { OpaqueToken } from '@angular/core'

export const AppStore = new OpaqueToken('App.Store');

**counter-action-creators.ts**

import { Action} from 'redux'

import { constants} from './constants'

export class CounterActionsCreator {

static increment( ):Action{

return { type: constants.INCREMENT}}

static decrement( ):Action{

return { type: constants.DECREMENT} } }

**counter-reducer.ts**

import { AppState } from './app-state';

import { CounterActionsCreator } from './counter-actions-creator';

import { constants} from './constants'

import { Reducer, Action} from 'redux';

let initialState: AppState = { counter: 0 };

// Create our reducer that will handle changes to the state

export const counterReducer: Reducer<AppState> =

(state: AppState = initialState, action: Action): AppState => {

switch (action.type) {

case constants.INCREMENT:

return Object.assign({}, state, { counter: state.counter + 1 });

case constants.DECREMENT:

return Object.assign({}, state, { counter: state.counter - 1 });

default:

return state;

}

};