Start the new angular application

1)download latest node.js from [www.nodejs.org.en](http://www.nodejs.org.en)

2)now go to [www.cli.angular.io](http://www.cli.angular.io) and install cli

3) install angular/cli 🡪**npm install -g @angular/cli ((((this is where you create your project)))**

4)create a new angular project 🡪 go to folder where you want to create your project using cd

5)create a project in this folder🡪**ng new PROJECT-NAME**

6)go to this project 🡪 **cd PROJECT-FOLDER**

**###if you get this error[[[Versions of @angular/compiler-cli and typescript could not be determined.The most common reason for this is a broken npm install.**

**Please make sure your package.json contains both @angular/compiler-cli and typescript**

**In devDependencies, then delete node\_modules and package-lock.json (if you have one) and**

**run npm install again.]]] then try this command** npm install --dev

7)run the project 🡪 **ng serve ( every time you start project you need to start the cli using this command) (first go to the project folder and then run this command)**

**8) run ng serve again**

9)go to browser 🡪 **localhost:4200** 🡪 (here we can see our project)

Adding bootstrap to your project

1. Download bootstrap to the folder where your project is(((for each project you have to install bootstrap ))) 🡺 **npm install - -save bootstrap@3**
2. Now it is download into the folder (**node\_modules** (this is name of folder where bootstrap is installed)) 🡺go to **.angular-cli.json** file in the folder 🡺 open it 🡺at **line #22** in **style** array write following path for bootstrap🡺 **“../node\_modules/bootstrap/dist/css/bootstrap.min.css”** (this means bootstrap is linked to the project (this we have to do for all the project))
3. Run **ng serve** again to successfully start the bootstrap to this project

­­­­how angular works

1. **ng serve** starts **main.ts** (import { **AppModule** } from './app/app.module';

platformBrowserDynamic().bootstrapModule(**AppModule**)

1. from **main.ts** it starts **app.module.ts**  (bootstrap: [**AppComponent**])
2. from **app.module.ts** it will assign to AppComponent (((here we have all the app.component files)))
3. from AppComponent it will goto **app.component.ts** (((@Component({

selector: **'app-root'**,

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

styleUrls: ['./app.component.css']

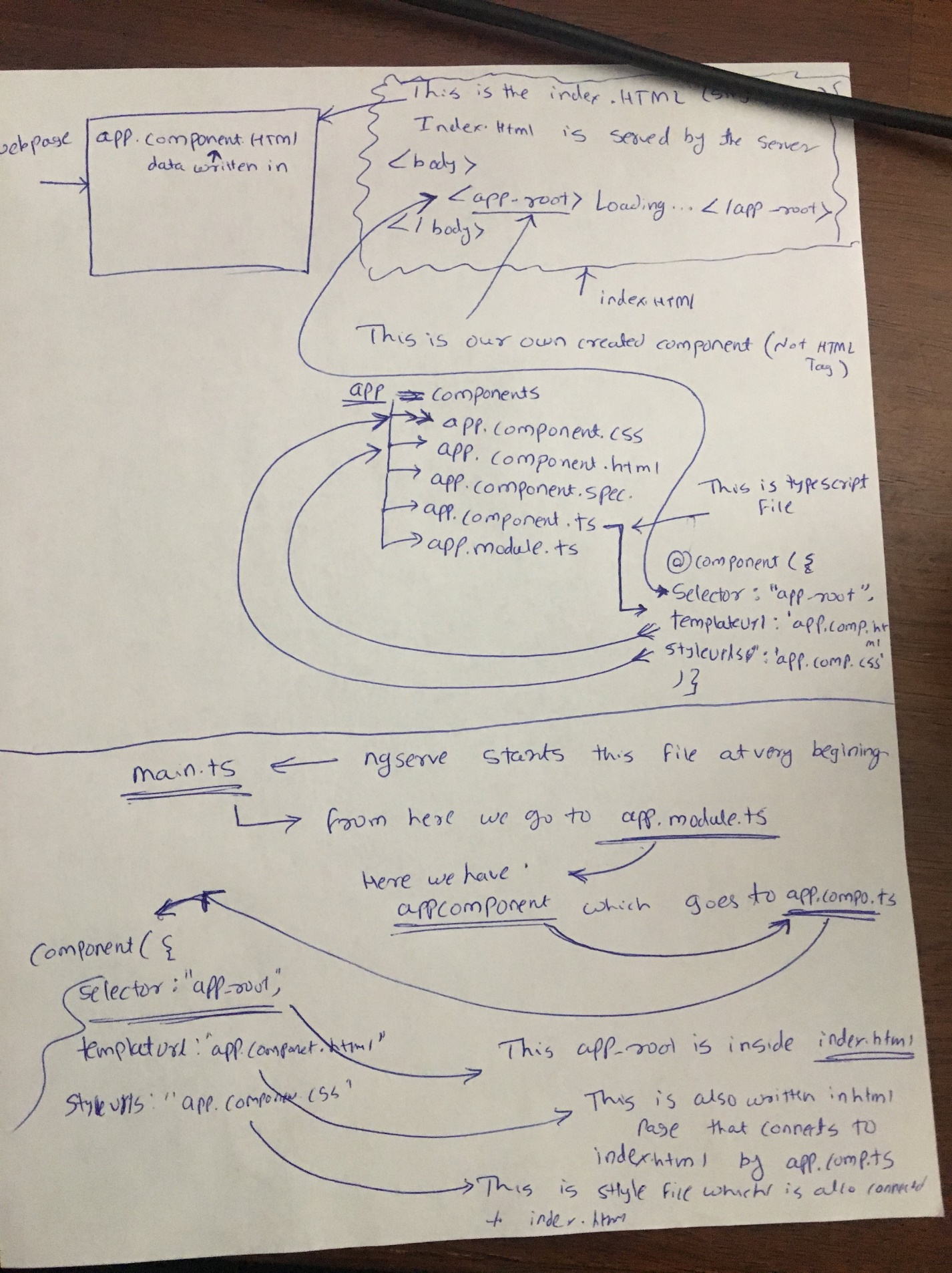
})

1. from **app.component.ts**  it will go to **index.html** where **app-root** works (((<body>

<**app-root**> </app-root>

</body>

Also, it will run **app.componetn.html** and **app.component.css** and this will be shown in index.html in browser (((this is the single page application)))



Create a Component (this is need for each part of the body)

1. create a folder for each component inside app folder 🡺 app/folder-name (e.g. server)
2. create a file inside this folder 🡺 app/server/**server.component.ts** (this is a typescript file for component)

import { **Component** } from '@angular/core'; 🡪imoprt component from @angular/core

@Component({ 🡪 this is main component decorator

selector: **'app-server'**, 🡪this selector is will be used in module.ts

templateUrl:'**./server.component.html'** 🡪this is html file for this component

})

export class ServerComponent{ 🡪created class of component for outside use

}

1. Create **server.component.html** file inside this server folder
2. Now use this component by adding it **into app.module.ts 🡺**

import { ServerComponent } from '**./server/server.component**'; 🡪 import **ServerComponent** from **server.component.ts** file

declarations: [ 🡪declare all the component here

AppComponent,

**ServerComponent 🡪** declare the ServerComponent class here for component use

],

1. Add **server.component.html** file to the browser

[[[important : we have **index.html** file, where **app-root** is used for body. Now to show **app.component.html** file’s content, we create a selector inside **app.component.ts**. This is how we use app.component.html file to display in index.html file]]]]

Now to display **server.component.html** file’s content in app.component.html, we have to use selector created in **server.component.ts** inside app.component.html

Adding new component using terminal by ng generate component component-name

1. Keep ng serve terminal running and Open new terminal
2. Use command **ng generate component component-name** or **ng g c component-name**
3. This will create a new folder inside the app folder and it also create 4 new files (just like app folder have 4 file)
4. This command also write inside the **app-module.ts** and import this component as well as add this component to the declaration array
5. Everything is done by this command. we only need to use the selector inside the **app.component.html**  to display it on the browser

Creating a directive using command 🡺 **ng g d 🡪** this will create a directive and write all the required format inside the file

Nested component

1. Create a new component in the same folder where you want to use it
2. Do everything like above but no need to write this selector inside **app.component.html**
3. Use this selector to the html component where you want to use it

**Coding notes from Udemy course**

* String interpolation 🡺 we use {{ variable }} in **template** file and we assign variable in **.ts** file
* [(ngModel)]= ‘name ’🡺 this is two-way databinding and it is used to bind data to html template and component.ts file. For that you need to use the same name given as the ngModel like here ‘name’ to the component.ts and then accessing this name you can use the two-way databinding. This will change the value on both side and that’s the concept of two way databinding.
* Property binding (html property like disabled ) 🡺 [disabled] = “function” (((here the function is assign in typescript of that same component)))
* Event binding 🡺 syntax is for onClick 🡪 (name-of-event) = “function()” (((name-of-event is click and function is what you want to do )))
* \*ngIf 🡺 this is an if statement in angular 🡪<p \*ngIf = “this needs to be true ; else marker” > here its paragraph which show depends on the if condition</p>
* \*ngFor🡺 this is a for statement in angular🡪 <p \*ngFor = “ let item of array; let i = index”>this is the for loop which run from all the item inside array and assign i to its index value</p>
* [ngStyle]🡺 this is for change the style of element 🡪<p [ngStyle] = “{backgroundColor: red}” > this will change the color of paragraph to red </p>
* [ngStyle] with \*ngIf 🡺to change the style of content according if condition🡪<p [ngStyle] =”{ backgroundColor: contentItem ==5 ? ‘blue’ : ‘transparent ’}”> this will change the color of contentItem to blue and make it transparent if it is equal to 5
* [ngClass]🡺this is for making changes inside the element 🡪<p [ngClass] = “{className : contentInside}”> this will change the style of the content inside the paragraph according to the class defined in the .ts file</p>
* @Input() 🡺 this is used when we are getting value from the parent component and then it will be displayed on the html template 🡪 parent component🡪 <app-childcomponent [value]=” this\_value” ></app-childcomponent> 🡪 child component.ts🡪 @Input() valueName : type; 🡪 child component.html 🡪 {{valueName.attribute}}

***Routing***

* routes: Routes =[{ path: ‘/location’, component: ComponentName}] 🡺 this is how we store the path for all the component inside the appModule (or we can create a different module and put everything inside it and import it in the appModule🡺 we can also show the nested component here using the childRoutes 🡪 routes: Routed= [{path: ‘/locationName’, component: componentName, children: childRoutes}] 🡪 then we can create new childRoutes and give path to all the children component
* <a [routerLink] = [‘/home’, ‘anything like string or number that will go and work as address bar’]> Home<a> 🡺 this is how we assign a link to the anchor tag (button) 🡺 to display the component inside the appComponent we have to add a tag <router-outlet> <router-applet> inside this we will show our components
* RouterModule.forRoot(appRoutes) 🡺 this is how we register our appRoutes array which contains all the routes inside the appModule
* <router-outlet></router-outlet> 🡺 this directive is used to display our component inside the template (we can remove our <app-component></appComponent> and instead we use < router-outlet></router-outlet>
* Constructor(private route: Router) ; then this.route.navigate([‘server’,’nest link’]) 🡺 this is how we going to navigate the link using the router method Navigate
* Constructor(private route: *activatedRoute*, private router: Router) ; then this.router.navigate([‘servers’), {relativeTo : this.route}) 🡺 this *activatedRoute* will give us the current route of the page we are at and then using the navigate and ralativeTo will attach this ‘servers’ to this activated path
* Constructor(private route: *activatedRoute)*  ; user : {id: number, name: string} ; then this.user = { id : this.route.snapshot.params[‘id’], name: this.route.snapshot.params[‘name’] } 🡺 this will give the value of id and name from the current address bar as it is assign inside the appModule and in that particular component path

**Coding Example for reference**

**🡺User Input 🡪** template: <input #box (keyup)="onKey(box.value)">

<p>{{values}}</p>

🡪 component.ts: values = ' ';

onKey(value: string){

this.values = value; }

=========================================================================================

**🡺User Input 🡪** template: <input (keyup)="onKey($event)">

<p>{{values}}</p>

🡪 component.ts: values = '';

onKey(event: any){

this.values = event.target.value ; }

==========================================================================================

**🡺User Input 🡪** template: <**input** name="title" #newtitle>

<**input** name="link" #newlink>

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

🡪 component.ts: *addArticle(title: HTMLInputElement, link: HTMLInputElement)**{*

*console.log(title.value* *link.value);}*

*==========================================================================================*

🡺**@Input()** 🡪 child-Component🡪 template: <h3>{{elementArtical.title}}</h3>

🡪 component.ts: @Input() elementArtical :{title:string,link:string};

**🡪parent-component🡪** template: <app-childComponent [elementArtical]="elementData” ></app-childComponent>

🡪 component.ts: elementData=”any string that want to display in child component”

============================================================================================

**Notes from angular book**

1. Page no. 22 🡪 syntax 🡪*< element \*ngFor =”let name of names”> {{name}}</element>* 🡪 this is a structural directive 🡪 name is single element of names array

Page no. 35,36 🡪 check how to use user input without using $event 🡪syntax🡪 <**input** #newtitle> <**button** (click)="addArticle()"

1. Page no. 223 🡪 <base href=”/”> 🡪 this is to show all the images and data required is inside the ‘/’ directory of that particular component

***Important terms required for angular coding***

* Events🡺 When a user presses and releases a key, the **keyup**event occurs.
  + 🡺