***Angular Notes***

1. Angular is a javascript framework which allows you to create reactive single page applications.

2. Updating npm:

3. Run [sudo] npm install -g npm (sudo is only required on Mac/ Linux)

Updating CLI :

[sudo] npm uninstall -g angular-cli @angular/cli

npm cache verify

[sudo] npm install -g @angular/cli@latest

4. CLI is the recommeded and best way of creating angular projects because angular projects are actually a bit more elaborate regarding their build workflow.

There are a couple of files that need to be converted before they can run in the browser and to see CLI does all of that and it also heavily optimizes our code so that we ship highly optimized code word in to the browser.

We won't write any node js code here but node JS will be used behind the scenes by CLI to bundle and optimize our project and will use NPM the node package manager to manage two different dependencies and angular project test dependencies are things like the angular framework itself but also some other libraries.

5. After npm install -g @angular/cli@latest, This should now download the angular CLI from this node package manager repository and install it

on your machine.

Creating a new Project:

ng new my-first-app

cd my-first-app

ng serve-o

6. One interesting file is the package.json file.Here you can see all the dependencies of your project like Angular 6 and these are third-party packages your project needs to run correctly.

7. Angular is, of course, not a tool to allow us to write static HTML files.We wouldn't need a framework for that.

It allows us to mix static HTML code and dynamic things we want to output in that code.

8. Ngmodel [(ngModel)] is directive and what it does is it basically tells Angular to listen to anything you enter here and store it in this name property. we have to add import in app.module -> import { FormsModule } from '@angular/forms';

imports:[

FormsModule

]

9. Installing bootstrap locally :-> npm install --save bootstrap@3

but to be able to use it, we also need to make Angular aware of this styling package we want to use and

we do that in one of the config files (the most important one actually) the angular.json file.

need to add in "styles":["node\_modules/bootstrap/dist/css/bootstrap.min.css"]

10. When we hit localhost:4200 then index.html file is served, that it will contain a bunch of scripts here which get executed which then basically start the Angular app, the Angular app gets the important information from main.ts file,that it should know, the app component that it should analyze it with that information the Angular code is able to parse this here, this app-root component here understand it and insert our Angular application at this point.

11. Component: Components are a key feature in Angular,you build your whole application by composing it from a couple of components which you

create on your own.Now we do start with this app component, the root component you should say which holds our entire application basically in the end.

Below the AppComponent there is a appmodule.ts in which bootstrap:[AppComponenet] say these is the root componenet you should bootstrap the whole application with that component being the root component.So all other components we create will not be added to the index.html file, there selectors will not be added here,there selectors will be added to the app.component.html file because this is now the root component of our app where we add the other parts.

12. Decorator : Decorators are a TypeScript feature which allow you to enhance your classes for example, enhance elements you use in your code.

@Component({

selector: 'app-root',

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

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

})

13. what is a module then?

Well it's as I said bundle of functionalities of our app and it basically gives Angular the information which features does my app have and use.

As you can see, it also is just an empty TypeScript class, like our component and as the component, we transform it into something else by adding a decorator, here it's the @NgModule decorator which is also imported from @angular/core. Now in there, we see four properties we set up on the object we passed to @NgModule, declarations, imports providers and bootstrap. We had a look at bootstrap, this was responsible for telling Angular hey which component should you be aware of at the point of time the whole application starts, so which component would you basically recognize in the index.html file and that was the app component. Now I already mentioned that we won't add any more component selectors to the index.html file, that's just not how it works, therefore we won't touch the bootstrap array. Still, we added a new component to Angular and this new component now has to be registered

here in the module so that Angular knows that this component exists because and this is important, by default Angular will not scan all your files here. So if you don't tell it that the server component exists, it doesn't know it, just creating the file is not enough. That is why we have

to register it here in the @NgModule to tell Angular hey part of this module and therefore right here of our whole app since we only have this module is the server component. We do register new components in this declarations array.

14. Creating component using CLI : ng g c 'component-name' , ng g c recipes --spec false ,ng g c recipes/recipe-list --skipTests true

15. Databinding: communication between typescript code and template.

1) String interpolation : <p>{{love}}</p>

2) Property Binding : <button type="button" class="btn btn-success" [disabled]="allow (click) = 'onCreated()'>Add server</button>

3) Event binding : <input type="text" class="form-control" (input)="onUpdateServerName($event)">

onUpdateServerName(event:Event){

this.serverName = (<HTMLInputElement>event.target).value;

}

16. Directives are Instructions in DOM. Two types of directices are there :

1) Components(directives with templates)

2) Directives without template.

1) Structural directives: changes the structure of the Dom, they add or remove the elements.

<h1 style="color: cornflowerblue;" \*ngIf="sample; else noLover">My love name is {{love}}</h1>

<ng-template #noLover>

<h1>Your single</h1>

</ng-template>

2) Attribute directices:They only change the element they placed on.

1) ngStyle

<p [ngStyle] = "{color:'red'}"> Iam single ready to mingle</p>

<p [ngStyle] = "{backgroundColor:getColor()}"> Iam single ready to mingle</p>

2) ngClass

<p [ngStyle] = "{backgroundColor:getColor()}" [ngClass]="{online:serverState==='online'}"> Iam single ready to mingle</p>

17.