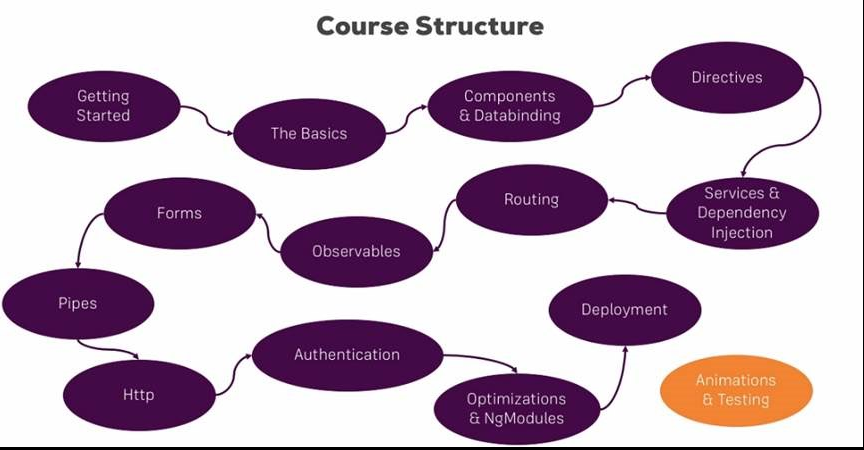
My node git credentials

Username- hemanthkaravalla



Specifically you will learn:

* + Which architecture Angular uses
  + How to use TypeScript to write Angular applications
  + All about directives and components, including the creation of custom directives/ components
  + How databinding works
  + All about routing and handling navigation
  + What Pipes are and how to use them
  + How to access the Web (e.g. RESTful servers)
  + What dependency injection is and how to use it
  + How to use Modules in Angular
  + How to optimize your (bigger) Angular Application
  + We will build a major project in this course
  + and much more!

Course started on may 22nd,should complete around june 18 may june 30th.

Start date -May 22nd

End date -June 18 th

Duration - 30 days

**1 week- 6:30 hrs. -revision on Thursday**

Basics, components and data binding, Directives, services, dependency injection, Type Script introduction, Course Roundup, Angular changes and new features.

**2 week-6:30 hrs. -revision on Thursday**

Routing, observables, forms **.**

**3rd week-6.30 hr -revision on thursday**

Pipes, Making HTTP requests, authentication and route protecting in angular app, using angular modules and optimizing apps, deploying and http client, Custom project and work flow setup.

**4th week-6.30 hr -revision on Thursday**

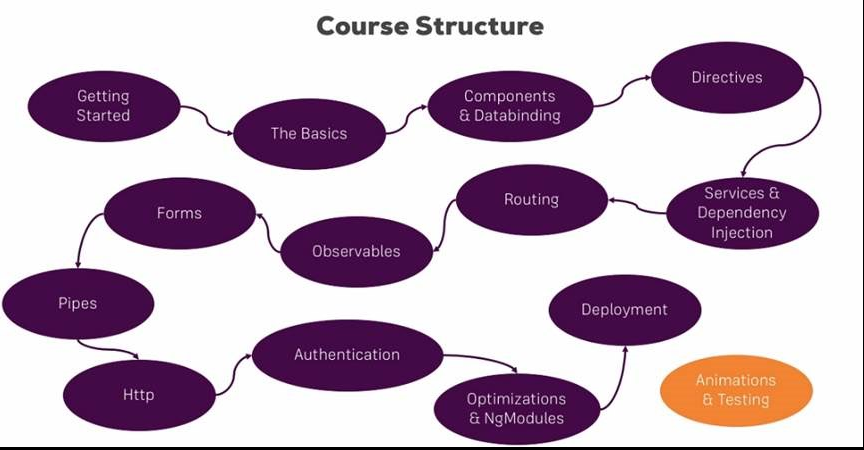
Working with NGRX in our project, Angular Universal, Angular Animations, Adding office capabilities with service workers, A basic introduction to unit testing in Angular apps

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| topic | No of lectures | days | Time required  (30 hrs) | Examples practice time | Things to revise |
|  |  | |  | | --- | | **1 week- 300min -6:30 hrs**  Basics,components and data binding,Directives,services,dependency injection, Type Script introduction, Course Roundup, Angular changes and new features. | |  |  | -Typess of data binding and syntaxes. |
| Getting started | 10 |  | 36 min |  |  |
| The basics | 33 | Wednesday | 1.53 hrs |  |  |
| Course project basics | 18 | Thursday | 1.03 hrs |  |  |
| debugging | 3 |  | 12.09min |  |  |
| Components and data binding deep dive | 20 | Friday | 1.24 hrs |  |  |
| Course project component and data binding | 6 | Saturday | 31 min |  |  |
| Directives deep dive | 11 | Saturday | 47 min |  |  |
| Project directives | 1 | Sunday | 6.25min |  |  |
| Using services and dependency injection | 12 | Sunday | 43 min |  |  |
| Course project services and dependency injection | 8 | sunday | 30.48 min |  |  |
|  |  | |  | | --- | | 2nd week - 6 :30HRS  Routing,observables,forms | |  |  |  |
| Changing pages with routing | 31 | Wednesday, Thursday | 2.18 hrs |  |  |
| Course project Routing | 15 | Friday | 45min |  |  |
| Understanding observables | 11 | Saturday | 38min |  |  |
| Course project observables | 1 | Saturday | 4 min |  |  |
| Handling forms in angular app | 34 | Sunday | 2.02 hrs |  |  |
| Course project forms | 22 | Monday and Tuesday | 1.14 hrs |  |  |
|  |  | |  | | --- | | 3 rd week - 6 :30HRS  Pipes,Making HTTP requests,authentication and route protecting in angular app,using angular modules and optimizing apps,deploying and http client | |  |  |  |
| Using pipes to transform output | 10 | Wednesday | 37 min |  |  |
| Making HTTP request | 14 | Thursday | 40 min |  |  |
| Course project Http | 5 | Thursday | 21 min |  |  |
| Authentication and route protecting in angular app | 15 | Friday | 45 min |  |  |
| Using angular modules and optimizing apps | 25 | Saturday | 1.27 hrs |  |  |
| Deploying an angular app | 4 | Sunday | 11.49 min |  |  |
| **Bonous**-The http Client | 14 | sunday | 52min |  |  |
|  |  | |  | | --- | | 4TH WEEK – 6 .30HRS | |  |  |  |
| Working with NGRX in our project | 49 | Wednesday,thrusday,Friday | 3.41 hrs |  |  |
| Angular Universal | 9 | Saturday | 27 min |  |  |
| Angular Animations | 12 | Saturday | 39.22 min |  |  |
| Adding office capabilities with service workers | 5 | Sunday | 27 min |  |  |
| A basic introduction to unit testing in Angular apps | 11 | Sunday | 45.19 |  |  |
|  |  | Topics remaining adjusted in every week |  |  |  |
| Angular changes and new features | 3 | 1 WEEK -Monday | 35 |  |  |
| Course Roundup | 2 | 1 WEEK -Monday | 2 min |  |  |
| Custom project and work flow setup | 13 | 3 RD WEEK -Monday | 51 min |  |  |
| Type Script introduction | 7 | 1 –WEEK –Tuesday and overall recall | 26 min |  |  |

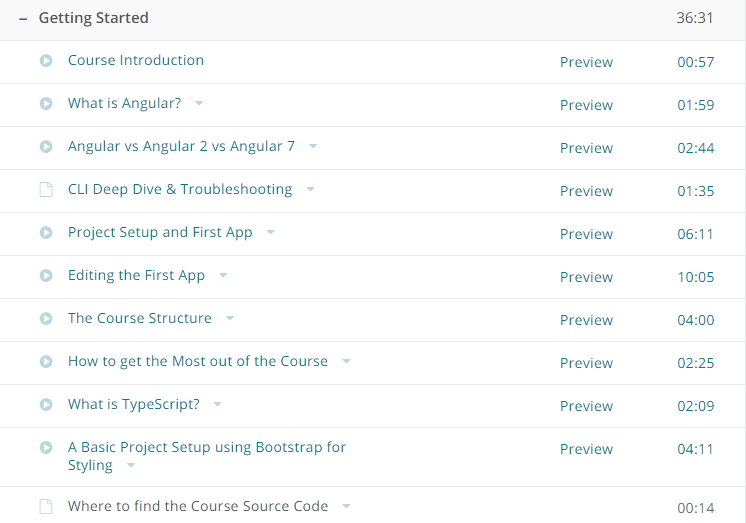
1. Introduction
   1. Angular Introduction
   2. Introduction to SPA framework
   3. Node Installation
   4. AngularJS vs Angular Comparison
2. TypeScript
   1. Introduction to Type Script
   2. TypeScript compiler
   3. TypeScript Data Types
   4. TypeScript Statements
   5. TypeScript functions and Classes
   6. Modules and Namespaces
   7. Decorators
   8. Generics
   9. Example codes
3. Angular Project setup
   1. Introduction to Angular/CLI
   2. Angular Project Structure
4. Module and Component
   1. Module
      1. Module Introduction
      2. Module Decorator
      3. Import and Export of Modules
      4. Example
   2. Component
      1. Component Introduction
      2. Component Decorator
      3. Template
      4. Typescript class
      5. Example
5. Databinding
   1. String Interpolation
   2. Property
   3. Event
   4. Two Way Data Binding
6. Component Communication
   1. Introduction to Component Hierarchy
   2. @Input Decorator
   3. @Output Decorator
   4. Event Emitter
7. Dependency Injection
   1. Introduction to DI
   2. Introduction to Services
   3. Component communication using Services
8. Directives
   1. Introduction to Directives
   2. Built-in-Directives
   3. Attribute Directives
   4. Structural Directives
9. Custom Directives
   1. @Directive decorator
   2. Custom Attribute Directives
   3. Custom Structural Directives
   4. @Input and @Output usage
   5. @HostListner
   6. Example Code
10. Lifecycle Hooks
    1. Component Lifecycle Hooks Overview
    2. Lifecycle hooks Sequence
11. Pipes
    1. Pipes introduction
    2. Built-In-Pipes
    3. Custom Pipes
    4. Angular-pipes
    5. Custom pipes Example
12. Forms
    1. Introduction to Forms
    2. NgForms
    3. Reactive Forms
       1. Forms Controls
       2. Form Group
       3. Form Array
       4. Validators

12.4 Example Code

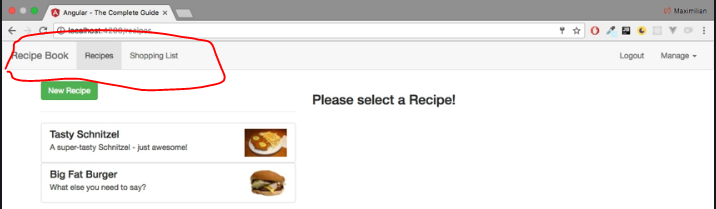
1. Pages with Routing
   1. Introduction to Routing
   2. Setting up and Loading Routing
   3. Navigating with Router Links
   4. Working with Route Parameters
   5. Router Outlet
2. Observables
   1. Introduction to Observables
   2. Reactive Programing
3. Http
   1. Introduction to Http
   2. CURD Operations
   3. GET
   4. POST
4. HttpClient
5. Build and Optimization
   1. ng build
   2. ng build vs ng serve
   3. AOT
   4. JIT
   5. Build Optimizer
6. Debugging
   1. Understanding angular messages
   2. Debugging code using Augury
7. Angular Best Practices



36min -



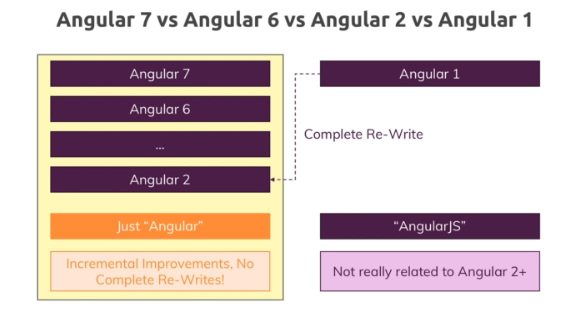
-Angular is a framework which allows you to create reactive, single page applications



you can see that we seem to visit different pages,but in the end, our page never changes.

* It's only one HTML file and a bunch of JavaScript code we got from the server and everything which you see here, every change, is rendered in the browser.
* JavaScript is much faster than having to reach out to a server for every page change and for every new,piece of data you want to display. Therefore, this approach allows you to create web applications, which look and feel almost like mobile applications; very fast!
* Well, JavaScript changes the DOM, changes whatever is displayed here (in the browser), by changing the HTML code during runtime,(so to say). That is why you never see the refresh icon on the top-left spin; because we're only changing

->the currently-loaded page. You can even see that if you inspect the source code of a page like this.That is the HTML fileand as you can see, it doesn't seem to contain the content you are seeing on this page.We only have one single HTML element which doesn't seem to be a built-in one (a native one), but that's Angular doing its job.



-complete re-write of angular1

- When it came to using browser side javascript to reach rendered a Dom to update the dom at runtime and,therefore provide highly interactive user experiences without reloading the page angular or was a complete

-angular 2 and it later versions started in 2016

-

Creating first project in Angular

npm i -g @angular/cli

ng new projectname ex-ng new my-app

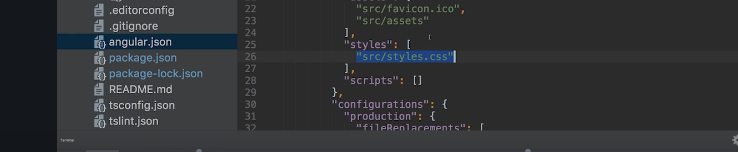
Basic project setup using Bootstrap for styling

Downloading 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. This configures the CLI project.



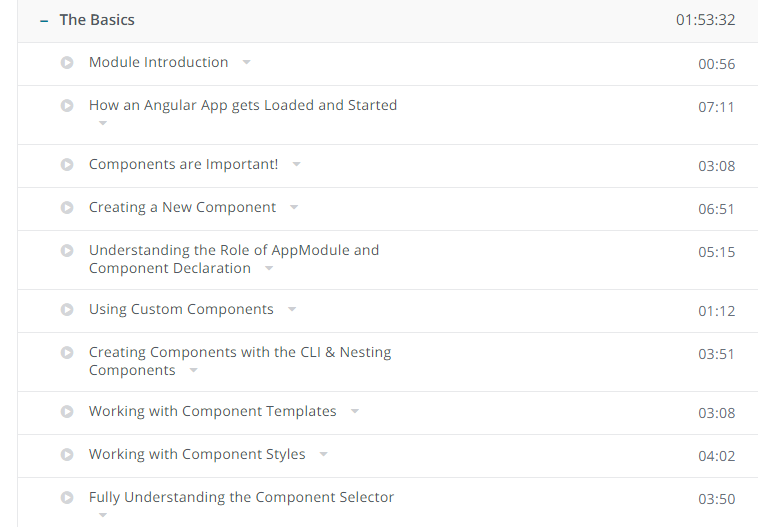
-styles.css is a file you can use to define some global styles you want to use application-wide.

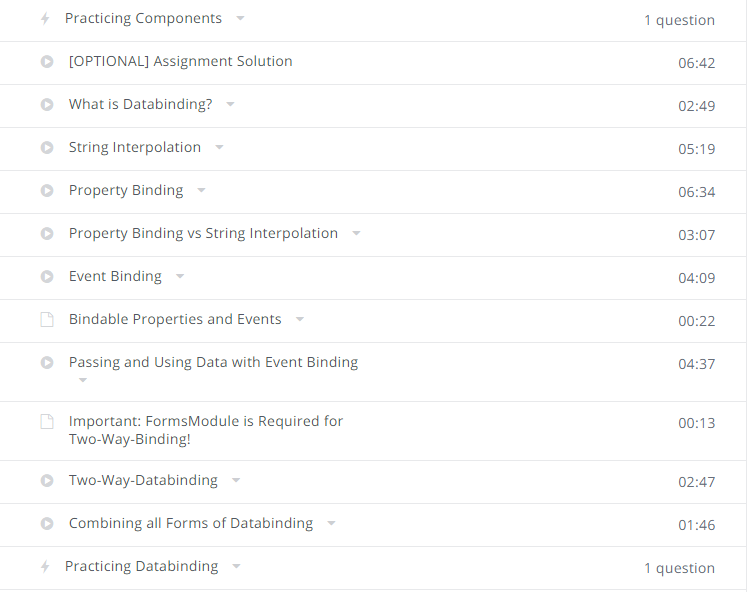
-But if you access the Elements tab in the Developer Tools, you should see that in the head section there

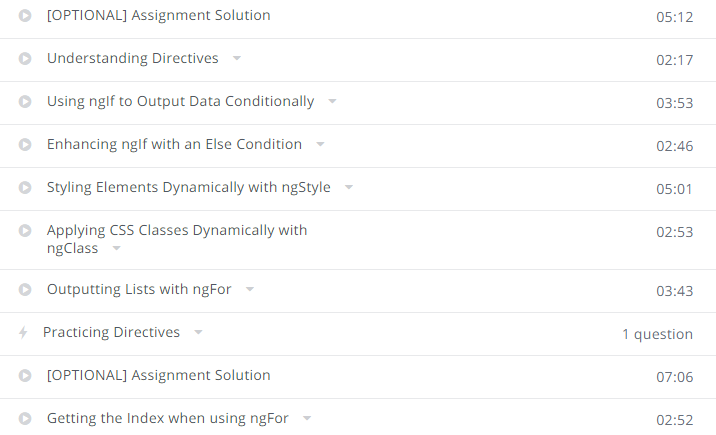
are two styles imports and the first one should be Bootstrap v3.3.7.

Now, this is the import you need there and with that, everything we do in the course will look correct.









Basics –

Module Introduction

-how angular app gets started

-By the end of this section, you will understand how you can build your basic Angular application, what

it then does and what you need to change to reach a different result

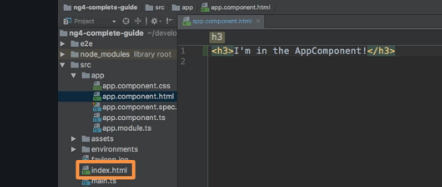
How angular app get started and loaded

-this is where our development

server hosted by the CLI or spun up by the CLI will host our Angular application.

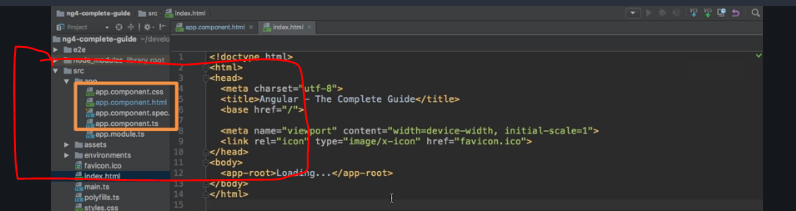
- the index.html file here is served by the server and remember that I told you that Angular

is a framework which allows you to create single page application,this is the single page which is served, the index.html file.



the CLI created one for us,

the root component of our application, the component which will tie together our whole application in



look at the app.component.ts file, the TypeScript file here.

Here you can see that we have this @component decorator, this seems to be important but more importantly

right now, you'll see that there, we have this selector property which assigns a string as a value and

this string holds

app-root. Now this clearly is the same text as in our index.html file

and this actually is the information Angular needed to be able to replace this part here in this index.html

**how is Angular triggered?**

how is it kicked off to actually run over our body here of this index.html file?

And the answer is in the final index.html file, getting served in the browser and we can verify

this by inspecting the source code here, we got a couple of script imports at the end.

These are injected by the CLI automatically,

so that is why we don't see it here in the raw index.html

file, here we don't have any script imports but whenever this **ng serve process rebuilds**

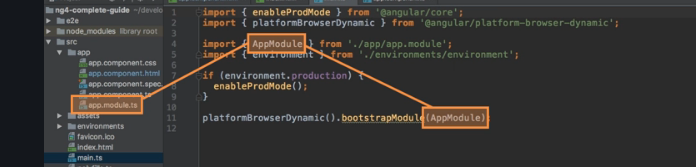
**our project, it will create bundles,Javascript script bundles and automatically add the right imports in the index.html file, a little convenience functionality for us.**

So in the final file, these script imports here are present and these script imports will contain our own code



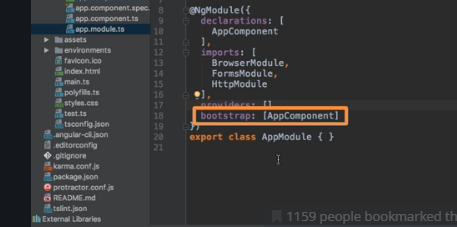
too. So these script files are **therefore executed and they're actually the first code to be executed,and that is just something you have to keep in mind, is the code we write in our main.ts file,that is why it's called main,**

this is the first code which gets executed.



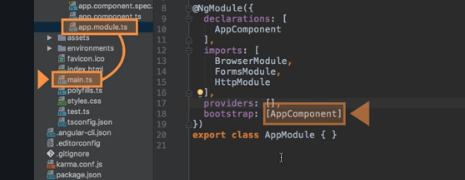
this line, this now bootstraps starts our Angular application by passing an app

module to this method and app module refers to this file here.



**here we get this bootstrap array which basically lists all the components**

**which should be known to Angular at the point of time it analyzes our index.html file and here**

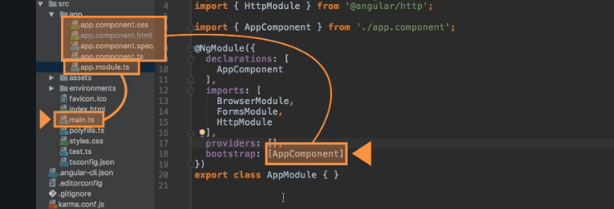


we bootstrap an Angular application and we pass this module as an argument. In this module, we tell Angular

hey there is this app component which you know when you try to start yourself and Angular

now analyzes the app component, reads the set up we pass here and therefore knows this selector, app-root

and now Angular is able to handle app-root in the index.html file



**So Angular gets started, this main.ts file gets started, there**

**we bootstrap an Angular application and we pass this module as an argument. In this module, we tell Angular**

**hey there is this app component which you know when you try to start yourself and Angular**

**now analyzes the app component, reads the set up we pass here and therefore knows this selector, app-root**

**and now Angular is able to handle app-root in the index.html file and it knows**

**all right this is the selector I know, you told me that I should know it because it was listed in this**

**bootstrap array in the app module, this component.**

**Last section overview**

In the last lecture, we had a close look at what happens at the point of time we visit localhost:4200

here in the browser.

We understood that the 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,

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

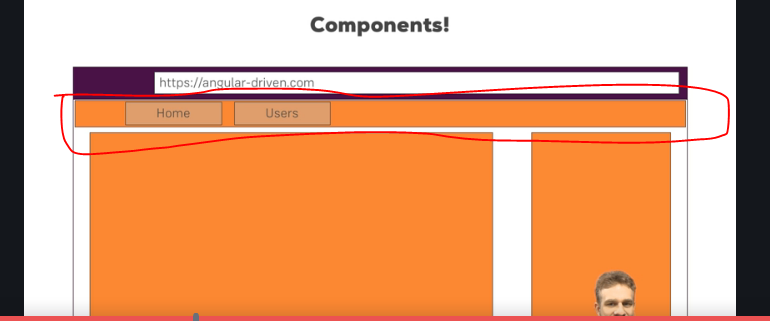
understand it and insert our Angular application at this point and that is why we don't see loading...

which would still be in the source code of this page as you can see

but instead why we see this because Angular overwrites this at runtime because that is what it does,

Components are important

So this root component, this app component will be the component where we later nest or add our other components too,



-Header may be a component

-home and users may be other components

each component has its own template, its own HTML code, maybe its own styling

and more importantly also, its own business logic and this is the great benefit.

It allows you to split up your complex application, your complex webpage into reusable parts,

you may use a component more than once and that allows you to easily replicate that business logic,

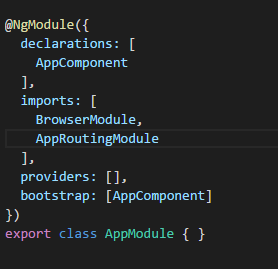
replicate that styling or in general, make a finely controlled piece in your application without having

to crunch everything into one single script file, one single HTML file, instead

it's very easy to update, very easy to exchange and again re-usable.

Creating a new component

-in appmodule.ts we will mention in bootstrap array that our root component is this.Ask angular to bootstrap the application with this.

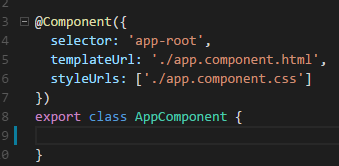


-if we want to add new component,it will be added in appcomponent.html and not appmodule.ts.

-A good practice is having a foldername = componentname.

Decorators

-ts feature that helps in enhancing elements that you use in code,ex -@component



-selector –by which name you will be using in code

-templateurl-url which html template need to use

Stylesurl- url which style file template need to use

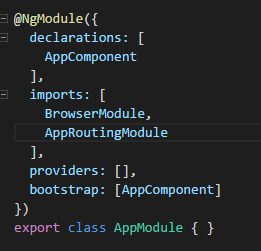
Understanding the role of appmodule and component

-appcomponents are used in angular to build web pages.

-appmodules helps to bind components to your application package.

-we will say this component is part of our modules.

@ngmodule



Imports adds other components into our module.

Using custom components

-one of the plugin which is used to write the html code much faster.

Working with components template

-template we can write code.

-We can inline your impliments:

Working with componts styles

-external & internal for template.

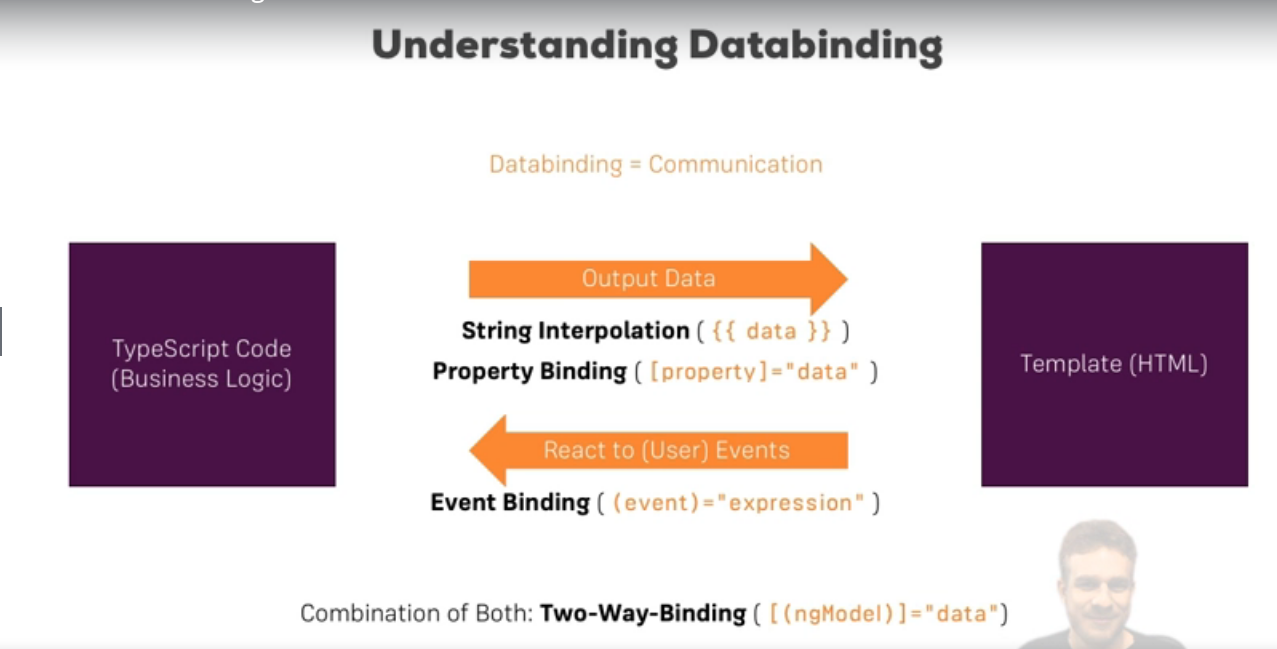
-same way we can do it for styling.

Fully understanding the component selector

-Angular selector by attribute also like example selector:’[app server]’

Now by <div app-server>

Data Binding



What is data binding?

- You could basically translate data binding with communication. Communication between your TypeScript code of your component, your business logic and the template.

We get different ways of communication now

-  for example we want to output data from our TypeScript code

in the HTML code in the template. We can use string interpolation for this,

- some property name or some expression in between or property binding.

syntax basically uses these strange square brackets around HTML attributes

Sometimes though, the other direction is interesting too,

- So if we click a button, we maybe want to trigger something in our TypeScript code,

so now we need the other direction and we can get this other direction,

we can react to user events with event binding. So we can bind to for example a click event to execute

- and we also have one additional form of data binding where we combine both directions, two-way data binding,

where we are able to react events and output something at the same time

- STRING Interprolation

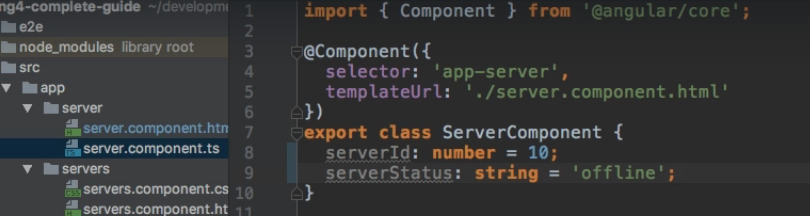
-server.compoent.js



Scenario where he want to display server with id (dynamic content) and status is (active)?

Server component.ts

-we will write the type script code.



-**Use case 2-**

you could also for example here for server simply hardcode a string in there.

any expression which canbe resolved to a string in the end,that's the only condition for a string interpolation syntax here.

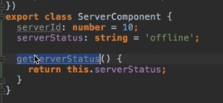
So whatever you have between the curly braces, in the end it somehow has to return a string,

so you could call a method here which returns a string in the end. The only other restriction is you

can't write multi-line expressions here,



So yes, you have to get a string in the end or something which can be converted to a string to reallybe correct here.



Calling a method

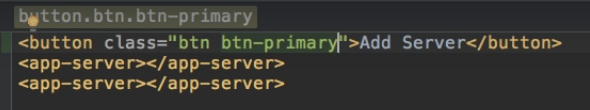


25.Property Binding

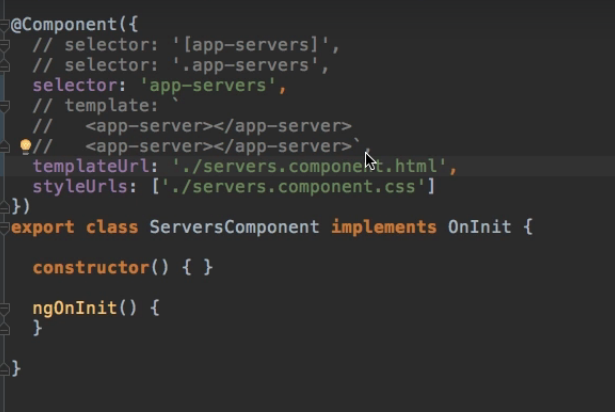
Senario- adding a server when button is clicked?

Server-component.html file

-I have to add a button



In server-cpmponent.ts file



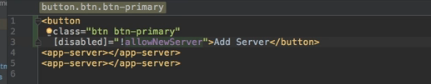
So I will add a new property in my TypeScript code here which I'll name allowNewServer and set it

to false.



Here we're binding to this disabled property, this native element property,

we're binding this to our own TypeScript property here and the convenient thing is and this is what Angular isall about, that this will update dynamically.



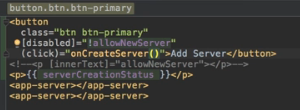
Property Binding vs String Binding

When to use Propety Binding and String Binding

Well basically if you want to output something in your template, print some text to it, use string interpolation.

if you want to change some property, be that of a HTML element or as you will later learn, ofa directive or a component,typically use property binding.

Event Binding





Two way Data Binding

we use it therefore by combining the syntaxes, square brackets and within these, parentheses.

Now here, we have to use a special directive and we will learn more about directives in a second, @NgModel.

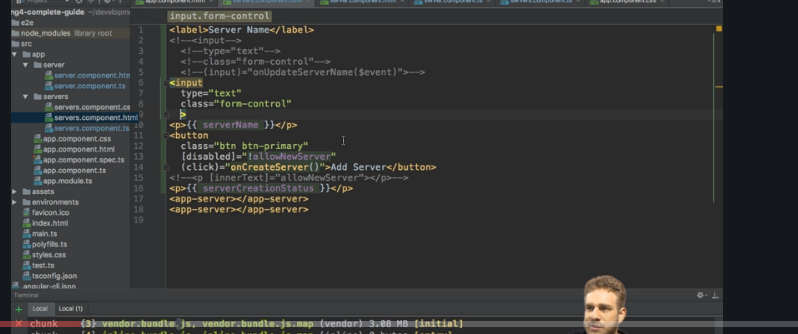
Now we can set this equal to some property defined in our TypeScript code,

so of to the server name for example. This set up will do the following,

it will trigger on the input event and update the value of server name in our component automatically.

On the other hand, since it is two-way binding, it will also update the value of the input element

if we change server name somewhere else and I can demonstrate this by going back to the TypeScript code





This set up will do the following,

it will trigger on the input event and update the value of server name in our component automatically.

On the other hand, since it is two-way binding, it will also update the value of the input element

if we change server name somewhere else and I can demonstrate this by going back to the TypeScript code

