# Angular

## Introduction

### 

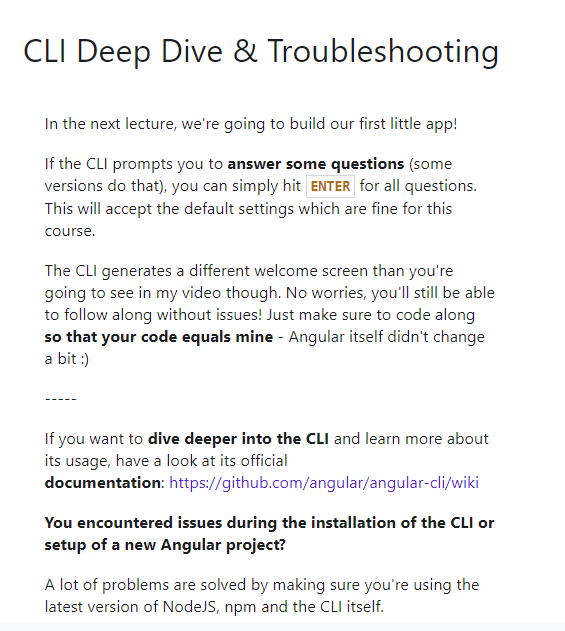
### What is Angular



### no code

### no code

### CLI Deep Dive

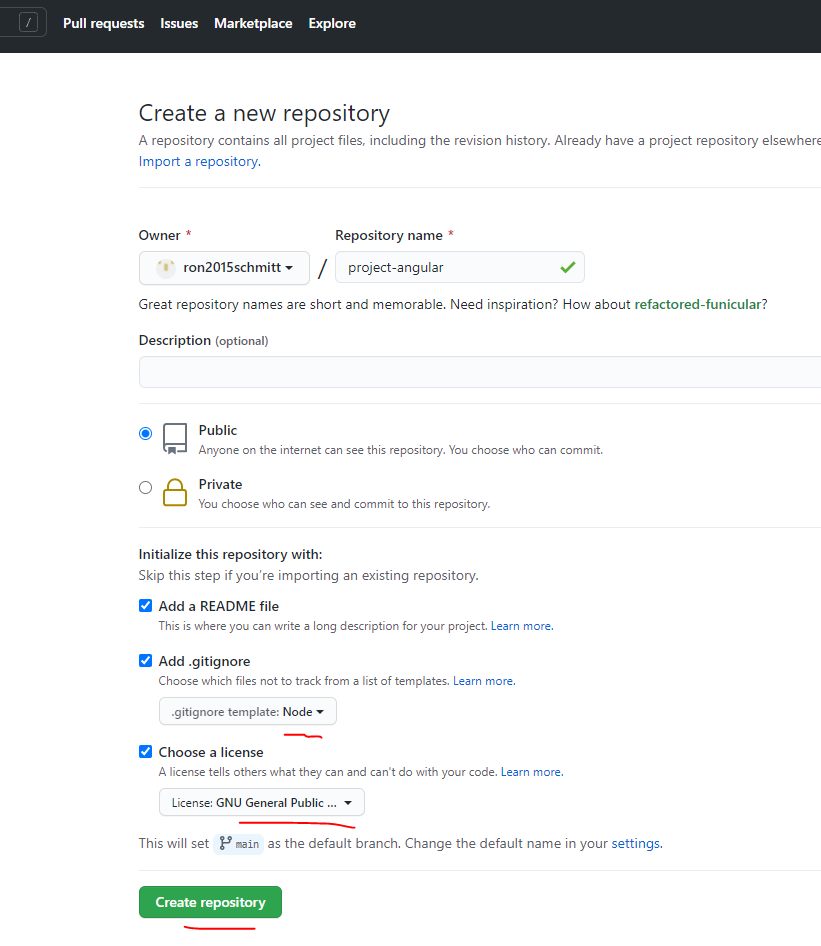


[**https://github.com/angular/angular-cli**](https://github.com/angular/angular-cli)

### Project set up and first app

Need to isntall Node js

Create repo on github



git clone <https://github.com/ron2015schmitt/project-angular.git>

Then install angular using npm:



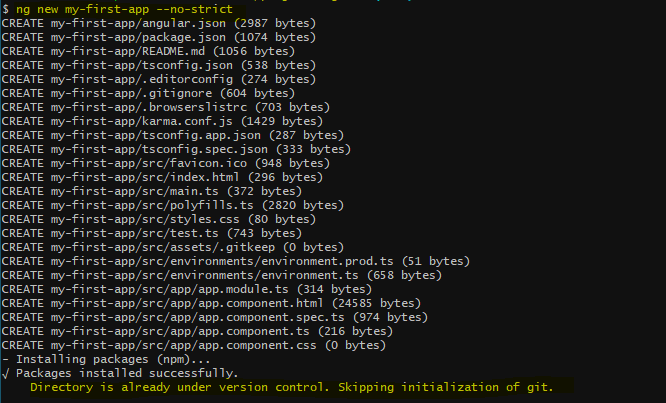
cd project-angular/

**global** install of angular

npm install -g @angular/cli@latest

create an angular project (ng is the CLI for Angular). This takes a few minutes

ng new my-first-app --no-strict

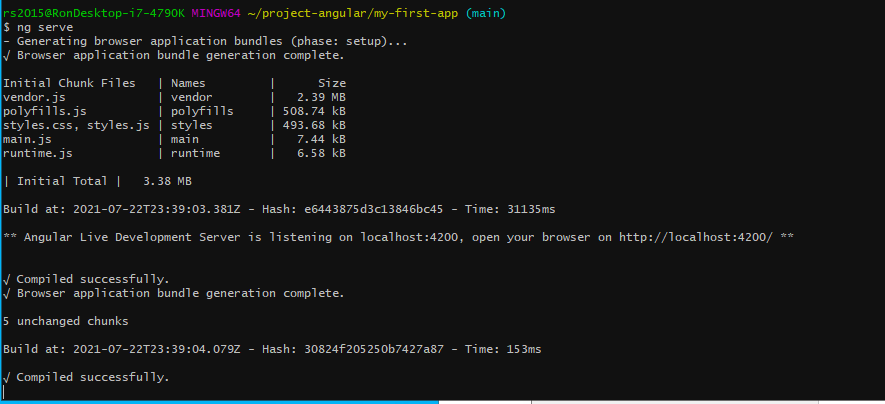


Now run the angular web server on the project

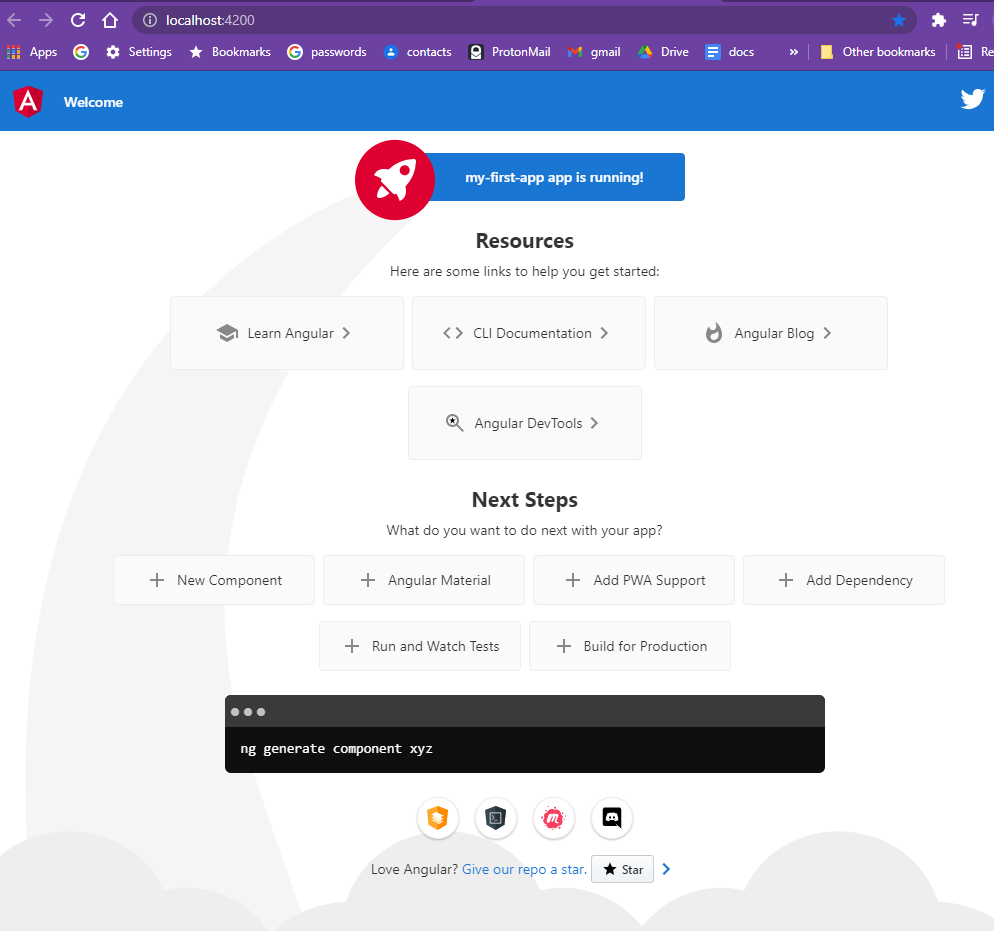
cd my-first-app

ng serve

eventually, after about a minute, it should finish compiling:



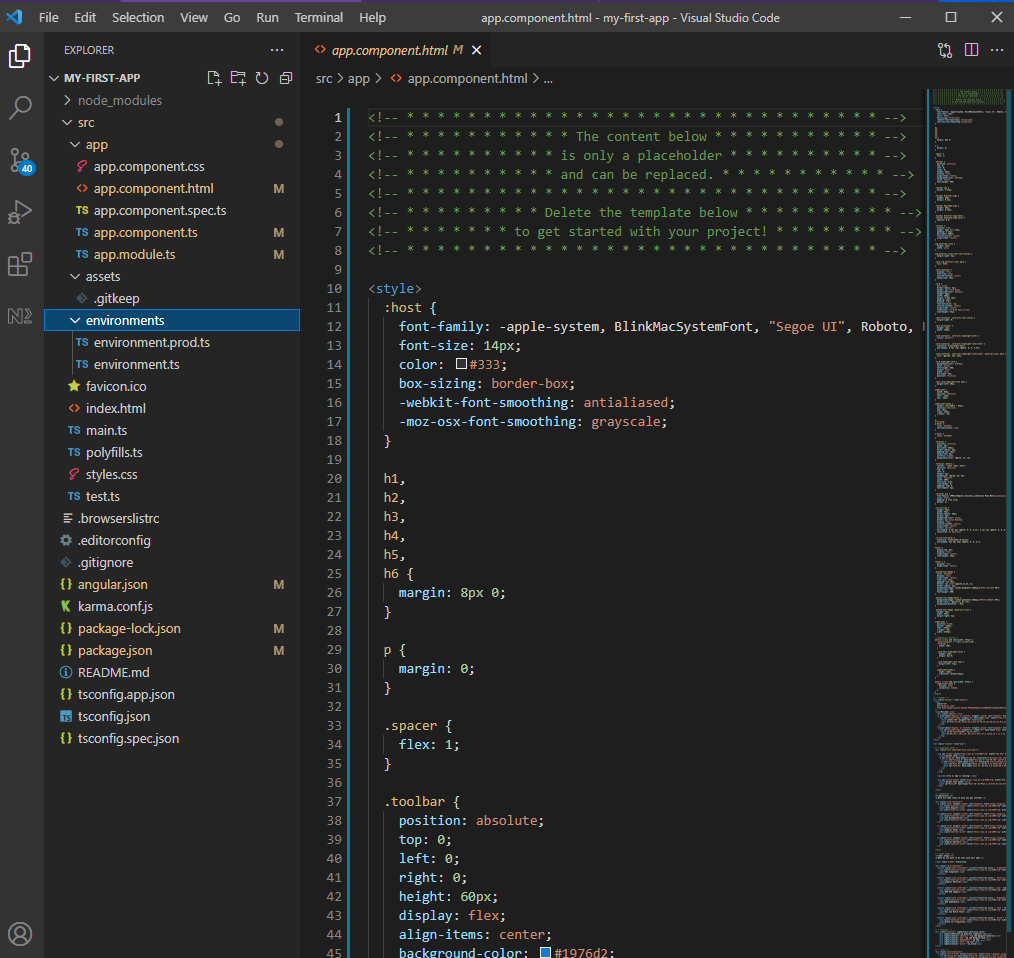
Type <http://localhost:4200/> in chrome and the following web page should appear (Angular is acting as the server)



### Editing the First App

Install VSCode (or other IDE)

Open folder in VSCode. You will see a whole slew of files that angular created



Copy my-first-app folder to my-first-app-chap-7 and open with VSCode

Open a terminal window and build the node\_modules folder

|  |
| --- |
| npm install |

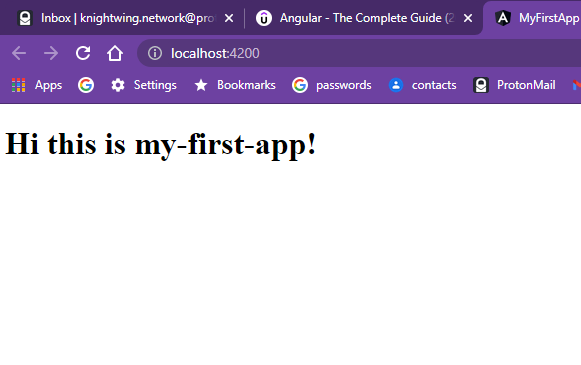
Edit the html file by deleting everything and writing below

|  |
| --- |
| <div >    <h1>Hi this is {{ title }}!</h1>  </div> |
|  |

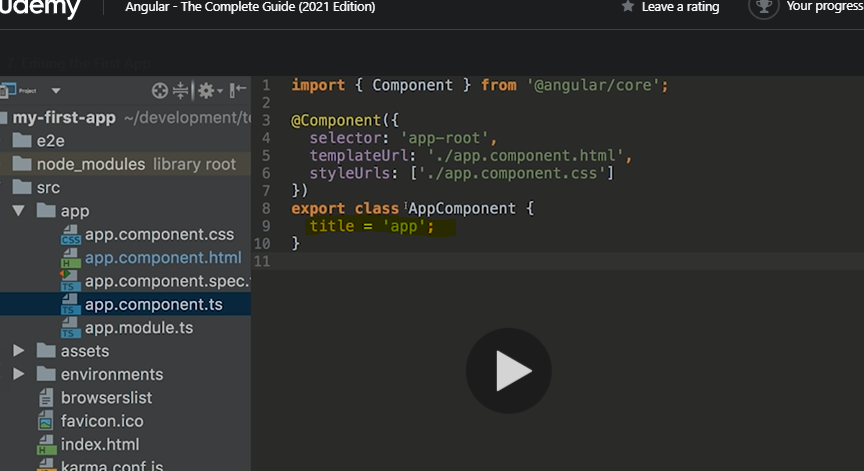
Now compile and runthe web server

|  |
| --- |
| ng serve |

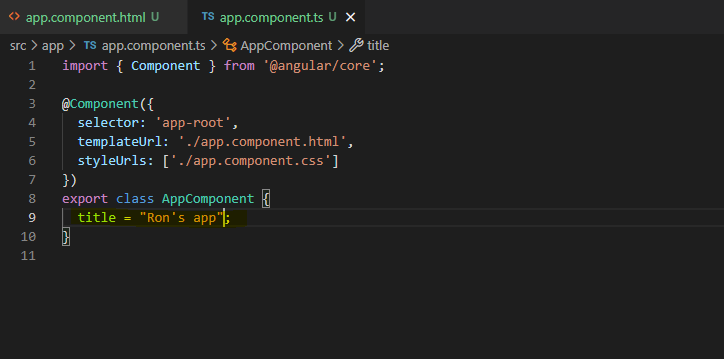
Type <http://localhost:4200/> in chrome



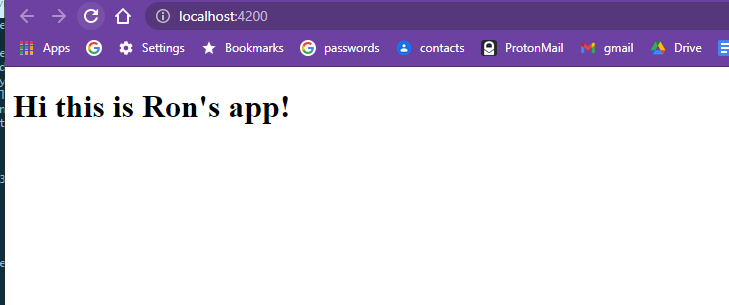
Variable title used in the html file above is set as shown below



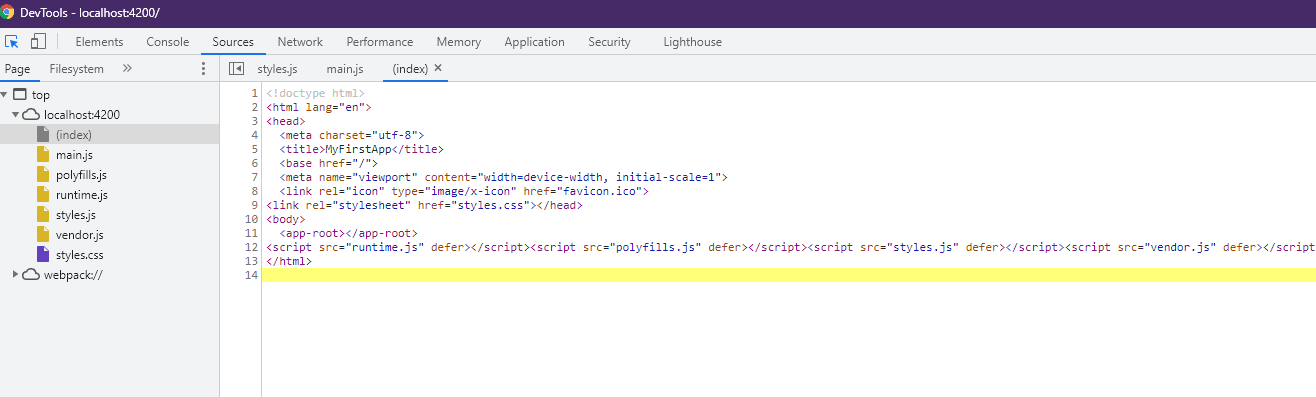
Change the title variable as shown below

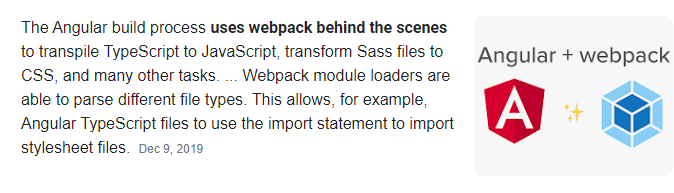


And the web page will automatically update

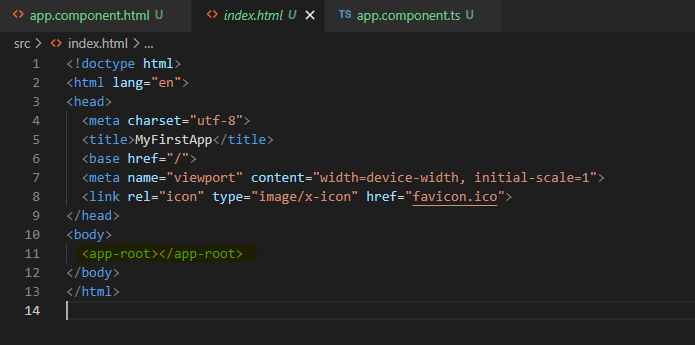


Hit F12 to see source code in browser. The index.html file is the starting point of the app. Notice that the index.html is just a bunch of scripts: angular creates code dynamically. Also notice webpack listed on the left. Angular uses wbepack!



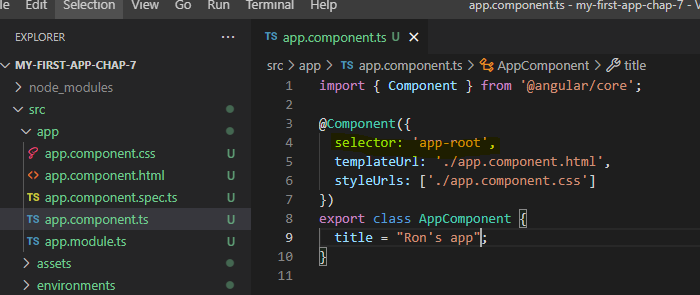


Also notice the use of custom html tag app-root

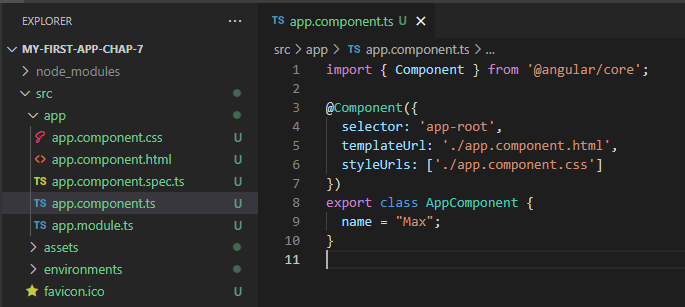




This is defiend inside the Angular component definition for app



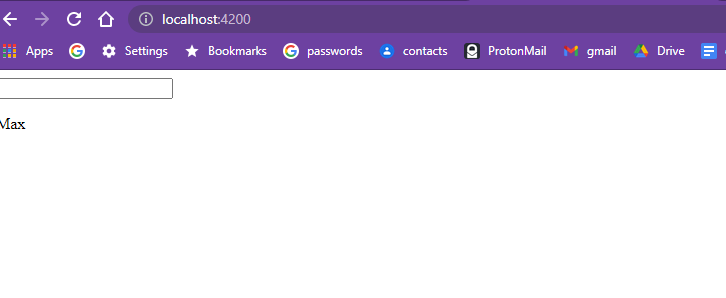
Now change the variable title to name and give it a new value shown below in the ts file



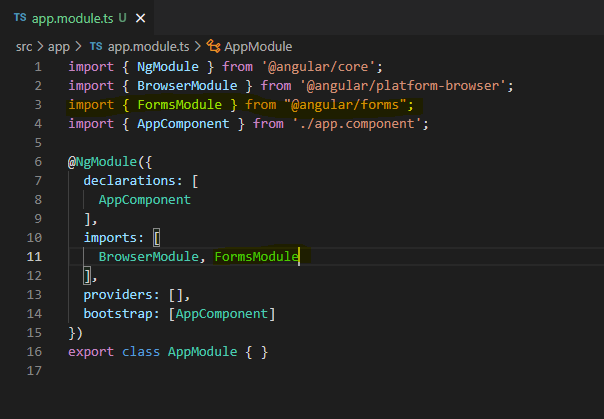
Now change the html file:

|  |
| --- |
| <input type="text">  <p>{{ name }}</p> |
|  |

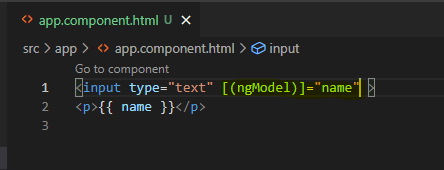
Result is



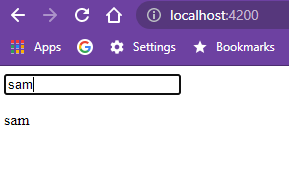
Now add FormsMoldule to src\app\app.module.ts



Now add [(ngModel)]=”name” to input which bind the input value to the variable name



Reuslt is as follows and it automatically updates as you type!!

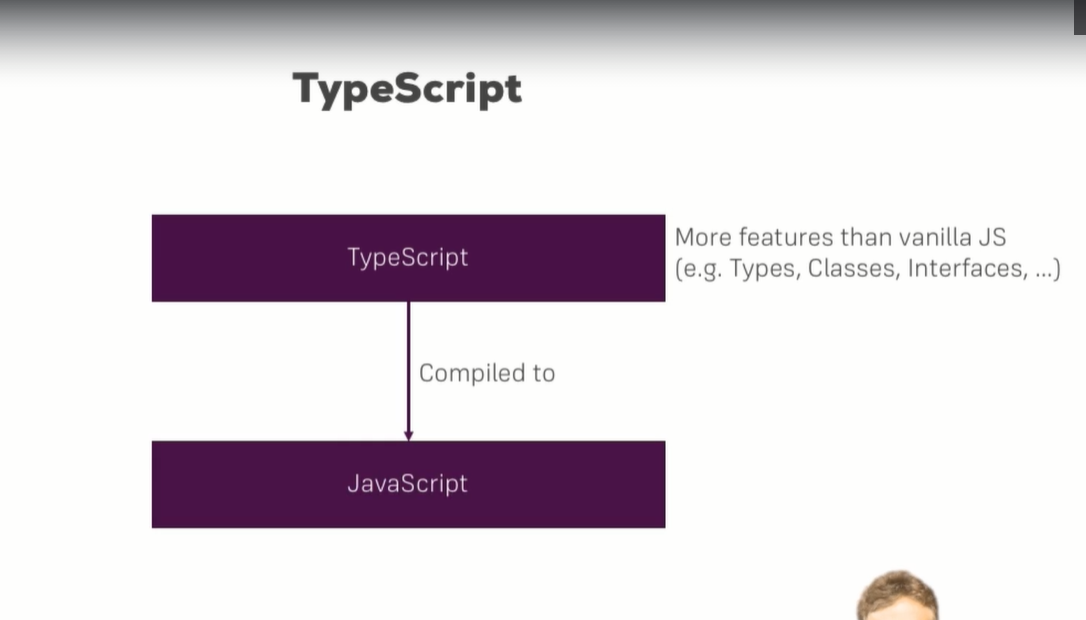


### no code

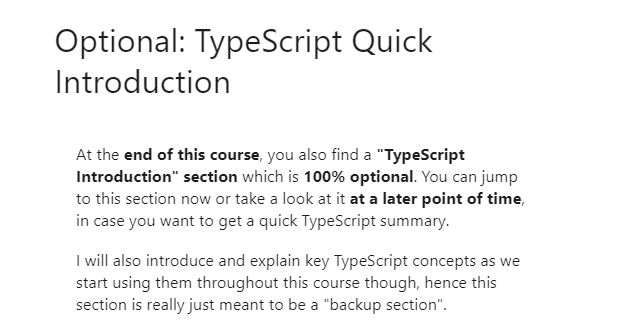
### no code

### What is TypeScript?

A super script of JavaScript and is compiled to JavaScript



### Optional TypeScript Intro as appendix



### A Basic Project Using Bootstrap for Styling

Copy project and open in vscode

|  |
| --- |
| npm install |

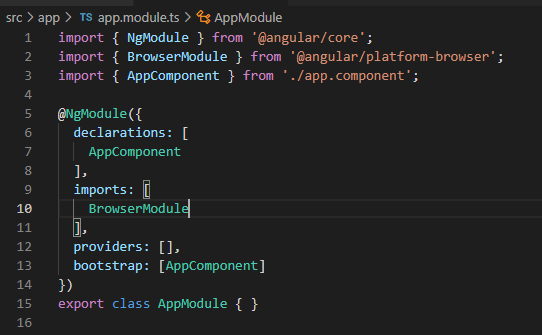
Now install bootstrap 3

npm install --save bootstrap@3

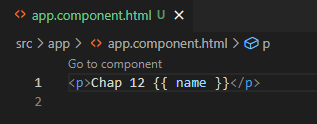
add bootstrap to the project json

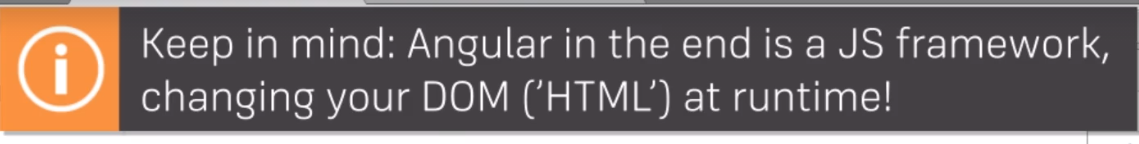
|  |
| --- |
| my-first-app\angular.json |
| {    "$schema": "./node\_modules/@angular/cli/lib/config/schema.json",    "version": 1,    "newProjectRoot": "projects",    "projects": {      "my-first-app": {        "projectType": "application",        "schematics": {},        "root": "",        "sourceRoot": "src",        "prefix": "app",        "architect": {          "build": {            "builder": "@angular-devkit/build-angular:browser",            "options": {              "outputPath": "dist/my-first-app",              "index": "src/index.html",              "main": "src/main.ts",              "polyfills": "src/polyfills.ts",              "tsConfig": "tsconfig.app.json",              "assets": [                "src/favicon.ico",                "src/assets"              ],              "styles": [                "node\_modules/bootstrap/dist/css/bootstrap.min.css",                "src/styles.css"              ],              "scripts": []            },            "configurations": {              "production": {                "budgets": [                  {                    "type": "initial",                    "maximumWarning": "2mb",                    "maximumError": "5mb"                  },                  {                    "type": "anyComponentStyle",                    "maximumWarning": "6kb",                    "maximumError": "10kb"                  }                ],                "fileReplacements": [                  {                    "replace": "src/environments/environment.ts",                    "with": "src/environments/environment.prod.ts"                  }                ],                "outputHashing": "all"              },              "development": {                "buildOptimizer": false,                "optimization": false,                "vendorChunk": true,                "extractLicenses": false,                "sourceMap": true,                "namedChunks": true              }            },            "defaultConfiguration": "production"          },          "serve": {            "builder": "@angular-devkit/build-angular:dev-server",            "configurations": {              "production": {                "browserTarget": "my-first-app:build:production"              },              "development": {                "browserTarget": "my-first-app:build:development"              }            },            "defaultConfiguration": "development"          },          "extract-i18n": {            "builder": "@angular-devkit/build-angular:extract-i18n",            "options": {              "browserTarget": "my-first-app:build"            }          },          "test": {            "builder": "@angular-devkit/build-angular:karma",            "options": {              "main": "src/test.ts",              "polyfills": "src/polyfills.ts",              "tsConfig": "tsconfig.spec.json",              "karmaConfig": "karma.conf.js",              "assets": [                "src/favicon.ico",                "src/assets"              ],              "styles": [                "src/styles.css"              ],              "scripts": []            }          }        }      }    },    "defaultProject": "my-first-app"  } |

Remove the FormsModule from app.module.ts

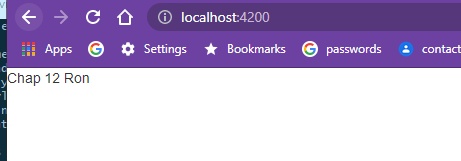


Change app.component.thtml file to

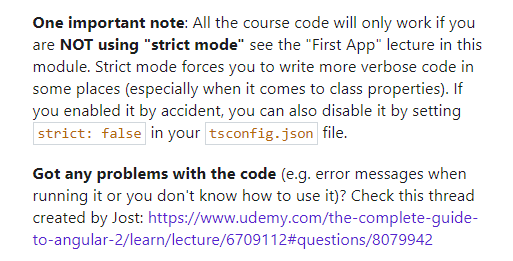




Run ng serve



### About Course Code



<https://www.udemy.com/course/the-complete-guide-to-angular-2/learn/lecture/6655698#questions/8079942>

## The Basics

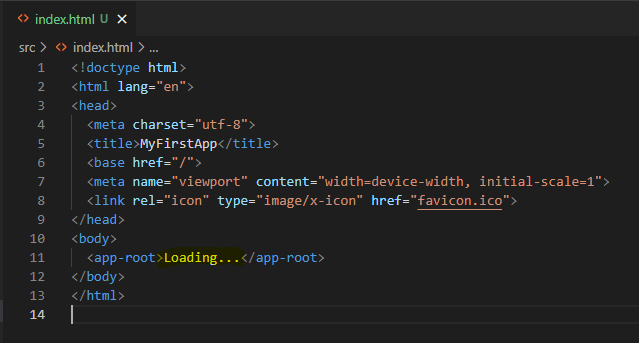
### Intro – no code

### How an Angular App gets loaded and started

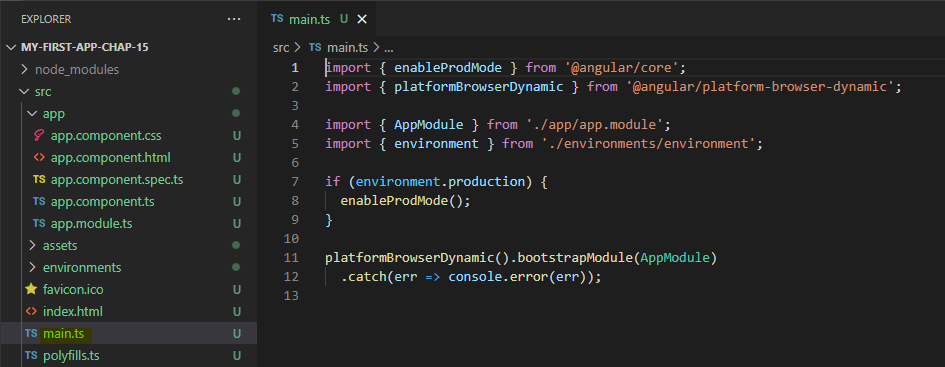
Copy project and open in vscode

|  |
| --- |
| npm install  ng serve |

Add Loading… to the index.html



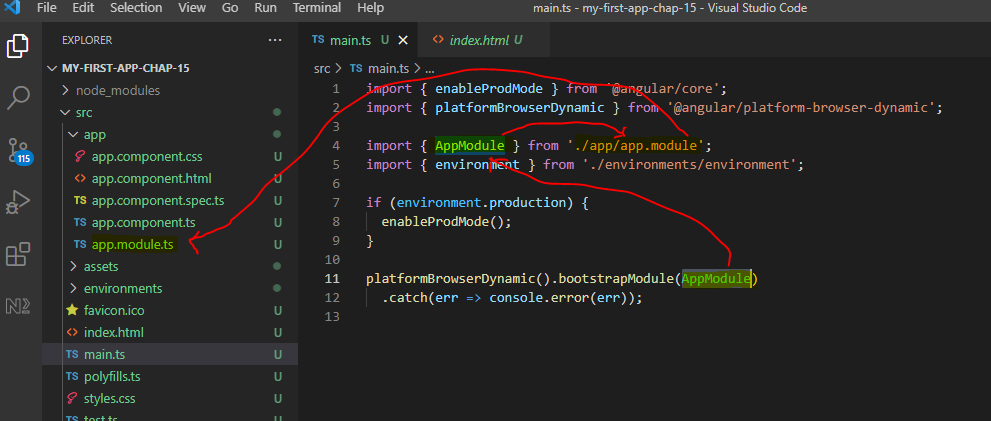
The first script executed is main.ts



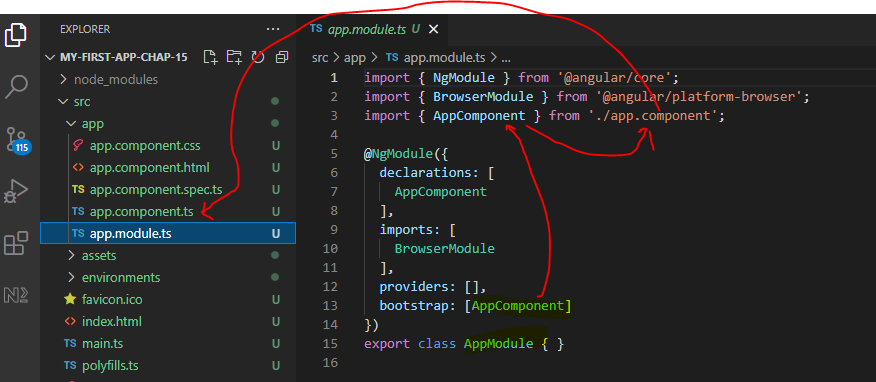
The line

platformBrowserDynamic().bootstrapModule(AppModule)

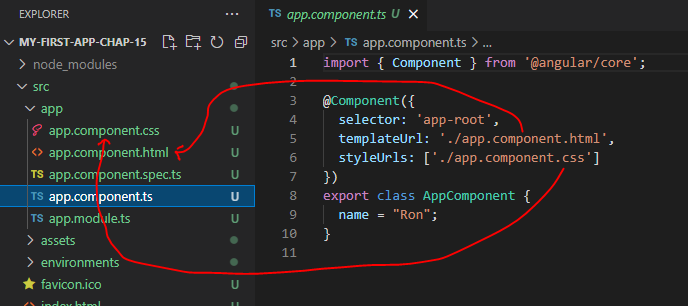
causes the module to be loaded



Inside app.module.ts the bootsrap array lists all the components used, in this case AppComponent



The app.component.ts file then calls out the html and css files of the component



### Components are important

Angular is based upon components

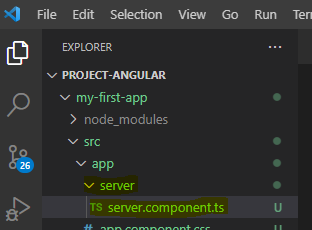
### Creating a component

Copy project and open in vscode

|  |
| --- |
| npm install  ng serve |

Time to create a component named server

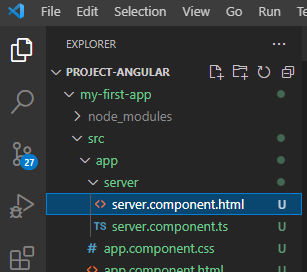
Add a folder and file as shown below, as a sub-folder of app



Enter file contents

|  |
| --- |
| src\app\server\server.component.ts |
| import { Component } from '@angular/core';  @Component({    selector: 'app-server',    templateUrl: './server.component.html',  })  export class ServerComponent {} |

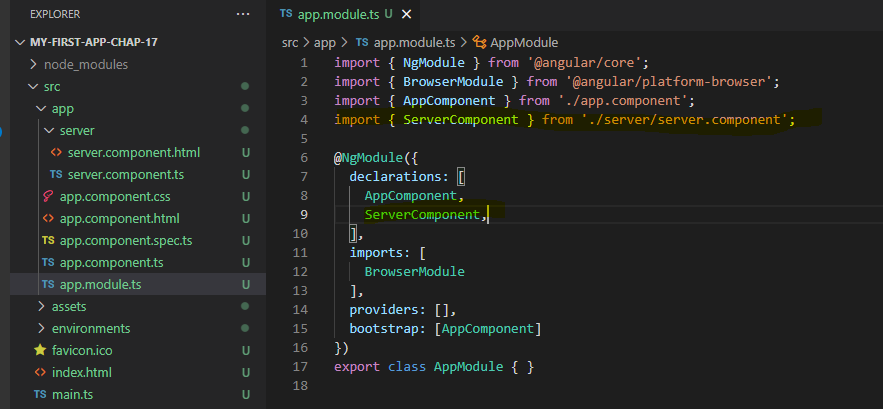
Create a blank file as shown below



### Understanding the role of AppModule and Component Declaration

Continue with chap17 version of code

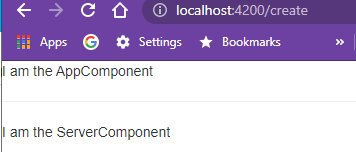
Now we need to declare our server component in the app.module.ts



### Using Custom Components

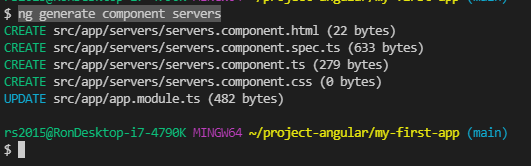
|  |
| --- |
| **server.component.html** |
|  |

|  |
| --- |
| **app.component.html** |
|  |

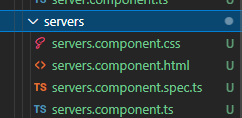


### Creating Components with the CLI & Nesting Components

ng generate component servers



The following files are added



The following lines are added to app.module.ts

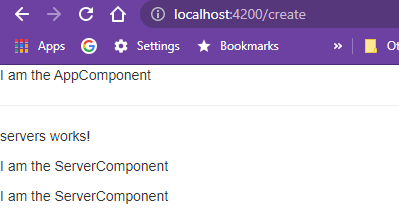


Make the following edit

|  |
| --- |
| **app.component.html** |
|  |

Make the followign edit

|  |
| --- |
| **servers.component.html** |
|  |



### Working with Component Templates

You can also define the component templete inline using template instead of templateUrl

|  |
| --- |
| **servers.component.ts** |
|  |

|  |
| --- |
| **http://localhost:4200/** |
|  |

### Working with Component Styles

Define styles in the css files.

|  |
| --- |
| **app.component.css** |
|  |

|  |
| --- |
| **app.component.html** |
|  |

|  |
| --- |
| **http://localhost:4200/** |
|  |

or you can also define the component stle inline using styles instead of styleUrls

|  |
| --- |
| **app.component.ts** |
|  |

|  |
| --- |
| **http://localhost:4200/** |
|  |

### Fully Understanding the Component Selector

using syntax

@Component({

  selector: 'app-servers',

we call out a component directly as

<app-servers>

but we can also put the name in brackets

@Component({

  selector: '[app-servers]',

we then call out using app-servers as an attribute inside another tag, such as

<h3>I am the AppComponent</h3>

<hr>

<!-- <app-servers></app-servers> -->

<div app-servers></div>

<span app-servers></span>

|  |
| --- |
| **servers.component.ts** |
|  |

thirdly we can use dot syntax

@Component({

  selector: '.app-servers',

which defines a class and is called out by

<div class="app-servers"></div>

**Assignment 1: Practicing Components**

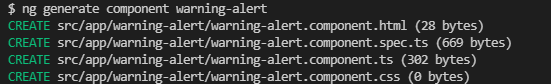
Time to practice what you learned about Components. In this assignment, you're going to create, use and style your own components and see practice how you can build up your Angular app with Components.

download basics-assignment-1-start.zip and open in VSCode

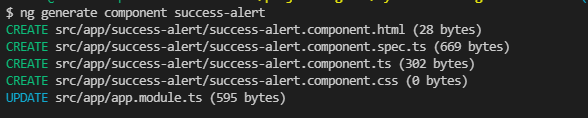
npm install

ng serve

ng generate component warning-alert



ng generate component success-alert



|  |
| --- |
| **app.component.html** |
|  |

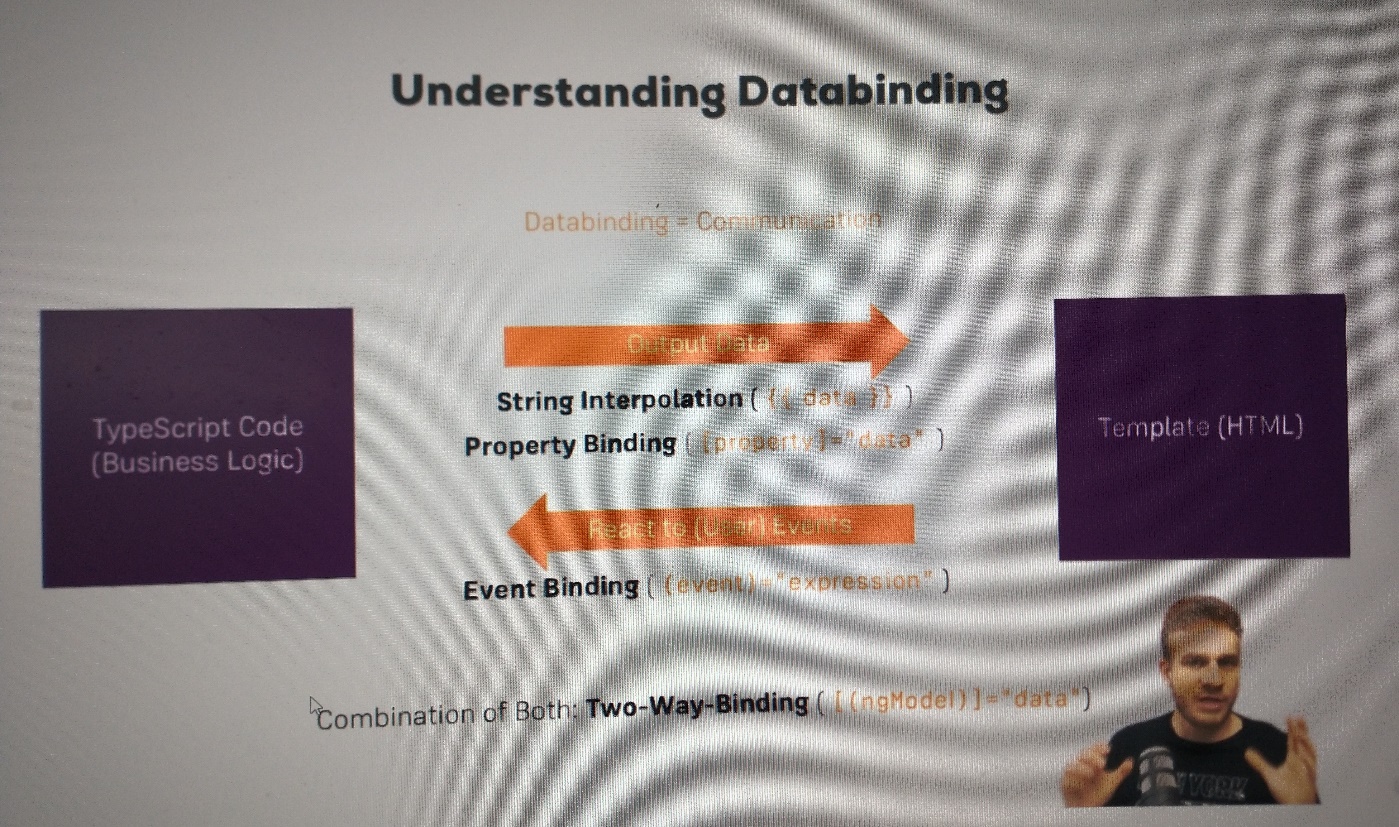
|  |
| --- |
| **warning-alert.component.css** |
| p {    color: black;    background-color: yellow;    font-weight: 600;    position: fixed;    left: 300px;  } |

|  |
| --- |
| **success-alert.component.css** |
| p {    color: black;    background-color: yellowgreen;    font-weight: 600;    position: fixed;  } |

|  |
| --- |
| **http://localhost:4200/** |
|  |

### [OPTIONAL] Assignment Solution

### What is Databinding?



### String Interpolation

You can write any TypeScript expression in {{ code\_here }} as long as it returns a string or an object that can be converted to a string, ie has a toString() method.

|  |
| --- |
| **server.component.html** |
|  |

|  |
| --- |
| **server.component.html** |
|  |

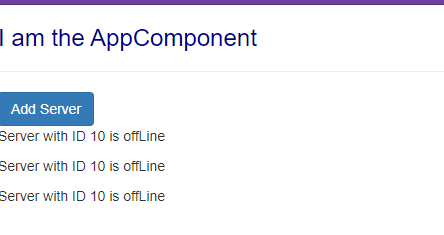
### Property Binding

we add [disabled]="code\_here" to the button. code\_here is any TypeScript code that return a bool.

|  |
| --- |
| **servers.component.html** |
|  |

|  |
| --- |
| **servers.component.ts** |
|  |

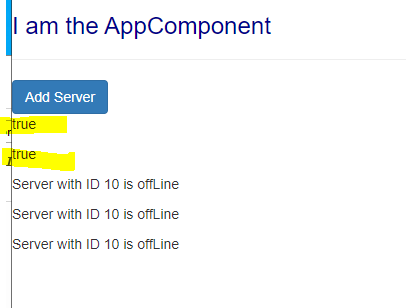
The button will be disabled at first, then enable after two seconds is up:



### Property Binding vs String Interpolation

Here we use both techniques to display allowNewServer on our site

|  |
| --- |
| **servers.component.html** |
|  |



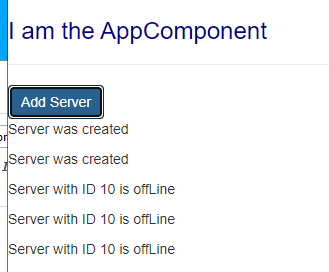
### Event Binding

the syntax for event binding is (event-name)="code-to-execute"

|  |
| --- |
| **servers.component.html** |
|  |

|  |
| --- |
| **servers.component.ts** |
|  |

state after clicking button:



### Bindable Properties and Events

How do you know to which Properties or Events of HTML Elements you may bind? You can basically bind to all Properties and Events - a good idea is to console.log()  the element you're interested in to see which properties and events it offers.

**Important**: For events, you don't bind to onclick but only to click (=> (click)).

The MDN (Mozilla Developer Network) offers nice lists of all properties and events of the element you're interested in. Googling for YOUR\_ELEMENT properties  or YOUR\_ELEMENT events  should yield nice results.

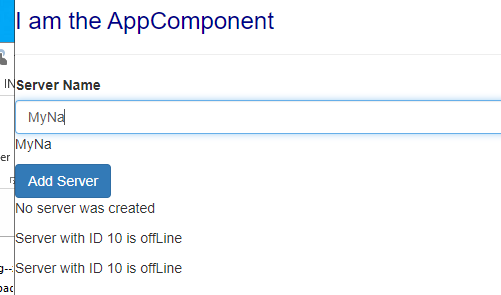
### Passing and Using Data with Event Binding

Here we add a text field with a label above it and bind it to a variable

|  |
| --- |
| **servers.component.html** |
|  |

|  |
| --- |
| **servers.component.ts** |
|  |

Note that the bound variable gets updated with each keystroke!



### Important: FormsModule is Required for Two-Way-Binding!

Important: For Two-Way-Binding (covered in the next lecture) to work, you need to enable the ngModel  directive. This is done by adding the FormsModule  to the imports[]  array in the AppModule.

You then also need to add the import from @angular/forms  in the app.module.ts file:

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

### Two-Way-Databinding

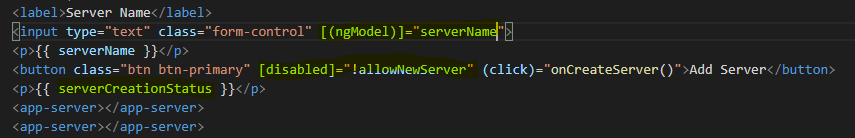
add FormsModule to the app:

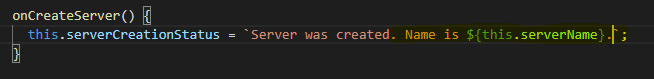
|  |
| --- |
| **app.module.ts** |
|  |

|  |
| --- |
| **servers.component.html** |
|  |

|  |
| --- |
| **servers.component.ts** |
|  |

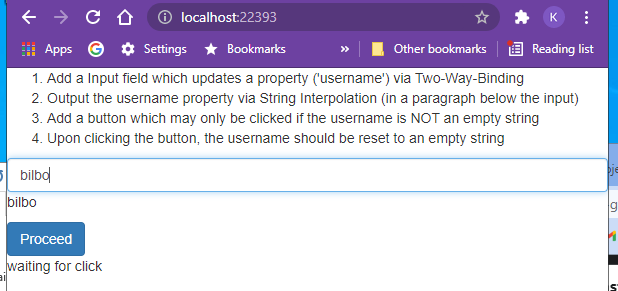
### Combining all Forms of Databinding





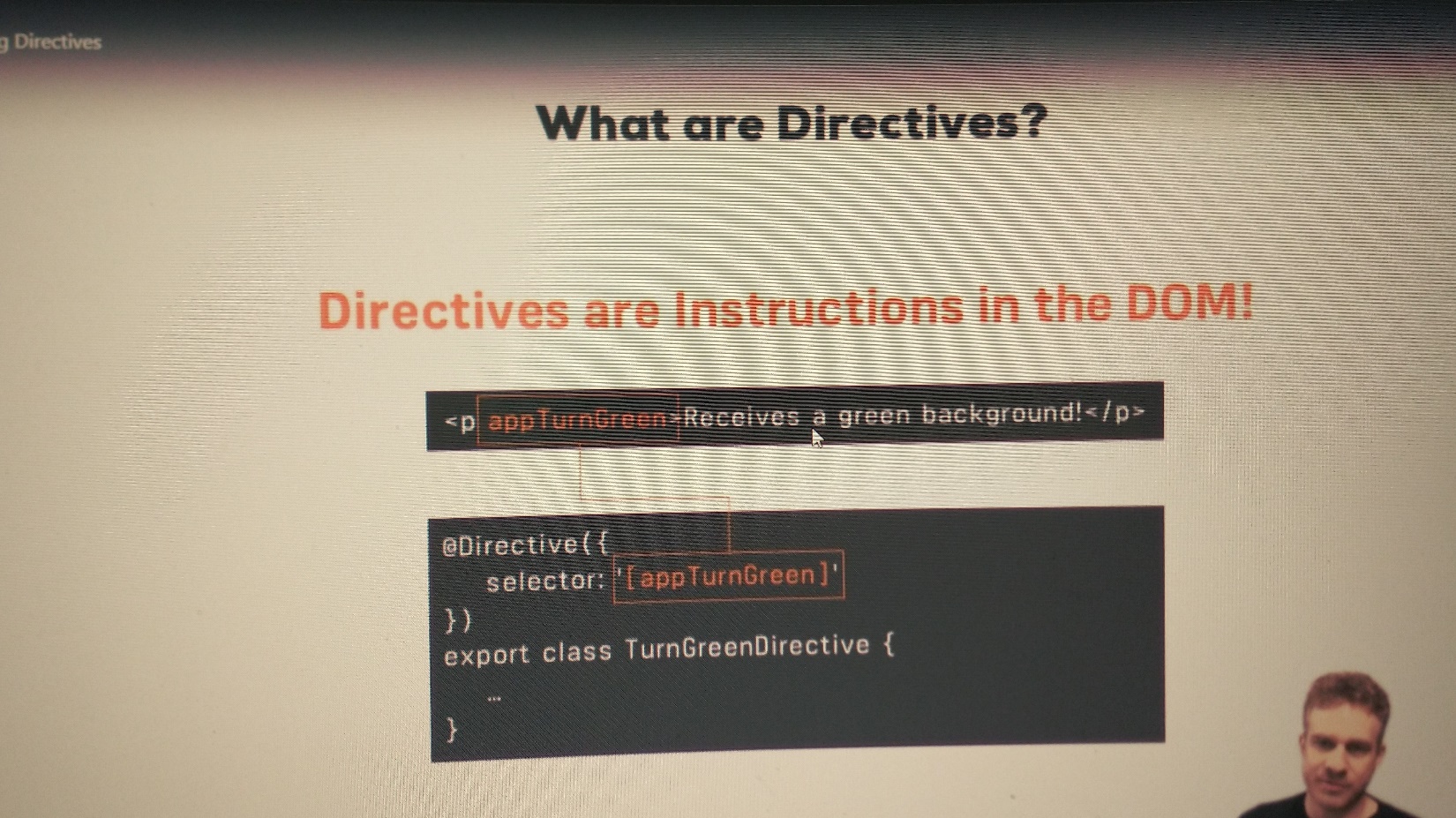
**Assignment 2: Practicing Databinding**

You learned a lot about Databinding! Time to practice it on your own. In this assignment, you're going to use the different forms of Databinding and see how you may use them in your app.



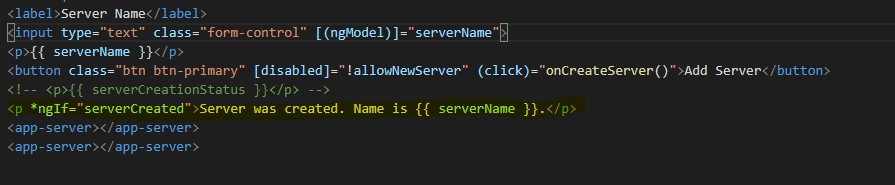
### Assignment 2 Solution

### Understanding Directives



### Using ngIf to Output Data Conditionally

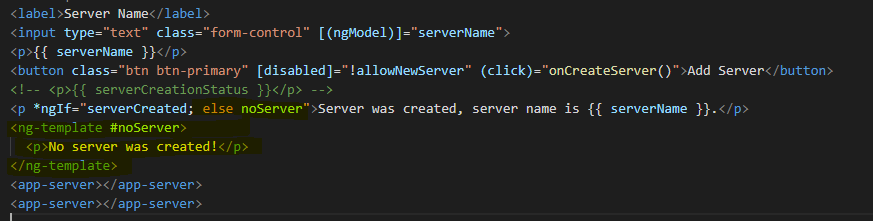
ngIf is the Angular if directive. We can use this to rewrite our code

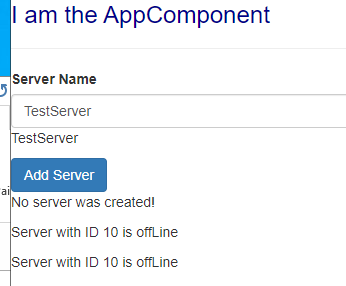




### Enhancing ngIf with an Else Condition

No we add an else clause that displays a message before the server is created





Completed thru Chap 38