**Angular**

What is Angular: -

Angular is a Javascript framework which allows you to create reactive single page applications (SPAs)

**Updating npm:**

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

**Updating the CLI**

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

npm cache verify

[sudo] npm install -g @angular/cli

npm install -g @angular/cli@latest

ng new my-first-app –no-strict

ng serve

<https://code.visualstudio.com/>

npm install --save bootstrap@3

add in angular.json file..

"styles": [

              "node\_modules/bootstrap/dist/css/bootstrap.min.css",

              "src/styles.css"

            ],

**One important note**: All the course code will only work if you are **NOT using "strict mode"**see the "First App" lecture in this module. Strict mode forces you to write more verbose code in some places (especially when it comes to class properties). If you enabled it by accident, you can also disable it by setting strict: false in your tsconfig.json file.

Application Flow: - index.html 🡪 main.ts 🡪 app.module.ts

Creating New Component: -

First create a folder under app folder 🡪 create component.ts file. Class Name userdefined. Import Component decorator. In Component mention selector, then mention html file in templateUrl.

import { Component } from "@angular/core";

@Component({

    selector: 'app-server',

    templateUrl: './server.component.html'

})

export class ServerComponent{

}

Create server.component.html file..

<h4>This is Server Component</h4>

Add ServerComponent in app.module.ts under @ngModule 🡪 declarations

import { AppComponent } from './app.component';

import { ServerComponent } from './server/server.component';

@NgModule({

  declarations: [

    AppComponent,

    ServerComponent

  ],

Then add ServerComponent in app.component.html file with mentioned selector – app-server

<h3>Hello... This is App Component..</h3>

<hr>

<p>

  <app-server></app-server>

</p>

To Create component through CLI 🡪 ng generate component servers OR ng g component servers

And for creating without test file - ng g component servers –skip-tests true

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';

String Interpolation

Property Binding

Event Binding

Two Way Binding Using ngModel

Directives

In **Angular 8+**, the @ViewChild() syntax which you'll see in the next lecture needs to be changed slightly:

Instead of:

1. @ViewChild('serverContentInput') serverContentInput: ElementRef;

use

1. @ViewChild('serverContentInput', {static: true}) serverContentInput: ElementRef;

The same change (add { static: true } as a second argument) needs to be applied to ALL usages of @ViewChild() (and also @ContentChild() which you'll learn about later) IF you plan on accessing the selected element inside of ngOnInit().

If you DON'T access the selected element in ngOnInit (but anywhere else in your component), set static: false instead!

If you're using Angular 9+, you only need to add { static: true } (if needed) but not { static: false }.

* ngOnChanges: - Called after a bound input property changes.
* ngOnInit: - Called once the component is initialized.
* ngDoCheck: - Called during every change detection run.
* ngAfterContentInit: - Called after content(ng-content) has been projected into view.
* ngAfterContentChecked: - Called every time the projected has been checked.
* ngAfterViewInit: - Called after the component’s view (and child views) has been initialized.
* ngAfterViewChecked: - Called every time the view (and child views) has been checked.
* ngOnDestroy: - Called once the component is about to be destroyed.