Angular: (Vanilla JS)

* Angular is java script frontend framework
* Angular is single page Application (ex: youtube)
* Angular is used for website structuring

* Installation:
→ npm, nodeJS Installation.
* node JS → JS we can use as programming language outside the browser is known node JS

* Angular CLI installation.

* steps
→ open nodeJS in command prompt.
→ npm install -g @angular/cli (Angular CLI install)
→ ng new cwh-todo-list (create new project)
↳ filename
→ ng serve (run the program & show in browser)

*
→ index.html is entry point of ~~app~~ any app
→ app.-routing.module.ts ⇒ Routing model, else if we want to change the url without ~~loading the page~~ reloading the page.
→ app.component.ts ⇒ type script file.
* .html => structure
  • css => style
  . ts => how it will work.
  .spec.ts => used for testing. ~~pe~~

* 'app.module.ts' => it is entry point

* Bootstrap: It is one html, css, JS framework used for responsive mobile-first sites

* Boot strap

=> npm install bootstrap

* Generate the components:

→ ng generate component xyz
↳ filename

Ex: ng generate component MyComponents /todos ~~todos~~

* How to create module:

→ create one filename.ts file

Ex: Todo.ts

.ts file is main entry point here selector whatever we maintain that we can written in index.html body of the index.html

[Angular Language Service]

* Event handling:

```
<div class="my-3">
  <h3> {{todo.title}} </h3>
  <p> {{todo.desc}} </p>
  <button class="btn btn-danger" (click)="onclick()"> Delete </button>
```

↳ Event handling

.html

.ts

```
onclick(){
  console.log("onclick has been triggered");
}
```

* Event emitter:

```
<button class="btn btn-danger" (click)="onclick(todo)">Delete</button>
```

```
.ts { onclick(todo:Todo){
        console.log("...");
      }
```

```
@output() todoDelete: EventEmitter<Todo>
        = new EventEmitter();
```

then

```
onclick(todo:Todo){
  todoDelete.emit(todo);
  console.log("...");
}
```

* Angular:

→ Angular 1, 2, 4, 5 ---- 10, 12, 13, 14, 15

→ Angular 3 was skipped

* FormsModule:

[(ngmodel)] this is forms module

ex:

```
<input type="text">
  <p>{{title}}</p>

<input type="text" [(ngmodule)]="title">
  <p>{{title}}</p>
```

↓ import FormModule

→ Create a fi component using ng new Filename

→ Main.c

→ components are key feature in angular.

* component:

→ Component is just type script class

→ create a folder or Directory like 'server' (Module name)

then create file (server.component.ts)

* component:

→ Component is just type script class

→ ng generate component Mycomponents/Server

↓ folder ↓ Filename

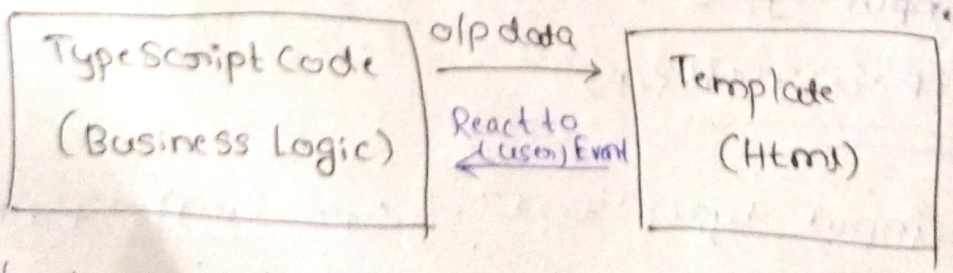
or

ng g c Mycomponents/servers

alt + up → select & move the elements to up

* Data binding:

→ Data binding means communication



{ string interpolation ({{data}})
property Binding ([property] = "data") } o/p data

{ Event building ((event) = "expression") } React to (user) Event

* Passing & using data with event binding:

example:

~~* Event binding:~~

* property binding & event binding:

```
<button class=btn btn-primary
 [disabled]="!allowAddUser" (click)="creation
                                    Server()">
```

property binding

event binding

.html

.ts

```
constructor()
{
 setTimeOut(){() => {
   this.allowAddUser = true;
 }, 2000);

creationServer()
 {
   this.ServerCreationStatus = 'server is
                                created';
 }
}
```

example2:

```
<label> Add User </label>
< input type="text" class ="form-control"
(input) ="onUpdateServerName($event)">
```

(input) → event

$event → pass by argument

* Two way binding:

→ It is combination of property & event binding

[(ngModel)] = "servername"

instead of (input) event use ngmodel

Ex:

```
<input type="text" class="from-control">
  [(ngModel)] = "serverName">;
```

* Combining all forms of data binding:

Types are

(1) String interpolation => {{ }}

(2) property binding { [ ] }

(3) Event binding { ( ) }

Ex:

[disabled] = "username === ''" (equal operator)

(click) = "username = ''" (assign operator)

* Directives: {ngIf, else, ngstyle, ngclass, ngFor}

→ Directives are instructions in the DOM!

, <h2> attributes

~~<div app-selector>~~

<p appTurnGreen> Receives a green background!

```
@Directive({
  selector: '[appTurnGreen]'
})
```

* ngIf and else condition:

<p *ngIf="serverCreated"> Server was created,
Server name is {{serverName}}

* ~~else~~ * else:

<p *ngIf="serverCreated; else noServer">

<ng-template #noServer>
 No server was created!
</ng-template>

* Styling Elements Dynamically with 'ngstyle':

~~M~~ath.random() > 0.5 ?

↓

given number blw (0 to 1) which is float point number

*

<p [ngStyle]="{backgroundColor: getColor()}">

<p [ngClass]="{ online : this.serverStatus == 'online'}"> ...

<p *ngFor="let server of servers">
{{serverName}} with id {{serverId}} is
{{getStates()}}

* Directives:

add class ↑ add styling

→ *ngIf, else, *ngFor, ngClass, ngStyle.

eg:

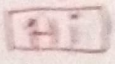
<p *ngFor = let server of servers > ...

* => ng class → used for add the class as hidden

* 4 html snippets:

1. tooltip

 ToolTip



2. download

Download

Download

3. contentedittable (it is used for editing the text)

you can edit me

you can edit as me

4. marquee (used for animation thinkss)

<marquee direction="right">
Hey this is amruta!
</marquee>

Hey this is amruta!

* Github tricks:

1. http://github.com/.....

↓type

github1s.com (then it will show in correct order)

2. vscode.dev. => type in browser then vscode is opened in browser.