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

Fast

You are changing the only one page. That’s why you will never see refresh icon circulating.

Which single page is always serve? Ans: index.html

Versions

Angular 1=Angular js

Angular 2=completely rewritten

Angular 3 was skipped for internal reasons

Angular 4 ,5,6,7,8 was released(every 6 months)

Angular is full form of ng

ng v command is used to check angular version

Setup

Nodejs= server side language

We need npm (node package manager ) to manage different dependencies in angular

First app

ng new (name of app) : to create application

ng server: to run the server (Server runs on localhost: 4200 as default)

Data Binding: It is used for communication between typescript file (B.L.)and html file(template)

* One way Binding

1. Typescript to html

html

Typescript

1)String Interpolation: {{ }}

* When you want to print text use interpolation
* Any expression which results into string there you can use string interpolation
* You can not write block statements here or multiline . So you can not write if else
* You can use ternary
* You can call method which returns string
* Number gets converted in string

2)Property binding: [property]=”variable”

When you want to change the value of property use property binding

2)html to typescript

typescript

html

1)event binding (event)=”function”

<button class="btn btn-primary" [disabled]="!status" (click)="onSubmit()" >Submit</button>

<input type="text" class="form-control" (input)="onUpdate($event)">

onUpdate(event:Event){

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

}

Here click and input are events and respective function will get called

Two way binding

1. Banana operator : [( )] = [(ngModel)]

<input type="text" [(ngModel)]="title">

Typescript

* More features than javascript
* Strong type checking(code gets checked when you write it)

(Loose type checking(code gets check at runtime) = you can assign string to number e.g javascript)

* More classes and interfaces
* Typescript is compiled to javascript and javascript is run on browser

How the code is executed

1. Main.ts is executed

platformBrowserDynamic().bootstrapModule(AppModule)

2)In app.module we declare the components that we are gonaa use in our application

@NgModule({

Declarations:[ Components that we have used in our appln]

Imports:[to import module],

Providers:[services],

Bootstrap:[to add base component]

})

bootstrap: [AppComponent]

3)App.component is executed

Index.html

Componet

Create one folder per component.

Ng g c (component-name)= generate component

Directive

Structural directive: add or remove elements

1. ngIf :

\* is used for structural directive

# is used for local variable or reference in html file(template)

Variable.value –in typescript file

<p \*ngIf="status; else close">its open</p>

<ng-template #close>

hey its closed

</ng-template>

2)ngFor

<p \*ngFor="let pname of pnames">{{pname}}</p>

Attribute directive:

1)ngStyle: Style elements dynamically with ngStyle

<p [ngStyle]="{backgroundColor:'red'}">hey good morning</p>

<p [ngStyle]="{backgroundColor:getColor()}">hey good morning</p>

getColor(){

return this.name==='shri' ? 'green': 'red';

}

2)ngClass: dynamically add or remove css classes

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

[ngClass]="{online: name==='shri'}">hey good morning</p>

@Component({

selector: 'app-root',

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

styles :[`

.online{

color: white;

}

`]

})

Bootstrap

Bootstrap, as you might know, is an open source toolkit for building very powerful, responsive and dynamic mobile-first user interfaces. It uses jquery (and popper sometimes) as a dependency. We can easily install bootstrap 4 on our angular project with one NPM command in our terminal.

npm install bootstrap jquery popper

|  |
| --- |
| "styles": [ |
|  | "src/styles.scss", |
|  | "node\_modules/bootstrap/dist/css/bootstrap.min.css" |
|  | ], |

Bootstrap 4 has some issues with navbar so installed bootstrap 3.

npm install bootstrap@3.3.7 --save

npm install bootstrap@3.3.7 tether jquery --save

In .angular-cli.json file add the dependencies

"styles":[

"styles.css",

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

],

"scripts":[

"./node\_modules/jquery/dist/jquery.js",

"./node\_modules/tether/dist/js/tether.js",

"./node\_modules/bootstrap/dist/js/bootstrap.js"

],

Error Debugging

1)cntr+shift+i

Check the console for error

2)For debugging

* for logic cntrl+shift+I check the source-> Main.bundle.js
* Webpack ->. -> src

3)Augury for debugging

You can see al components, dependencies, ngModule

For inter component communication following techniques are used

1)custom property binding

2)Custom event binding

Custom property binding

* For inter-component communication we use custom property binding
* By default property of component is accessible to that component only.
* Suppose we have child component and we need to access the property of child component in parent component we need to add a Decorator i.e. **@Input** in the child.component.ts file .
* Also this decorator must be imported from @angular/core .
* Import{ Component , Input} from @angular/core ;

export class childComponent implements OnInit{  
   @Input() customproperty : { name: string , content: string };  
  constructor(){}  
}

type definition

export class childComponent implements OnInit{  
   @Input(‘myproperty’) customproperty : { name: string , content: string };  
   constructor(){} }

And therefore in the app-component.html we have to change the property name from [customproperty] to [myproperty]

**export class ServerElementComponent implements OnInit {**

@Input()element:{type:string,name:string,content:string};

constructor() { }

ngOnInit() {

}

}

--serverElementComponent.ts(child component)

<div class="panel panel-default">

<div class="panel-heading">{{element.name}}</div>

<div class="panel-body">

<p>

<strong \*ngIf="element.type=='server'" style="color:red">{{element.content}}</strong>

<em \*ngIf="element.type ==='blueprint'">{{element.content}}</em>

</p>

</div>

</div>

serverElement html file

export class ServerComponent implements OnInit {

serverElements=[{type:'server',name:'server1',content:'it just a test'}];

}

appComponent.ts

<div class="container">

<app-cockpit (serverCreated)="onServerAdded($event)"></app-cockpit>

<hr>

<div class="row">

<div class="col-xs-12">

<app-server-element \*ngFor ="let serverElement of serverElements"

[element]="serverElement"></app-server-element>

</div>

</div>

</div>

AppComponent.html file

Custom event binding

Event is created in cockpit element- creating server and blueprint.

Once the event is emitted then in app component data is push in server Element array and then is displayed on browser. Event emitter should be imported from angular/core

export class CockpitComponent implements OnInit {

@Output() serverCreated=new EventEmitter<{serverName:string,serverContent:string}>();

@Output() blueprintCreated=new EventEmitter<{serverName:string,serverContent:string}>();

newServerName='';

newServerContent='';

constructor() { }

ngOnInit() {

}

onAddServer(){

this.serverCreated.emit({

serverName:this.newServerName,

serverContent:this.newServerContent

});

}

onAddBlueprint(){

this.blueprintCreated.emit({

serverName:this.newServerName,

serverContent:this.newServerContent

});

}

}

CockpitComponent.ts

--

Cockpit.html

<div class="row">

<div class="col-xs-12">

<p>Add new server or blueprint!</p>

<label>Server Name</label>

<input type="text" class="form-control" [(ngModel)]="newServerName">

<label>Server Content</label>

<input type="text" class="form-control" [(ngModel)]="newServerContent">

<br>

<button class="btn btn-primary" (click)="onAddServer()">Add Server</button>

<button class="btn btn-primary" (click)="onAddBlueprint()">Add blueprint</button>

</div>

</div>

<app-cockpit (serverCreated)="onServerAdded($event)"

(blueprintCreated)="onBlueprintAdded($event)"></app-cockpit>

appComponent.html

appComponent.ts

onServerAdded(serverData:{serverName:string,serverContent:string}){

this.serverElements.push({

type:'server',

name:serverData.serverName,

content:serverData.serverContent

});

}

onBlueprintAdded(blueprintData:{serverName:string,serverContent:string}){

this.serverElements.push({

type:'blueprint',

name:blueprintData.serverName,

content:blueprintData.serverContent

});

}

---------------------------------------------------------------------------------------------------

Deployment

1)use &check environment variable

2)polish & test code

3)ng build –prod (for optimization- making application smaller)

4)Deploy artifacts(generated file)in static host.

View Encapsulation

For each component there is different css .

The elements in that particular component will affected by its own css only.

It is achieved in angular by view encapsulation. Angular adds it own element to one component and different element to another component.

We can override this encapsulation by adding

Encapsulation:viewEncapsulation.None

Lifecycle of component

1)ngOnChanges: when we assign new value to @Input decorated property.

2)ngOnInit: Called Once when the component is initialized

3)ngDoCheck: it will execute when there is change in template(if any event occurs).