Angular.... ....Material

Eng. Ayah Alrefai

Jun/2022

what is Angular ?

Javascript Framework.

use Typescript

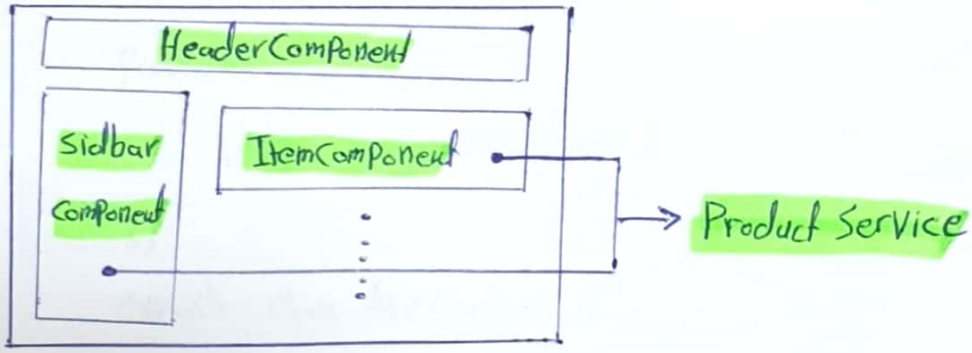
Build client-side (Browser) web App.

Single Page Application (SPAs)

Highly scalable, Highly Performance.

Supported by Google.

Core Concepts.



Run Angular App

1. install nodejs.
2. cmd → npm install -g @angular/cli
3. create Angular App.

   cmd → ng new App-name
   → cd App-name
   → ng serve

how exactly is the flow ?

in src there are "index.html" have this tag <app-root> </app-root>

"main.ts" first file executed when the app starts.

main.ts

```
...
import {AppModule} from './app/app.module';   → no extention .ts
...
platformBrowserDynamic().bootstrapModule(AppModule).catch(err => log(err));
```

app.module.ts

```
@ngModule({
    declarations: [ AppComponent ],
    imports : [BrowserModule, AppRoutingModule],
    Provider : [],
    bootstrap: [AppComponent]
    })
    export class AppModule {

    }
```

decorator "meta Data"

1. Declaration: declar all components you are using it in Project (Array)
2. Imports: other components declared by angular
3. Provider: it use to provide services.
4. Bootstrap: main component of your Application.

APP.Component

```
@Component({
    selector : 'app-root',
    templateUrl: './app.component.html',
    styleUrls: ['./app.component.css']})
export class AppComponent {
    title = 'app';
```

→ in index.html

app.component : Selector

we use app-root because this component is main component

in app.module the bootstrap is app.component

ممكن تكون القيمة مختلفة بس بشرط الـ main يلي تابع إلها يكون فيه tag بنفس الإسم " يعني كأنك بعمل div وإلها id خاص فيها "

بس لازم أضيفه على app.module

After add ProductsComponent

app.module

```
@NgModule({
    declarations: [
        AppComponent,
        ProductsComponent ]
    :
})
```

Template Syntax

Products.component.ts

```
export class ProductsComponent {
    name = 'Book';

    constructor() {
        setTimeout(() => {
            this.name = 'Tree';
        }, 3000);
    }
}
```

Products.component.html

```
<div> {{ name }} </div>
```

Add Event

 Hello world

<button (click) = " name = 'Tree' "> Change Name </button>

 {{ name }}

Properties

old ⇒ <button disabled> click me </button>

new ⇒ <button [disabled] = "isDisabled"> click me </button>

should add brackets

↳ declar it in class (.ts)

isDisabled = true;

FormsModule

it is an Angular module so we should add it on app.module.ts and add it on imports

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

the init value of input same as the value of name and when the value of name change the value of <input> will be change and when the value of <input> change the value of name will be change

Directives

change the structr of Page by adding and removing elements

Products.componeut.ts

```
products = ['Book', 'Tree'];
onAddProduct() {
    this.products.push(this.name);
}
```

Products.componeut.html

```
<input type="text" [(ngModel)] = "name" />
<button (click) = "onAddProduct()" >
    Add Product
</button>
<div> {{ name }} </div> X  delete this line

<div *ngFor = "let product of Products" >
    {{ product }}
</div>
```

*ngIf

syntax ⇒ `<div *ngIf= "true" > hi </div>`

if Value [true] show elemeut

if Value [false] remove elemeut

[بتأثر على نفس Elemeut اذ]

syntax ⇒ `<div *ngIf = " isDisabled ; else elseBlock "> </div>`

`<ng-template #elseBlock > isDisabed Fasle </ng-template>`

syntax ⇒ `<div *ngIf = " isDisabled ; then thenBlock else elseBlock "> </div>`

`<ng-template #thenBlock > isDisabled true </ng-template>`

`<ng-template #elseBlock > isDisabled Falso </ng-template>`

[بتأثر على Elements مختلفة]

*ngFor

syntax ⇒ *ngFor = "let input of inputs"

syntax ⇒ *ngFor = "let input of inputs; let i = index"

Can't looping on more than one array, if you ~~want~~ wanna looping on more than one array use [index]

Ex: <tr *ngFor="let input of inputs; let i = index">
<td> {{ input.name }} </td>
<td> {{ input.age }} </td>
<td (click)="onDeletePerson(i)"> X </td>
</tr>

row ← بعت الـ index
يلي بدي أحذفه

Custom property [From Parent to child]

Product.component.ts (new sub component)

```
export class ProductComponent implements onInit {
    @Input() ProductName: String;
    constructor() {}
    ngOnInit() {
    }
}
```

Same

Products.component.html

<app-product *ngFor="let product of products" [ProductName]="product">
</app-product>

Event Binding

ProductComponent (sub component)

```
:
@output productClicked = new EventEmitter();
:
onClick() {
    this.productClicked.emit();
}
```

أي تعديل على الـ Prop بالـ super class أو مباشرة على الـ sub class بين المكونين غير مهم — ليك يعاود EventEmitter

ProductsComponent (super component)

```
<app-product (productClicked) = "onRemoveProduct(Product)" ....
```

ProductsComponent (super component)

```
:
onRemoveProduct( ProductName : String) {
    this.product = this.product.filter( P => P !== ProductName) ;
}
```

Form

```
<form (ng Submit) = "onAddProduct(f)"  #f= "ngForm" >
        <input type="text"  ngModel  name = "productName"  required >
        <button type= "submit" > Add product </button>
</form>
```

```
onAddProduct( form) {
    form.valid → boolean
    form.value.ProductName → the value of input where name equal ProductName
}
```

Services and Dependancy

الهدف الأساسي منها إنه أعمل global prop كل الـ child يقدروا يشوفوها، تكون by Reference

مثل Use Context بالـ React

1. Create services [Product.service.ts]

```
export class ProductServices {

  Private Products = ['A Book']; → Private معناها ما حدا يعدل عليها مباشرة من برا

  ProductUpdated = new Subject(); → معناها بعمل update في حال حدا عمل إضافة أو حذف

  addProduct(name: string) {
    this.products.push(name);
    this.productUpdated.next(); → بعمل Value الـ update
  }

  deleteProduct(name: string) {
    this.products = this.products.filter(p => p !== name);
    this.productUpdated.next(); → بعمل Value الـ update
  }

  getProducts() {
    return [...Products]; → object ما بيعمل نفس الـ بيعمل نسخة عنه
  }
}
```

Providers بالـ app.model لازم أضيفه كمان

2. super Component

before Component render

Component will unmount

```
export class ProductsComponent implements onInit, onDestory {
  :
  ProductSubscription : Subscription = new Subscription();
  Constructor ( private productService : ProductService ) {
  }
  ngOnInit() {
    this.inputs = this.productService.getProducts();
    this.productSubscription = this.productService.productUpdated.subscribe(() => {
        this.products = this.productService.getProducts();
    });
  }
  onAddProduct ( form: any) {
    this.productService.addProduct( ... );
  }
  ngOnDestory() {
    this.productSubscription.unsubscribe();  when unmount
  }
```

· sub component

```
  :
  Constructor( private productService: ProductService) {
  }
  :
  onDeleteProduct(i:number) {
    this.prodadService.deleteProduct(i);
}
```

Angular Routing

create app-routing.module.ts

```
Const routs: Routes = [
    { Path: "" , Component : HomeComponent };
    { Path: "product", Component : ProductsComponent }
]
export class AppRoutingModule { }
@NgModule ({
        imports: [RouterModule.forRoot(routes)],
        exports: [RouterModule]
})
```

on app.module.ts
add AppRoutingModule
on imports

CanActivate: [ ]
go to Page 20

ngular Material

is a third Party Package, which you can use in angular Project

ngular Component Suite

It is a collections of pre-built and styled angular components

[material.angular.io]

nstall Angular Material & Angular CDK

npm install --save @angular/material @angular/cdk

أي component بدي أستخدمه
لازم أعمله import

tall Angular Animations

npm install --save @angular/animations

add animations [BrowserAnimationsModule] to app.module.ts
to Import Array

Way to manage all material pkg

create material.module.ts

```
@NgModule({
  imports: [module1, module2, ...],
  exports: [module1, module2, ...]
})
export class MaterialModule{
```

on app.module.ts

add MaterialModule to imports

هاي الطريقة أسهل من إنه اضيف كل module لي بدي اياها بـ app.module لي imports

clude a theme

add this import to styles.css

```
@import "~@angular/material/prebuilt-themes/indigo-pink.css";
```

file موجود أكثر من لـ node-modules بالـ
ممكن أستخدم أي واحد

esture support

npm install --save hammerjs "need it for mobile"

on main.ts add import 'hammerjs'

d Material Icons (option)

add this to index.html

```
<link href="https://fonts.googleapis.com/icon?family=material+icon"
      rel="stylesheet">
```

Button Component

1. add MatButtonModule to material.module.ts

2. <button mat-button> Basic </button>

   <button mat-button color="primary"> Primary </button>

   there are other colors
   such as : accent, warn,

   <button mat-button disabled> disabled </button>

   <a mat-button href="#" target="_blank"> link

3. there are another types of buttons such as:

   mat-raised-button
   mat-strocked-button
   mat-flat-button

tton Component Icon

<button mat-icon-button color="primary">
<mat-icon> home </mat-icon> → use Icon
</button>

*other Types:
mat-fab, mat-mini-fab,

rm Tips

1. <input matInput type='text' #name />
<mat-hint> {{ name.value ?.length || 0 }}/50 </mat-hint>

2. <mat-label> Gender: </mat-label>
<mat-radio-group name="gender" ngModel>
<mat-radio-button value="Male"> Male </mat-radio-button>
:
</mat-radio-group>

3. <input matInput [formControl] = "email" />

<mat-error *ngIf= "email.invalid" > {{ getErrorMessage() }} </mat-error>

---

email = new FormControl ('', [Validators.required, Validators.email]);

getErrorMessage () {
  if (this.email.hasError ('required')) {
    return "you must enter a value";
  }
  return this.email.hasError('email') ? 'Not a valid email' : '';
}

- FlexLayout

⇒ npm install --save @angular/flex-layout

⇒ add FlexLayoutModule to imports in app.module.ts

Same as flex in css

⇒ <form fxLayout= "column" fxLayoutAlign = "center center">

  ⋮

</form>

datePicker

⇒ add MatDatepickerModule, MatNativeDateModule to material.module.ts

⇒ <mat-form-field>

matSuffix

<input matInput placeholder= "Your birthdate" [matDatepicker] = "Picker" />

<mat-datepicker-toggle [for] = "Picker"></mat-datepicker-toggle>

لحتى يكون الـ مناسب

<mat-picker #Picker></mat-picker>

</mat-form-field>

button toggle

⇒ add MatButtonToggleModule on material.module.ts

⇒
```
<mat-button-toggle-group>
    <mat-button-toggle value="x">
      x
    </mat-button-toggle>
    <mat-button-toggle value="y">
      y
    </mat-button-toggle>
</mat-button-toggle-group>
```

→ add `multiple` to select multiple buttons

→ add `apperance="legacy"`

Dialog

1
```
<button mat-raised-button (click)="openDialog('3000ms', '1500ms')">
    open Dialog
</button>
```
bullet 8gi Page 12

2
```
export class DialogAnimationExample {
    constructor(public dialog: MatDialog) {}

    openDialog(enterDuration: string, exitDuration: string) {
        this.dialog.open(DialogAnimationExampleDialog, {
            width: '250px',
            enterAnimationDuration: enterDuration,
            exitAnimationDuration: exitDuration });
    }
}

export class DialogAnimationExampleDialog {
    constructor(public dialogRef: MatDialogRef<DialogAnimationExampleDialog>)
    {}
}
```

3

```
<h1 mat-dialog-title> title </h1>
<div mat-dialog-content>
    write content of dialog here
</div>
<div mat-dialog-actions>
    <button mat-button mat-dialog-close> No </button>
    <button mat-button mat-dialog-close cdkFocusInitial> ok </button>
</div>
```

4 make content scrollable

```
<mat-dialog-content class="mat-typography">
  :
</mat-dialog-content>
```

Align dialog Actions

```
<mat-dialog-actions align="end">
  :
</mat-dialog-actions>
```

```
class: DialogContentExample

onOpenDialog() {
  const dialogRef = this.dialog.open(
    DialogExampleContentDialog);
  dialogRef.afterClosed().subscribe(
    result => {
      → get result from dialog
    });
}
```

```
class: DialogContentExampleDialog
  empty class
```

Pass Value to Dialog and read result from Dialog

```
openDialog(): void {          DialogHost
  const dialogRef = this.dialog.open(Dialog, {
    width: '250px',
    data: { name: this.name, age: this.age },
  });
  dialogRef.afterClosed().subscribe(result => {
    this.age = result;
  });
}
```

Dialog

```
onstructor(Public dialogRef: MatDialogRef<DialogHost>,
    @Inject(MAT_DIALOG_DATA) Public data: DialogData) {}
```

```
<input matInput [(ngModel)]="data.age" />
⋮
<button mat-button [mat-dialog-close]="data.age"> ok </button>
```

on close return this value, so the result on host dialog is data.age

Expansion Panel

```
<mat-accordion multi>
```

multi: expand multi Panels at same time

```
⋮   ← add here all Expension Panel
</mat-accordion>

<mat-expension-Panel [expanded]="true" (opened)="fun()" hideToggle>
```

disabled: if you want to disable Panel

[expanded]="true": true if you want to expand Panel

(opened)="fun()": run function when Panel is open

hideToggle: to hide arrow icon on right of Panel ∨ ∧

```
⋮
</mat-expension-Panel>

<mat-expension-Panel-header>
    <mat-Panel-title> title </mat-Panel-title>
    <mat-Panel-description>
        description
        <mat-icon> home </mat-icon>
    </mat-Panel-description>
</mat-expension-Panel-header>
```

title description ∨

show it if not set hideToggle

```
<div>
    content of Panel
</div>
<mat-action-row>
    <button mat-button (click)="fun1"> X </button>
    <button mat-button (click)="fun2"> Y </button>
</mat-action-row>
```

Access Expasion panel on module

```
<mat-accordion #firstAccordion="matAccordion">
  !
</mat-accordion>
```

↳ no need it just #firstAccordion

```
@ViewChild("firstAccordion")
accordion!: MatAccordion;
```

} يمكن أستخدم هاي الطريقة لأي html عنصر element

Paginator

```
<mat-Paginator [1] (Page)="onchange()" [2] [length]="100" [3] [pageSize]="5"
  [4] [PageSizeOptions]="[5,10,15,20]" [5] #PageEvent>
</mat-paginator>
```

[1]: event when change Page or change Page size

[2]: number of all element.

[3]: the size of each Page

[4]: list of possible Page size

[5]: make referance on component

Items Per Page [5] 1-5 of 100 < >

```
<div>
  {{ PageEvent.PageSize }}
  {{ PageEvent.PageIndex }}
  {{ PageEvent.length }}
  {{ PageEvent.previousPageIndex }}
</div>
```

```
@ViewChild("PageEvent")
PageEvent: PageEvent;

onchange() {

}
```

Menu

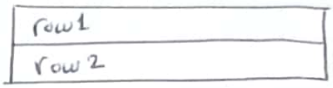
```
<button mat-button [matMenuTriggerFor]="menu"> click here </button>

<mat-menu #menu="matMenu" yPosition="above">
    <button mat-menu-item> item 1 </button>
    <button mat-menu-item> item 2 </button>
</mat-menu>
```

the Position of menu
- above
- below
- before
- after

ToolBar

```
<mat-toolbar>
    <mat-toolbar-row>
      ⇒ any thing
    </mat-toolbar-row>
    <mat-toolbar-row>
      :
    </mat-toolbar-row>
</mat-toolbar>
```

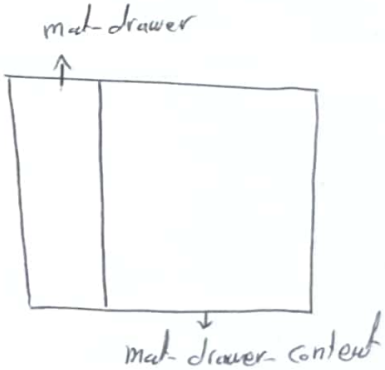
⇒ ToolBar have 2 rows



deNav

```
<mat-drawer-container>
    <mat-drawer mode="side" opened>
      ⇒ any thing
    </mat-drawer>
    <mat-drawer-content>
      ⇒ any thing
    </mat-drawer-content>
</mat-drawer-content>
```

render ways side, over, Push



lection List

```
<mat-selection-list #list [multiple]="false">
    <mat-list-option [value]="list" [selected]="true"> X
    </mat-list-option>
    <mat-list-option [value]="list"> Y
    </mat-list-option>
</mat-selection-list>
```

can select multiple options

option default false

left, right, center

Tab

```
<mat-tab-group  mat-align-tabs="left"  animationDuration="500ms">

    <mat-tab label="First"> content 1 </mat-tab>
    <mat-tab label="Second"> content 2 </mat-tab>

</mat-tab-group>
```

label → Name of Tab

content → Content of Tab

~~to read value~~

Card

```
<mat-card>
    <mat-card-title> Title </mat-card-title>
    <mat-card-content> content </mat-card-content>
    <mat-card-action> .... </mat-card-action>
</mat-card>
```

ist Select

```
<mat-form-field>
    <mat-select placeholder="favorite Framework"  #selectedValue>
        <mat-option value="react">
            React
        </mat-option>
        <mat-option value="angular">
            Angular
        </mat-option>
    </mat-select>
</mat-form-field>
```

#selectedValue → To access the selected Value

{{selectValue.value}}

Progress spinner

to Access it on Module

```
<mat-progress-spinner #spinner color="accent" mode="determinate" value="25">
</mat-progress-spinner>
```

set it when mode is determinate

fixed width and can use indeterminate value

```
@ViewChild("spinner")
spinner: any;
```

CanActivate - Route

```
{ path: '', component: Component, canActivate [C1, C2, ...] }
```
Guards

if all guards return true navigation continues if any guards return false, navigation is cancelled

```
export class Guard implements CanActivate {
    constructor(private router: Router) {
    }
canActivate(route: ActivateRouteSnapshot, state RouterStateSnapshot) :
  Observable<boolean> | Promis<boolean> | boolean {
    .
    .
    .
    return true;
}
```

children - Route

```
{ path: "x", component: Component, children [
    { path: "y", component: Component },
    { path: "z", Component: Component }
]}
```

typeScript - enum

```
enum Gender {
    MALE,
    FEMALE
}
```

Gender.MALE 0 } number
Gender.FEMALE 1
Gender[Gender.MALE] "MALE" } string
Gender[Gender.FEMALE] "FEMALE"

TranslateService

1. npm install @ngx-translate/core @ngx-translate/http-loader
2. import { TranslateService } from '@ngx-translate/core'

```
export class TranslateExample {

    constructor() {}
    private translate: TranslateService
    ngOnInit() {
        TranslateExample.LOCALE =
        (this.translate.currentLang == 'ar') ? 'ar' : '';
    }
}
```

3. on assest folder make new folder i18n then make these files
   - ar.json
   - en.json

4. this.translate.get("hello-world")

```
{{ 'hello-world' | translate }}
<element [translate]="'hello-world'"></element>
<element translate>id</element>
```

5 Using Parameters

{{ 'id' | translate {parameter: 'value'} }}

{{ 'hello' | translate {name: 'Ayah'} }}

en.json ⇒ {

"hello": "Hello {{name}}"; ⇒ Hello Ayah

}