Angular

by Google

* ال angular هو client side framework
هو framework مبني بيتم application بيشتغل
من ال client-side

* فيه اساسيات لازم تكون عارفها قبل ما تبدأ Angular :

① لازم تكون عارف TS كويس جدا
② ~ ~ ~ oop كويس جدا
③ ~ ~ ~ design patterns

الممسوحة ضوئيا بـ CamScanner

intro

session ①

1- angular intro
2- angular setup
3- angular files
4- SPA technique / concept
5- angular architecture
6- project flow
7- install bootstrap 4 fontawesome

الممسوحة ضوئيا بـ CamScanner

① Angular Intro

ايه هو ال angular ؟

ال angular هو development framework مبني على لغة ال typescript ، الهدف منه بناء single page application ، وبيحتوي على libraries بتسهل عملية التعامل مع ال API وال validation والتعامل مع ال DOM وال testing .....

ايه الفرق بين angularjs و angular2 ؟

ال angular ظهر في 2014 كتحديث لل angularjs ، لكنه كان تحديث مختلف تماما الاختلاف عن ال angularjs كان عملية هدم كامل لل angularjs وبناء framework جديد تماما باسم Angular2 .
ال angular مبني بال cs .
ال angular2 مبني بال ts وال performance بتاعه اعلى من angularjs

ايه هو ال @angular/cli ؟ (ng) بنحتاجه في مرحلة ال development فقط

ال cli angular هي tool بتسهل عملية تطوير وبناء ال angular applications من خلال ال command shell .
cli هو اختصار لـ command line interface .

ايه هو ال angular routing ؟
ال routing هو التنقل بين الصفحات / تغيير ال view اللي بيتعرض في ال SPA عن طريق ال URL path

② Angular setup

* pre-requistes? ① HTML, CSS, JS, TS
② Node.js / npm LTS long term support

* install the Angular CLI in your machine

```
>> npm i -g @angular/cli
```

(-g) يعني global يعني أعمل install على مستوى الجهاز كله، يعني أعمل install في الـ (C:)

* create angular project:

1- create a workspace by using CLI

```
>> ng new app1
>> cd app1
```

2- `>> ng serve --open` → run local server

3- create components

```
>> ng g c navbar
```

4- `>> npm i package` ← اسم الـ dependencies هينزل مع الـ
`>> npm i pack-name --save-dev`
هينزل مع الـ devdependencies يعني هيشتغل في مرحلة الـ development فقط، ومش هيترفع في مرحلة الـ production / عالشغل deployment

5- `>> npm i` ← هينزل كل اللي موجود في الـ package.json

الممسوحة ضوئيا بـ CamScanner

(3) angular files

① node_modules ← بيحتوى على كل الـ node dependencies / كل الـ packages اللى بنحتاج الـ install بالـ npm
ودية الـ packages اللى بنحتاجها فى المشروع.

② src ← دى عبارة عن الـ source files.
بيحتوى على:
① app folder ← بيحتوى على كل الـ components وكل الداتا وكل الـ app logic

③ assets ← بيحتوى على الـ static files زى الصور والـ icons والـ css والـ js و....

④ favicon.ico ← دى الـ icon اللى بيظهر جنب الـ tab.

⑤ index.html ← دى الـ entry point، دى الـ main html file

⑥ .gitignore ← بيحتوى على الـ path بتاع الـ files والـ folders اللى مش هيتعمل track يعنى مش هيتبعتوا للـ version control

⑦ package.json ← بيحتوى على اسامى الـ packages اللى المشروع بيعتمد عليها والـ packages اللى بنحتاجها فى مرحلة الـ development ومرحلة الـ production.
دى الـ reference بتاع دى ملف الـ requirements

④ SPA technique

قبل ال SPA concept :

مقدمة :

* أي HTML page بنعوز نتوصل بنبعت ال Http request الخاص بال page دية وال server هيرد ب Http response عبارة عن ملف ال Html والملفات اللي ال Html بيستخدمها ، ال Browser بيعمل render للملفات دية بالتالي لازم هيعمل refresh أو reload ال page عشان يعرض الملفات الجديدة / ال page الجديدة .

* كل html page بتوصلة ب http request
* كل request هيرجع ب response
* كل response هيعمل reload للصفحة الحالية .

ايه المشكلة ؟ العملية بين ال request وال response بتاخد وقت بالتالي دة هيأثر على performance ، عملية ال refresh هتأثر على ال user experience

ايه هو ال SPA ؟ التعامل مع كل ملفات ال html كأنها ملف html واحد .

ال spa هو technique / concept فكرته إنه ازاي نقلل عدد ال requests اللي بتتبعت قدر الإمكان بالتالي نقلل عدد ال refresh اللي بيحصل ، ودة عن طريق ان احنا نخلي ال server يبعت كل صفحات ال html مرة واحدة مع اول request .

الممسوحة ضوئيا بـ CamScanner

(5) Angular Arch.

ال angular بيشتغل على : component-based-arch.
ودة معناه إن الوحدة الاساسية / ال main unit لاى مشروع
مبنى بال angular هو ال component.

يعنى ممكن نقول إن ال angular عبارة عن مجموعة components
ولو حاجات تانية بتخدم ال components دية.

ايه هو ال component؟

١- ال component هو جزء معين من مشروع ال angular
والجزء دة بيكون scalable يعنى مجمع فى المشروع
ممكن يصغر او يكبر.
٢- ال component بيكون reusable يعنى أقدر أعيد إستخدامه
أكتر من مرة فى المشروع عن طريق ال selector بتاعته.

٣- كل component من ال angular بيتكون من
3 files اساسيين :
① ملف ال html
② " " " css
③ " " " ts

٤- كل component بيحتوى على selector خاص بيتكون عبارة
عن custom html tag (يعنى tag أقدر أسميه بأى اسم)
من طريق ال <selector> نقدر نوصل ال component بتاعته.
ال <selector> بيتشال وبيتحط مكانه ال html بتاع ال component
(بصحة ال call بتاع ال function) ☺

الممسوحة ضوئيا بـ CamScanner

ايه هو ال app component ؟

ده ال root component او ال parent component
اللي بيحتوي على كل ال components بتاعة المشروع،

ال app component بيتم انشائه by default عن
طريق ال angular.

ال selector الخاص بال app component هو اللي
بيتكتب في ملف ال index.html وعن طريقه بنوصل
لكل ال components التانية.

ايه هو ال app module ؟ NgModule

ال module هو مجموعة من ال components،
لازم ال component يبقى تبع module،

ال app.module.ts هو ال default module اللي بينزل
مع ال Angular، لازم نعرف ال component في ال
declaration بتاع ال app module عشان يتم التعرف
على ال Angular component بالتالي نقدر نستخدم
ال selector بتاعه،

ال angular component هو عبارة عن class
، بيتم تمييز ال class على إنه component عن
طريق decorator function إسمه @Component

الممسوحة ضوئيا بـ CamScanner

app component

app.component.html
app.component.css
app.component.ts
<app-root></app-root>

<app-nav></app-nav>
<app-home></app-home>

nav component

nav.component.html
nav.component.css
nav.component.ts
<app-nav>

home component

home.component.html
home.component.css
home.component.ts
<app-home>

index.html

<app-root></app-root>

NgModule

ال modules ببساطة هو إن عندي أكثر من file عايزهم كلهم يشتغلوا ببعض، فبعمل import من ال module

⑥ project flow

* ال component ما هو إلا typescript class ،
* بيتم التعامل مع ال class / أو بيتم تمييز ال class على إنه angular component عن طريق decorator اسمه "Component" بيتم استيراده من module اسمه "@angular/core".
* عن طريق ال Component decorator @ بيتم:
  ① تمييز ال class اللي بعده مباشرة" ك component.
  ② بيحتوي على ال metadata الخاصة بأي component زي:
  ① اسم ال selector الخاص بال component
  ② مكان ال html file ~ ~ ~
  ③ مكان ال css file ~ ~ ~

(ex)

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

@Component ({
    selector: 'app-home',
    templateUrl: './home.component.html',
    styleUrls: ['./home.component.css']
});
export class HomeComponent {

}
```

* أي angular component لازم نعرفه جوا angular module علشان قدر نستخدم ال selector بتاعه.

الممسوحة ضوئيا بـ CamScanner

recap

١- ال angular قائم على فكرة ال component-based-arch، وده معناه إن مبقاش عندنا حاجة اسمها page لكن عندنا حاجة اسمها component.

٢- كل component بيتكون من: ① html file
② css file
③ ts file

٣- لازم أي component يتسمى بـ selector عشان من خلاله هنقدر نوصل ال component من أي مكان تاني.

٤- by default ال angular بيكون معاه default component اسمه app component.

٥- عشان ننشأ أي component جديد بنستخدم ال angular/cli ← (ng g c home ←)
الأمر ده هيخلي ال component بكل ال files وال configuration بتاعته وحط ال class بتاعه في ال app module.

٦- ال angular module هو ال container بتاع ال components
أي component لازم يكون تبع angular module.

٧- ملف ال index.html بيحتوي على ال selector بتاع ال app component اللي هيحتوي على كل ال components بتاعة ال project، مش هنحرب من ال index file خالص.
كل حاجة هتبقى في ال html file بتاع ال app component

الممسوحة ضوئيا بـ CamScanner

٨- ملف package.json ده أهم ملف في المشروع
لأن الملف ده عبارة عن ال history لأي package
اتعملها npm install ، الملف ده بيكون فيه ١
١- اسم ال packages اللي اتعملها install
٢- ال version بتاع كل package منهم

٩- عشان نعمل install لكل ال packages في ملف
ال package.json بنكتب ← لـ: npm

١٠- ال module في ال angular هو عبارة عن مجموعة
components مع بعضها.

١١- ال decorator ده function بيضيف behavior
و metadata لل class.

هنستخدم ال decorator عشان نضيف معلومات
عن ال component او عن ال file اللي فيه.

١٢ ال decorator الخاصة بال component ← @Component()
" " " " " module ← @NgModules()

١٣ ملف ال index.html مش هنقرب منه

١٤ في ال VS code ممكن نفتح 2-terminals.
* الأول عشان بـ run ال server.
* الثاني عشان نستخدم ال ناك في اي حاجة ثانية.

الممسوحة ضوئيا بـ CamScanner

(7) angular.json

* ال angular.json هو ملف بيكون فيه ال configuration الخاصة بال angular زي مثلا :

① اي component هتعملها create من طريق ال cli هتكتب قبل اسمه app-.

② ملفات ال .css الخارجية زي ال bootstrap .... etc
③ ملفات ال .js الخارجية زي ال jQuery .... etc

* اي تغيير في ملف ال angular.json ، عشان يتسمع لازم نقفل ال server ونشغله تاني .

* جوا ملف ال angular.json فيه property :
١- اسمه (styles) نوعه (array) بنكتب فيه ال relative path بتاع ملف ال .css الخارجي .

٢- اسمه (scripts) نوعه (array) بنكتب فيه ال relative path بتاع ملف ال .js الخارجي .

* ملف ال angular.json ال configurations اللي فيه بتسمع على كل ال angular components ، عشان كده إحنا بنربط ملفات ال .css وال .js الخارجية في ملف ال angular.json بحيث يتسمع في كل ال components .

الممسوحة ضوئيا بـ CamScanner

(ex) How to install bootstrap?

1-) >> npm i bootstrap ↵

2-) in angular.json file:

```
{ ≡
  "styles" : [
    "./node_modules/bootstrap/dist/css/bootstrap.css" ,

  ],
  "scripts" : [
    "./node_modules/bootstrap/dist/js/ ——— .js" ,

  ]
≡
}
```

3-) اقفل ال serve واعمل run تاني
عشان التغييرات تتحفظ وتتسمع
في كل ال components.

>> ctrl + c

>> ng serve -o

ملحوظة: جوا ملف ال angular.json فيه أكثر من "styles" و "scripts"، إتأكد إنك
كاتب ال path بتاع ال css أو ال js في
أول property "styles" & "scripts"

الممسوحة ضوئيا بـ CamScanner

(ما هو ال webpack؟)

* هي tool بتعمل packaging لملفات ال html وال css وال js، بمعنى إنها بتجمع كل ملفات ال html من المشروع في ملف html واحد، وكل ملفات ال js في ملف js واحد، وكل ملفات ال css في ملف css واحد مع مراعاة ترتيب الملفات.

* ده هيكون مرعب جدا لو كتبنا ال path بتاع ملفات ال css وال js في ملف ال index.html بالتالي يفضل تجميعهم في مكان واحد، في حالتنا المكان ده هو ال (angular.json)

* ال module بيربط ال components مع بعضهم.

* ملف ال ts هو رئيس مجلس إدارة ال component

```
>> ng new p1 --style=scss
>> cd p1
>> ng add @angular/material
```


الممسوحة ضوئيا بـ CamScanner

13

session ②

1- typescript intro
2- typescript datatypes
3- genaric types
4- interface , class , type
5- enum
6- oop
7- decorator
8- tsconfig

الممسوحة ضوئيا بـ CamScanner

① TypeScript intro

* ts is js + more.

١- ال typescript هو لغة برمجة مبنية على لغة ال كز.
ال ts ما هي إلا ال كز مع features زيادة.
ts = js + typed + interface

٢- ال ts هي strongly typed يعني لازم نحدد نوع ال variable.

٣- ال Browsers مبتفهمش ال ts بالتالي لازم نحول
ال ts لـ كز عشان ال Browser يفهمه، وده عن
طريق transpiler حول هيحول ال ts لـ كز.

٤- ايه ميزة ال typescript؟

* ال ts هيحسن كتابة الكود عن طريق إنه:
① بيظهر ال errors في مرحلة ال development يعني
قبل مرحلة ال runtime على عكس ال كز.

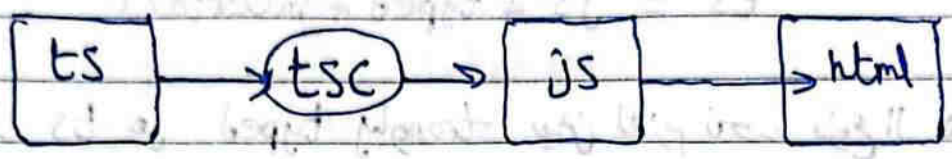
② بيخلي الكود يبقى predictive يعني متوقع نتيجته ايه.

* ال ts خلى الكز أقرب للغات البرمجة التانية.

٥- بنستخدم ال ts في مرحلة ال development فقط، لكن في
مرحلة ال production لازم نحوله لـ كز عن طريق ال transpiler.

```
>> npm i -g typescript ↵
>> tsc test.ts ↵ → حول لـ كز فقط
>> tsc test.ts -watch ↵ حول لـ كز وتابع أي تغيير
```

٦- لغة الـ ts بيسموها development tool يعني tool بتساعد الـ developer في مرحلة الـ development فقط، لكن مبنستخدمش في الـ production.



٧- اللي عمل لغة الـ ts هي شركة microsoft واللي عامل هو نفس اللي عمل لغة الـ #C بالتالي لغة الـ ts قريبة من الـ #C، عشان كده بتلاقي الـ developers بيفضلوا يستخدموا الـ angular مع الـ .net.

٨- الـ ts ماهي إلا الـ js، يعني أقدر أكتب أي كود js جوا ملف (ts).

الـ ts إتعمل من أجلك انت يا developer عشان يخلي كود الـ js predictive، ويظهرلك الـ Error قبل الـ runtime.

اهم ميزة للـ ts انه typed.
الـ ts ظهر كبديل للـ js لتحسين كتابة الـ js وحل مشاكل الـ js.

الممسوحة ضوئيا بـ CamScanner

② typescript datatypes :

```
let name : datatype = value ;
```

```
① let x : string = 'hello' ;
② let x : number = 22 ;
③ let x : boolean = true ;
④ let x : string[] = ['hi', 'hello'] ;
⑤ let x : object = {}
```

⑥ tuple type → mix b/w types , الترتيب مهم

```
(ex) let x : [string , number] ;
     x = ["toshipa" , 159] ;
```

⑦ بنستخدم any في حالة إني مش عارف ايه الـ data اللي جاية

```
let x : any ;
let x : unknown ;
```

⑧ union type → may be type1 or t2 or ...

```
(ex) let x : string | null | undefined ;
```

قيمة الـ X ممكن تكون string أو null أو undefined

```
(ex) let x : string[] | number[] ;
```

mixed array

```
(ex) let x : (string | number) [] ;
     x = ['hi' , 15 , 20 , 'llb'] ;
```

```
let x : number[] ; ↔ let x : Array <numbers> ;
```


الممسوحة ضوئيا بـ CamScanner

③ generic types → على حسب

```
function test(x:number):void{
}
function test(x:string):number{
    return 20;
}
```

* الـ parameter بتاعي نحدد ليه type.
* الـ function دقدر نحدد نوع الـ data اللي هترجع.

* generic يعني حاجة عامة
* عن طريق الـ generic type دقدر نثبت الـ function الـ type اللي هيشتغل عليه.

```
function test<hambozo>(x:hambozo[]):hambozo[]{
    return ;
}
test<string>(['A','b','c']);
test<number>([1, 2, 6, 11]);
```

* جرى العرف ان احنا نسمي الـ generic type باسم Type
* generic type يعني type عام بيختلف حسب call التاني.

الممسوحة ضوئيا بـ CamScanner

(4) interface.

المشكلة؟ لو عملنا array of object بالتالي كل object من ال array ممكن يحتوي على property مختلفة عن التاني، وكل property ممكن ياخد datatype مختلفة عن التاني.

دلوقتي مش قادرين نتحكم في عدد ال property من ال object وال datatype بتاع كل property.

الحل للمشكلة دي هو ال interface.

ايه هو ال interface؟

* ال interface هو: custom typing بيسمحلنا نفرض او نفرض structure معين على ال object.

* ال interface هو tool في ال typescript بيسمحلنا نحدد بنية ال object بحيث نقدر نحدد:
1- ايه هي ال properties اللي لازم تكون في ال object؟
2- " " " " اللي ممكن " " " " ؟
3- " هو ال datatype بتاع كل property؟
4- " " " " ال method؟
5- " " " " ال return؟

* ال interface مش object او class، دي custom type.

ازاي نعمل interface ؟

(1)

```
interface Product {
    name: string;
    price: number;
    onSale: boolean;
    id(?): number;
        ↳ optional
    buy(): string;
}
```

* لو كتبنا interface بنفس الاسم، مش هيحصل overriding ولكن هيحصل inheritance يعني هورث من ال interface اللي واخد نفس الاسم.

```
Interface Product {
    category?: string
}
```

→ هورث من ال interface اللي فوقية ويزود عليها property جديد اسمه category.

```
let x: Product[] = [
  {name: 'TV', price: 600, onSale: true,
    buy: function(){
        return "hi";
    }, ....]
```

* ال interface مش class لأني مقدرش أعمل منه object لكن هو custom type يعني بيعمل ابن datatype جديدة.

الممسوحة ضوئيا بـ CamScanner

(2)

```
class Product {
    name : string ;
    price : number ;
    onSale : boolean ;
    id ? : number ;
    buy () : string ;
}
```

Q) ايه هو الفرق إن انا عمل custom type باستخدام ال class
وإن انا عمل custom type باستخدام ال interface ؟

الفرق الجوهري إن ال class هيتحول ل function
في ال js ومش هيكون لية استخدام في ال js لأنة
معمول ك custom type في ال ts. مش أكتر.

لكن ال interface مش هيتحول لأي حاجة في ال js
لأن هو مجرد custom type.

* الفكرة من ال interface ان احنا نعمل custom type
نحدد بية شكل ال object / ال structure بتاع ال object
في مرحلة ال development ، لكن في مرحلة ال production
لما نحول ملف ال ts ل js ال object هيرجع زي ما كان
يعني ملهوش structure محدد.

الممسوحة ضوئيا بـ CamScanner

(3)

```
type Product = {
    name : string ,
    price: number ,
    onSale : boolean ,
    id? : number
}
```

* ال type هو نفس ال interface الفرق الوحيد
إن ال type مقدرش آلدرة عشان اورث منه
كي ال interface اورث منه ازاي؟

١- هشكل type جديد باسم مختلف
٢- هيساوي الاسم بتاع ال type اللي عايز اورث منه
and & ال property الجديدة.

```
type Product2 = Product & {
    category : string ,
}
```

* الاختلاف بين ال interface وال type هو
طريقة ال inherit.

الممسوحة ضوئيا بـ CamScanner

٧ ايه هو الفرق بين ال interface وال class وال type ؟

١- ال class من ال compile بتحول لـ function .

٢- ال interface من ال compile مبتحولش حاجة .

٣- ال type هو نفس ال interface لكن طريقة ال inherit مختلفة .

⑤ enum

١ ال enum مش object أو class ما هو إلا نوع من انواع ال collection .

٢ ال enum بيسمحلي اعمل list من values محددة

٣ ال user يقدر يختار منهم .

```
(ex) enum Color{
        Red,   //0
        Green, //1
        Blue   //2
    }
```

٤ bydefault ال enum بيبدي numeric values بالترتيب للقيم اللي فيه .

٥

٦ let bgColor:Color; Color[0]

bgColor = Color.Red; ⟺ bgColor = 0;

* ال enum هو custom typing بيسمحلنا نعمل datatype جديد بيقبل قيم محددة .

(6) OOP in TS:

الـ syntax بتاع الـ classes والـ object هو نفس الـ syntax تبع ES6، لكن مع features زيادة، اهم feature إن فيه access modifiers الـ members تبع الـ class.

(ex)

```
class Car {
    private chasisNo:number;
    protected motorNo:number;
    public wheel:number;
    constructor(wheel:number){
        this.wheel = wheel;
    }
    disp():void{

    }
}

let obj1 = new Car(15);
```

① * كل الـ properties بتبقى public by default.

② * public يعني نقدر نـ access الـ property خارج الـ class

③ * private ~ مش قدرتش ~ ~ ~ ~ ~ ~ ~ بشكل مباشر، لكن نقدر نعمله access عن طريق method.

④ * readonly يعني نقدر نـ access الـ property داخل وخارج الـ class لكن مش قدرتش نغير قيمته

الممسوحة ضوئيا بـ CamScanner

* ال interface يحدد بنية ال structure اللي المفروض ال class أو ال object يبقى عليه.

* ال interface بتعرف فيه ال signature بتاع ال function فقط بدون body، والـ implementation بتعمله في ال object أو ال class.

* ال class يقدر يورث من class واحد فقط لكن يقدر يـ implement أكثر من interface.

```
interface I1 { }
interface I2 { }
class C1 { }

class C2 extends C1 implements I1 , I2 { }
```

* ال interface يقدر يورث من أكثر من interface.

```
interface I3 extends I1 , I2 { }
```

* إيه الفرق بين ال modules وال namespaces؟

الاثنين الهدف منهم هو تنظيم ال program عن طريق تقسيم الكود على أكثر من file وكل file بيختص بحاجة معينة.

الفرق إن ال files اللي متقسمة بـ namespace في ال combile كلهم هيتجمعوا في file واحد.

(٤) decorator

ال decorator عبارة عن design pattern بيدينا طريقة ان اضيف attach / نضيف حاجة في ال class او في ال function لكن في ال runtime يعني من غير ما نغير حاجة في ال class أو في ال function.

① ال decorator هو special kind of declaration نقدر نربطه / نعمله attach مع class أو method او property أو parameter.

② الهدف من ال decorator إنه يضيف functionality او behavior معينة لل class أو ال method أو ...

③ فيه أكتر من نوع ل decorator،
← ال class decorator بيتعرف قبل ال class مباشرة
← ال function ~ ~ function ~ ~

(الخلاصة) ال decorator هو عبارة عن declaration بنحطه قبل ال class أو قبل ال method أو ... عشان يضيف functionality زيادة على ال class أو على ال method أو ...

(مثال) في ال angular عشان نخلي class يتصرف ك component بنضيف قبل ال class ده decorator @Component()

الممسوحة ضوئيا بـ CamScanner

(8) tsconfig.json

- في المشاريع الكبيرة هيكون فيه files ts كتيرة جداً، وكلها لازم هيتحولوا لـ files js، وده شكل معقد شوية.

- الافضل إن احنا نفصل ملفات الـ ts في folder لوحده اسمه (src) وده الـ folder اللي هيحتوي على كل الـ files اللي مش هيتعملهم deploy.
وهنحط كل ملفات الـ js / الملفات اللي هيتعمل deploy في folder اسمه (public).

- ازاي ما نقل combine الملفات الـ ts جوا folder الـ src وازاي نخلي ملفات الـ js تتحط جوا folder الـ public؟

ده عن طريق ان احنا نعدل في الـ configuration الخاصة بالـ tsc، بالتالي اولاً لازم نعمل الـ config file الخاص بالـ tsc عن طريق الامر (tsc --init)

```
1) tsc --init ↵
2) open the tsconfig.json file ->
3) { "compilerOptions": {
        "rootDir": "./src",
        "outDir": "./public",
    }
4) tsc ↵
   tsc -w ↵
```

هيفتش جوا الـ rootDir ويحول كل الـ files ts لـ files js ويحطها في الـ outDir.

How to generate interface with cli?

1-) » ng g i user ↵

2-) user.ts ↵ export interface User {
}

3-) import {User} from './../user';

ليه نعمل file خاص لل interface؟

* لانه افضل كتنظيم وكـ reusability، حيث اقدر اعمله import واستخدمه من اى component ولو عملت تعديل هيسمع فى كله.

* يفضل نقلل استخدام الـ object type وبدالة نستخدم interface.

الممسوحة ضوئيا بـ CamScanner

routing intro

Session ③

Binding
Directives
pipes

1. binding
2. directives
   revision
3. angular routing

الممسوحة ضوئيا بـ CamScanner

① binding (عملية ربط كود الـ ts بكود الـ html)

* الـ binding هو feature في الـ angular وهو عبارة عن عملية ربط بين الـ ts والـ html.

* الـ binding ينقسم لنوعين:

① one way data binding	② two way data binding
"هي عبارة عن الـ binding المسؤول عن ارسال الـ data من الـ ts إلى الـ html."	الـ binding المسؤول عن ارسال الـ data من الـ ts للـ html والعكس.
ts $\xrightarrow{\text{one way}}$ html	ts $\xleftrightarrow{\text{two way}}$ html

One way binding {{}} , [] , ()

(1) data interpolation {{variable name in ts}}
(2) property binding [attribute] = "variablenameints"
(3) event binding (eventname) = "methodname()"

(ex)

```
class HomeComponent {
    xinumber = 20;
    winumber = 100;
    n:string = 'hello';
    m1(e:Event){
    }
}
```

```
<h1>{{n}}</h1>
<img [width]="w" ---->
<button (click)="m1($event)" ---->
```

style binding & class binding (property binding)

① style binding

```
class HomeComponent {
    mainColor = 'red';
    onSale = true;
    mainStyle = 'color:red; font-size:30px;';
}

<h1 [style.color] = "mainColor" ></h1>
<h1 [style.color] = "onSale? mainColor : 'blue'" ></h1>
<h1 [style] = "onSale? mainStyle : mainColor" ></h1>
```

② class binding

```
<h1 [class] = "onSale? 'bg-danger' : 'w-100' " ></h1>
```

• two way data binding

① import FormsModule
② [(ngModel)] = 'name'

ملف الـ app.module.ts بنستخدمه :
① عشان نربط الـ components ببعض.
② عشان نعمل import للـ built-in modules
اللي هنستخدمها مع الـ components.

الممسوحة ضوئيا بـ CamScanner

رايح جاي two way binding [( )]

الـ two way إنك أعمل تغيير في الـ html يـ update في الـ ts، ومن الـ ts الـ update يسمع في الـ html.

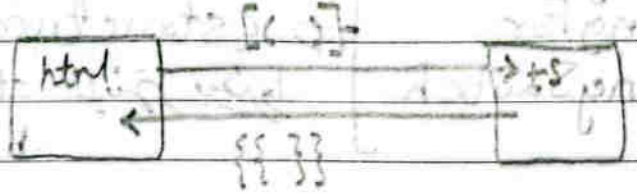
خطوات التعامل مع الـ two way data binding:

1- لازم نعمل import لـ module اسمه (FormsModule) في الـ root component ← app.module.ts وندرجه في الـ property 'import'.

```
import { FormsModule } from '@angular/forms';
@NgModule({
  imports: [
    FormsModule
  ]
})
```

2- نروح للـ input اللي عايزين ناخد منه الـ value، ونحط ngModel بين [(ngModel)] ← banana in a box والـ ngModel هيعمل bind لـ property في الـ ts.

```
<input [(ngModel)] = "userName" >
<p> {{userName}} </p>
```



③ directives ng—

* أي حاجة بتبدأ بـ ng يبقى تابع الـ directive .

Q) ايه هو الـ ngModel؟
الـ ngModel هو directive من الـ angular بتعمل
بيه two way data binding

Q) يعني ايه بتعمل two way data binding ؟
يعني بيجي مثلي input بتغير من الـ value بتاعته
فيروح يـ update من الـ ts بعد كده الـ ts يعرض من الـ view.

Q) ايه هو الـ directive ؟

الـ directive من الـ angular هي حاجة بتغير
يا إما من خلال الـ html / الـ style بتاع الـ html .
يا إما من الـ content

الـ directive بتغير يا إما styling يا إما content

directives :

1- ngClass , ngStyle (to change styling)
2- ngIf
3- ngFor
4- ngSwitch

(structural directives) ⟸ اسمهم structural لأنهم بيغيروا
من كود الـ html

الممسوحة ضوئيا بـ CamScanner

[ngClass] = "{ 'classname' : condition }"

"لما element ال class اهنا بنضيف ngClass"

الممسوحة ضوئيا بـ CamScanner

② *ngFor

ال ngfor بتطلع من ال element اللي مايزين ذكرته.

- *ngFor = " let x of y " → {{x}} (array)
- *ngFor = " let x of obj " → {{x.—}}
- *ngFor = " let x of y ; let i = index ; "
- *ngFor = " let x of y ; let isEven = even ; "

لي بيرجع ب true لو ترتيب ال index زوجي.

(ex)

```
<tr *ngFor = "let u of users ; let i = index ; let e = even ;">
    <td [ngClass] = "{'w-o': e}" > {{i}} </td>
    <td> {{ u.name }} </td>
    <td> {{ u.age }} </td>
</tr>
```

③ *ngIf

ال ngIf بتاخد boolean exp لو رجع ب true ال element هيظهر ولو ب false ال element هيتشال تماما ومش هيتعرض ولو فيه else هيروح لل <ng-template> بتاعته.

- *ngIf = " condition "
- *ngIf = " condition ; else hamboZo "

```
<ng-template #hamboZo> </ng-template>
```

بنكتب هنا ال html اللي مايزينه يظهر لو ال condition رجع ب false.

الممسوحة ضوئيا بـ CamScanner

④ [ngSwitch]

```
<div [ngSwitch]="username">
    <p *ngSwitchCase="'ahmed'">--</p>
    <p *ngSwitchCase="'ali'">--</p>
    <p *ngSwitchDefault></p>
</div>
```

```
<div *ngIf="isThing ; then b1 ; else b2"></div>

<ng-template #b1> -- </ng-template>
<ng-template #b2> -- </ng-template>
```

```
*ngFor="let x of X ; index as i ; odd as o ; even as e"
```

let i = index let o = odd let e = even

الممسوحة ضوئيا بـ CamScanner

revision

١- امتى نحتاج الـ angular ؟
لو الـ project متقسم لـ backend بالـ api و frontend.
هنحتاج الـ angular لو الـ backend عبارة عن apis

Frontend	Backend
Angular	APIs

٢- امتى نفصل الـ backend لوحدة كـ apis ؟
امتى نعمل الـ frontend ضمن الـ backend بدون استخدام الـ angular ؟
امتى نشتغل fullstalk ؟

* لو الـ project صغير ومش هشتغل بـ architecture pattern زى الـ MVC أو الـ MVT.

frontend	backend
no need for angular →	MVC MVT
we need Angular →	api

* الـ project اللى هيكون فيه web app و mobile app هيفضل نفصل الـ backend عن الـ frontend ونخلى التواصل بينهم بالـ API.

الممسوحة ضوئيا بـ CamScanner

٣- ال rest api :
١- هي ال api اللي بنقدر نتعامل معاها عن طريق ال url.
٢- كل api / كل endpoint ليه ال url الخاص بيه اللي
أقدر أعمله call من خلاله.
٣- ال api هتكون عبارة عن function في ال backend
وبيستقبل parameter ويرجع response بشكل معين.
ويفضل ال format بتاع ال response يكون JSON.
٤- ال rest api هو ال standard اللي بنمشي عليه
عشان أي api يكون متشابه مهما كانت اللغة
اللي اتكتبت بيه.
يعني التعامل مع api معمول بال php هو نفسه لو
مكتوب بال #C أو ال C، لازم ماشيين
على standard واحد وهو ال rest API.

٤- ال rest api هو ال api اللي ماشي على ال standard
بتاع ال rest / مثال rest architecture style.

* rest api is api that follow REST architecture style.
* rest is simplicity.
* rest rules: ① client-server model
② stateless
③ uniform interface
④ layered system
⑤ cacheable

٥- think components يعني كل حاجة من component

تقسيم ال web page لـ components وكل component ليه ملف ال html وال css وال js الخاص بيه، دة هيفدنا في reusability و scalability.

٦- بنضيف functionality و behavior لل class عن طريق class decorator.

٧- عشان نخلي ال class يتصرف كـ component من ال runtime بنديله ال builtin decorator اسمه @Component({}).

٨- angular interpolation:
→ interpolation refers to embedding expression into markup text.
(ex) <h1>{{ }}</h1>

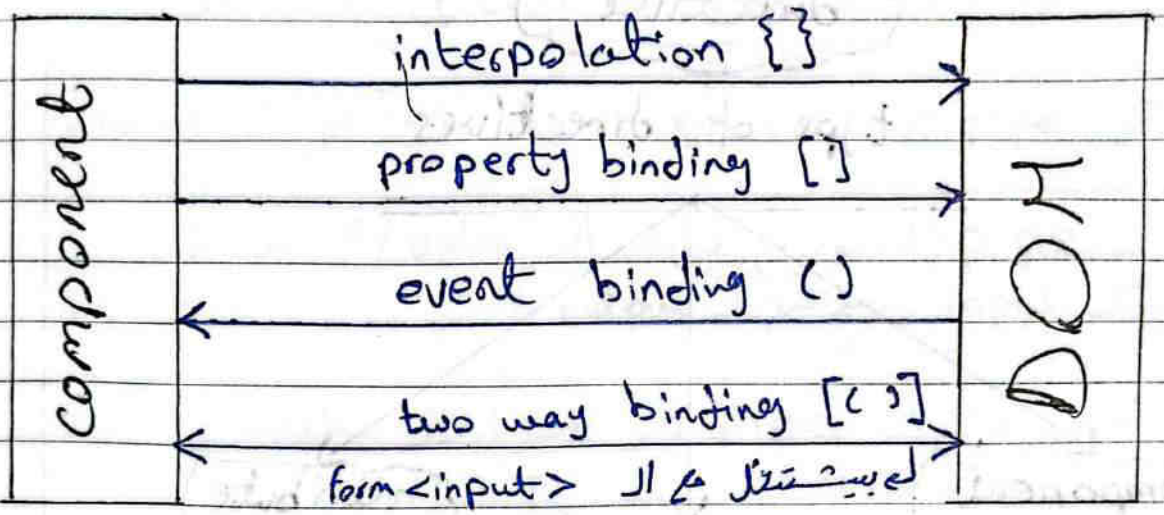
→ <h1> {{ user.username }}</h1>
<h1> {{ user?.username }}</h1>

→ ? is (safe navigation operator) used to check if the object is null or not.
→ if user object is not null then get the username.
else don't get anything.

الممسوحة ضوئيا بـ CamScanner

Data binding.

الـ data binding هو إن احنا نعرض data موجودة من ملف الـ ts الخاص بالـ component في ملف الـ html ـــ ـــ ـــ .



one way binding ← لو الـ value اللي موجودة في الـ component اتغيرت التغيير هيسمع في الـ dom / html.
(ex) الـ interpolation والـ property binding

او التغيير يحصل في الـ html ويسمع في الـ component.
(ex) event binding

two way binding ← التغيير في الـ component هيسمع في الـ dom والعكس.

الـ directive هو attribute أو tags جديدة
مش تبع الـ html لكن بنستخدمها من الـ html
ميزاتها : ① تغيير الـ design / الـ style بتاع element
او ② التحكم في اضافة content
او ③ اضافة content خاص بالـ component.

directive

type of directives

- component
  - (ex) selector tag of component
    - <app-home>
    - </app-home>
- structure directive
  - used to change the layout of the elements
  - (ex) ngIf, ngFor, ngSwitch
- attribute directive
  - used to change the behavior of an element
  - (ex) ngClass, ngStyle, ngModel

الممسوحة ضوئيا بـ CamScanner

ال pipe شبه ال directive لكن الفرق:
* ال directive بنفذه على element معين.
* ال pipe بشتغل على value.

ال angular بتوفر built-in pipes كتير.

ex {{ title | uppercase }}
pipe

كل pipe من ال angular بيأخذ حاجة فيه ال value.

ال pipe (الانبوب) بياخد ال value / ال data كـ input ويحولها لـ output.

الغرض من ال pipe هو تحويل ال value.

value → [pipe] → transformed value

app-routing.module.ts

③ Angular routing

www.example.com / home / ....
domain name — path

/home —display→ HomeComponent
/about → AboutComponent

* If the path is 'home', display the 'HomeComponent' in the place of <router-outlet></router-outlet>

ايه هي فكرة ال routing ؟

فكرة ال routing ان احنا نتنقل (navigate) بين ال components بناءاً على url path معين بنكتبه في ال (browser url)

كل component بنعمله / url path (route) خاصه بيه بنوصله من خلاله. لي مسار، طريق ي

ال <router-outlet> دي عبارة عن component directive وظيفته إنه بيستقبل وبيتحط مكانه ال selector الخاص بال component بناءاً على path معين، كل component بنعمله path خاص بيه.

ال routerLink="" دي عبارة عن built-in attribute بنستخدمه بدلك من ال attribute href="" . وظيفته إن احنا لما ندوس على ال link يروح يكتب الكلام اللي فيه كـ path في ال browser URL . من غير ما يحصل reloading لل page.

1-) {path: '', component: ■}
2-) routerLink = " "
3-) <router-outlet></router-outel>

إذا لو ال path فاضي، نعمل redirect ل component معين؟

فيه طريقتين:

① نعمل path فاضي و path مش فاضي لنفس ال component وده مش ال best practice لأن كده إحنا هنعمل load لنفس ال component مرتين.

{path: "", component: HomeComponent},
{path: "home", component: HomeComponent}

② نعمل path هيعمل redirect ل path تاني

{path: "", redirectTo: "home", pathMatch: 'full'},
{path: "home", component: HomeComponent},

ايه هو ال {pathMatch
هيستخدمه لو ال path اللي هنعمله redirectTo كان متكرر بس url patterns مختلفة.

path: "home",
path: "home/setting"
path: "home/category"

الممسوحة ضوئيا بـ CamScanner

① in navbar.component.html

```
<a [routerLink]= "home" > Home </a>
```

② in app-routing.module.ts

interface

```
const routes : Routes = [
  { path : 'home' , component : HomeComponent } ,
]
```

③ in app.component.html

بيتغير حسب الـ URL

```
<app-navbar></app-navbar>
<router-outlet></router-outlet>
<app-footer></app-footer>
```

دي هتتشال وتتحط مكانه الـ selector الخاص بالـ component تبع الـ path.

localhost:4200/home

↓

```
<router-outlet></router-outlet>
```

↓

```
<app-home></app-home>
```

الـ routing بيتعامل مع الـ browser URL كـ instruction بيغير الـ view بناءً على الـ instruction دة.

pathMatch: 'full' ← يعني ال path اللي هتعمله redirect
الترتيب هنا لازم يكون نفس اللي مكتوب في
مش مهم ال redirectTo: "".

pathMatch: 'prefix' ← يعني اعمل match مع أول path يقابلك
الترتيب هنا اول كلمة فيه هي نفس اللي مكتوب في
بيفرق ال redirectTo: ""

"إزاي لو ال user كتب path غلط، نظهرله صفحة معينة؟ عن طريق ال wild card ← (**)

(١) أولاً اعمل component ل create عادي جداً" سميه مثلاً: (notfound).

(٢) هنعمل ل component ده path اسمه (**) وده معناه all يعني اي حاجة مش بتعمل match باقي ال pathes اللي فوقيه.

wild card

{path: '**', component: notfoundComponent}

(٣) لازم لازم ال path: '**' يتكتب آخر path وده لأن ال route بيمشي عال pathes من فوق لتحت بالترتيب واحد واحد، ال route بيمسك اول path بيشوف هل بيعمل match ال url path ولا لأ، لو لأ هينزل عاللي بعده وهكذا لغاية ما يوصل ل (path: all).

لو كتبنا path: '**' في الأول، كده اي url هنكتبه هيعمل match علاه ☹

الممسوحة ضوئيا بـ CamScanner

ازاى ندى style معين للـ link معين لما ت active
الـ router بتاعته ؟

① in home.component.html

```
<a routerLink="home" routerLinkActive="test">click</a>
```

اسم الـ path — اسم الـ class

② in home.component.css

```
.test {

}
```

الـ routerLinkActive عشان تحط فيه أكتر من class

```
routerLinkActive="test bg-danger p-0"
```

ازاى نغير الـ <title> بتاع الـ <head> على حسب
الـ component ؟

```
{path: 'home', component: HomeComponent, title: 'home'},
```

عن طريق الـ property دة نقدر نغير الـ title
على حسب الـ component.

```
<title>—</title>
```


الممسوحة ضوئيا بـ CamScanner

nested routes

* nested route يعني نمثل path جوا path components جوا component.

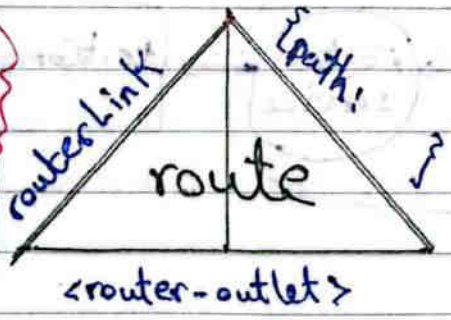
* هنستخدم property اسمه (children:).

(ex)
```
{path: 'settings', component: SettingComponent,
  children: [
    {path: 'profile', component: ProfileComponent},
  ]},
```

⇐ داخل الـ SettingComponent هنحط الـ <router-outlet>
وهنحط الـ routerLink اللي هيكتب الـ path
في الـ browser URL.

⇐ localhost:4200/settings/profile
nested route in path → (settings)

* دائماً وابداً الـ route هو عبارة عن مثلث متكامل.



اي مشكلة تقابلك في الـ route، ببحث أكيد عن ضلع من دول

كل ضلع من دول لازم يبقوا موجودين في أي route هتعمله
path

Router service (set parameter to url)

Route parameter

هنستخدم ال Router service

① {path: 'products/:id', component: testComponent}

② inject Router service.

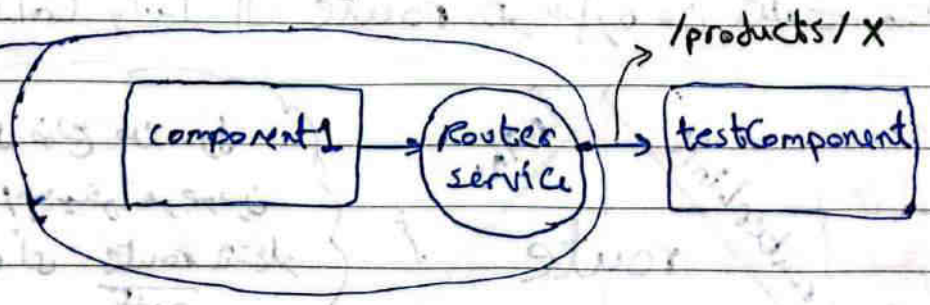
ال service دي بيسهلنا navigate ل router
ثاني من داخل ال typescript/ ال component.

```
constructor(private _router: Router){}
```

③ gotoTestcomponent(product){
    let x = product.id;
    this.router.navigate(['products', x]);
}

/products/x

④ <div (click)="gototestcomponent(x)"
    *ngFor="let x of products">
    <h1> {{x.id}} , {{x.name}} </h1>



الممسوحة ضوئيا بـ CamScanner

ActivatedRoute service

(get parameter from URL)

ازاي نقدر ال parameter اللي جاي مع ال route ؟
ازاي نقدر ال parameter اللي في ال URL ؟

عن طريق service اسمه ActivatedRoute.

```
① constructor(private ar:ActivatedRoute){}

② ngOnInit():void {
    let x = this.ar.snapshot.paramMap.get('id');
    let y = parseInt(x);
  }
  this.prop=parseInt(this.ar. — .get('id'));
```

بيرجع كـ string

جولنا اللي راجع من string لـ integer

ال snapshot مشكلته إنه بيتنفذ مرة واحدة، ومش بيحس باي تغيير جديد عالـ parameter.

بالتالي أحيانا لازم نراقب التغيير عالـ parameter، يعني نستخدم ال observable ونعمل subscribe.

```
③ ngOnInit():void {
    this.ar.paramMap.subscribe(
      (x:ParamMap) => {
        this.prop = parseInt(x.get('id'));
      })
  }
```

أي تغيير اديلي notification.

الممسوحة ضوئيا بـ CamScanner

Session (4)

1. components life cycle
2. components interaction
3. angular services
4. dependency injection

الممسوحة ضوئيا بـ CamScanner

① component life cycle

* ال component بيمر لية life cycle بيتم إدارته بواسطة ال angular.

بداية من ال lifecycle بتاعة ال component من اول مايفتح لغاية مايقفل؟
بداية دورة حياة ال component من اول ما نفتحه؟
* اية هو ال sequence اللي بيتم من اول ما ن access ال component؟ اول ما ال component يشتغل؟

① كل component بيمر لية lifecycle / لية مراحل بيمر بها
② بيتم ادارة المراحل دية من طريق ال angular:

① angular create the component.
② angular renders it.
③ angular create & renders its child component.
④ angular check the component when its data-bound properties change.
⑤ angular destroy the component before removing it from the DOM

③ ال angular بيوفرلنا lifecycle hooks وهول عبارة عن special methods عن طريقهم نقدر نشغل run لكود معين عند cycle معينة / عند مرحلة معينة من ال lifecycle الخاصة بال component.

hooks يعني خطاف

- اول حاجة بتتفتح من ال component هو ال ts بتاع ال component او ال class بتاع ال component.

- ايه هو ترتيب تنفيذ ال methods في ال component class؟

① اول method بيتنفذ من ال component هو ال constructor.
بعد كده ال component بيكون فيه مجموعة functions جاهزة، وكل function منهم بيتم تنفيذهم عند مرحلة معينة من مراحل دورة حياة ال component.

② ngOnChanges() دي lifecycle hook بيتم تنفيذها بعد ال constructor وبيتعمل call في كل مرة بيحصل فيها تغيير في اي input parameter جاية لل component
(يعني اي تغيير في ال property اللي ليها @Input decorator)
(او اي input data جاية عن طريق ال @Input decorator)

③ ngOnInit() دي function بتتنفذ / بيتعمل call بعد ال ngOnChanges() وبيتعمل call مرة واحدة فقط لما ال component يبقى initialized / لما يتفتح.

④ ngDoCheck() دي function بيتعمل call "مباشرة" بعد ال ngOnChanges() وال ngOnInit() وبيتعمل call عند اي change detection

⑤ ngOnDestroy() دي function بيتعمل call "فوراً" لما ال componet بيتعمل destroy (مثال) لما ال uses يتحرك من component لـ component تاني عن طريق ال routing. يعني لما ال component يقفل.

الممسوحة ضوئيا بـ CamScanner

Q) ايه هو الفرق بين الـ constructor() والـ ngOnInit()
طالما الاثنين بيتعملوا call لما الـ component يفتح
/ لما الـ component يتعمل initialized ؟

الـ constructor() دايماً بيتعمل call أول واحد.
الـ ngOnInit() بيتعمل call بعد الـ ngOnChange()
لما يخلص.
نستخدم ايه؟ ايه هو الـ best practice ؟
الـ constructor نستخدمه في حاجتين فقط،
① في الـ variable initialization،
② في الـ dependency injection
اي حاجة غير الاثنين دول عايزهم يتنفذوا اول لما الـ component
يفتح ـ يبقى ننفذها في الـ ngOnInit().

Q) ايه هي اهمية الـ lifecycle hooks ؟

اهميتها ان احنا نقدر نعمل run لكود معين عند مرحلة
معينة من مراحل تشغيل الـ component.

(مثال) نفذ الكود دة لما الـ component يفتح.
" " " " " " يقفل.
" " " لما يحصل تغيير في قيمة @Input property.

اشهر مثال زي فيه الـ api هو الـ ngOnInit().

ازاي نستخدم الـ lifecycle hooks ؟

← كل function من الـ lifecycle hooks ليه interface خاص بيه، يعني كل function منهم موجود في interface معين.
بالتالي عشان نقدر نستخدم أي function منهم، لازم نعمل implement للـ interface بتاعته عن طريق الـ class بتاع الـ component.

ليه الـ ngOnChange() بيجي fire قبل الـ ngOnInit ؟

لأن دي الـ sequence المنطقي والطبيعي اللي angular مرتباه، أنا لازم أعرف إيه التغيير اللي حصل في الـ data بتاعة الـ component قبل ما نعرض الـ component زفت.

إيه هو لازمة الـ SimpleChanges ؟

الـ ngOnChanges() بيحتوي على parameter اسمه changes من النوع SimpleChanges، عارضة إنه عبارة عن object بيحتوي على الـ input data اللي حصل لها change.

```
ngOnChanges(changes:SimpleChanges){
    console.log(changes);
}
```


الممسوحة ضوئيا بـ CamScanner

② Components Interaction

* الـ components interaction هو ان احنا ننقل الـ data sharing بين الـ components .

* ممكن نقسم الـ components interaction لنوعين

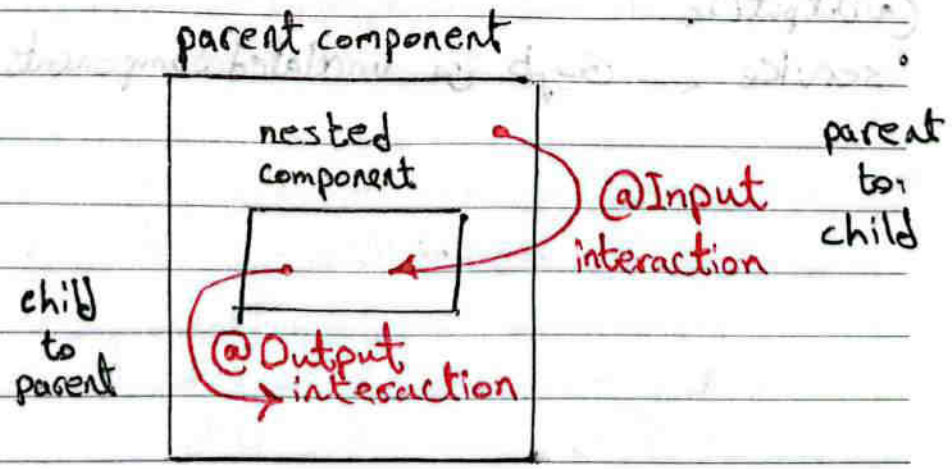
① nested component interaction

الـ nested component هو الـ child component
هو component داخل component تاني .

* لو الـ nested componedt عايز ياخد داتا / يقرا داتا من الـ parent component ، فيه اسمه @Input

* لو العكس يعني الـ nested عايز يـ share الداتا مع الـ parent فيه اسمه @Output

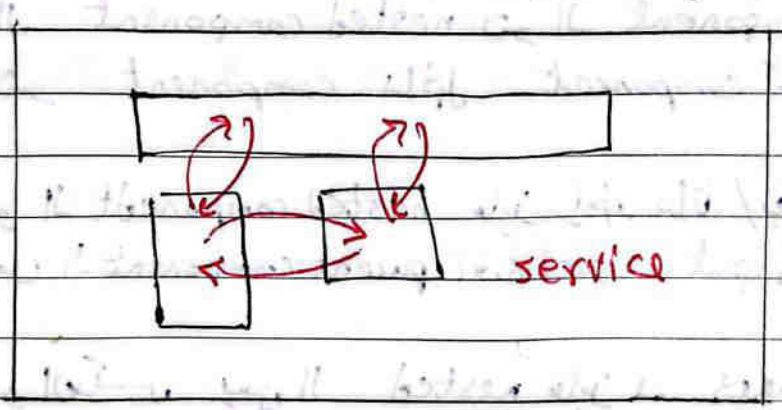
* الـ @Input و الـ @Output بنسميهم الـ sharing data بين الـ nested component .



unrelated components interaction ②

وده عبارة عن ان احنا نعمل sharing للداتا بين components ملوش علاقة ببعض ، يعني مش nested.

هنا بنعمل sharing لل data عن طريق ال service



الخلاصة

بنعمل sharing لل data ما بين

← ال nested components عن طريق ← @Input() @Output()

← ال unrelated components عن طريق ← service

الممسوحة ضوئيا بـ CamScanner

① share data from parent to child :

at parent.component.ts

```
export class ParentComponent {
    selectedID:number = 0;
}
```

اى ربط بين كود ال html وكود ال ts يوجد في طول (binding)

at parent.component.html

```
<select ---- [(ngModel)] = "selectedID" >
=
</select>
② <app-child [idfromparent] = "selectedID" ></app-child>
```

at child.component.ts

```
export class ChildComponent {
① @Input() idfromparent:number = 0;
}
```

الخطوات اية؟

١- أكتب property decorator اسمه @Input() قبل ال property اللى هيستقبل data من ال parent.

٢- اعمل binding بين ملف ال ts بتاع ال parent وال child من طريق ال selector الخاص بال child component

```
<app-child [@Input() اللى جاية = "اسم ال property"] >
</app-child>
```

اسم ال property: اللى جاية ال child
اسم ال property: اللى جاية للـ child

الممسوحة ضوئيا بـ CamScanner

child.ts ← <app-child [child]="parent"> <\/app-child> → parent.ts

الخلاصة: عشان انقل data زي ال api مثلاً، من ال parent
ال child، اولاً بجهز ال data مكان في ال child
واديله @Input decorator،
بعدكده/ثم وانا بنادي ال child جوا ال parent
١- بكتب اسم ال property اللي هحط الداتا فيها،
٢- واكتب الداتا اللي عايز اعطها.

ليه نعمل كده؟ ليه ننقل ال data من ال parent
ال child؟ اذكر مثال على استخدامه؟

بنعمل كده عشان منكررش ال http request،
يعني مثلاً لو استقبلنا data من api في ال parent
واحتجنا ال data دي تاني في ال child، ممكن
ننقل من ال parent ال child بدل ما نعمل request
تاني لل api بتاع ال child.
بالتالي هنحسن ال performance.

ننقل ال data من ال parent لل child
هنا خطوتين.
① اجهز مكان لاستقبال ال data عن طريق @Input()
② ابعت ال data من مكان في ال parent مكان في ال child
من خلال ال component selector بتاع ال child.

الممسوحة ضوئيا بـ CamScanner

① @Input() child: any = 1;

② <app-child [child] = " p " ></app-child>

(ملحوظة) أي تغيير هيحصل في ال @Input() ، هيحصل call لل ngOnChange().

ال ngOnChange() بيبين تنبيه إن ال value في ال input parameter اتغيرت.

(note)

* عندنا 3 cases عشان نعمل sharing لل data.

① أعمل share لل data من الأب للابن:
② من الابن للأب
③ أعمل file بيـ share على كل ال components وبتاعة service.

* مفيش share لل data بين siblings ،

هو share يا إما بين الأب للابن ،
يا إما بين الابن للأب ،
يا إما file بيـ share لكلهم.

الممسوحة ضوئيا بـ CamScanner

(2) share/send data from child component to parent component.

1- هننقل data من ال child component لل parent component، ده معناه ان: ال data موجوده في ال child component.

∴ at child.component.ts

```
export class ChildComponent {
① @Output() childEvent = new EventEmitter();
```

2- عند ارسال المعلومات من ال child لل parent هنستخدم ال EventEmitter وال @Output()

2- بعد كده هننشاء function بأي أسم عادي، وهذه ال function هتستقبل من ملف ال html لأن ال event هيحصل من ال html من button مثلاً، ومنها المعلومات هتتبعت من ال child لل parent.

emit يعني هيبعت لبره

```
② eventChild() {
    this.childEvent.emit("the data");
}
```

هنا هنحط المعلومات اللي عايزين نبعتها لل parent.

```
}
```


الممسوحة ضوئيا بـ CamScanner

٤- هنستخدم ال function اللي هتبعت البيانات من ال child لل parent في ملف ال html الخاص بال child.

at child.component.html

```
③ <button (click) = "eventChild()" >
    go to parent
  </button>
```

لما ال user يضغط عالزر، ال function هتشغل ال call، جوا ال function هيحصل إبعاث / emit ل data معينة.

٥- هنروح لل parent ونجهز property هتستقبل قيمة المعلومات اللي جاية من ال child.

at parent.component.ts

```
export ParentComponent {
④ message: string = " " ;

}
```

٦- هنروح لملف ال html الخاص بال parent ونعمل select لل child component،

من خلال ال child component selector هنبعت الداتا / هنعمل binding بين ال output property @ وال property اللي هتستقبل قيمة ال data:

```
⑤ <app-child (childEvent) = "message = $event" >
  <app-child>
  <p>{{message}}
```

③ angular service

الـ service هو concept من الـ angular
عبارة عن file ts بيحتوي على data بنقدر نعمل
share بين أكثر من component فلو شي علاقة
ببعض.

ايه اهمية الـ service؟ ايه هو الـ service؟

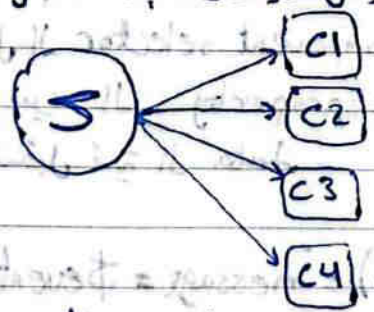
الـ service هو عبارة عن file Ts الـ business logic
والـ data اللي فيه بتكون shared بين الـ components.

ازاي نعمل service بالـ CLI؟

>> ng g s userService ←

امتى نستخدم الـ service؟

لو فيه عندي data أو functions عايز استخدمها في
أكثر من component، يعني بحقق مبدأ الـ reusability.



هي بخزن الـ data، وبخزن
الـ function هستخدمها
في أكثر من component.

الـ service في الآخر هو class فيه properties و
methods بنقدر نستخدمها في أي component.

الممسوحة ضوئيا بـ CamScanner

ايه هو ال service decorator ؟
ايه هو ال @Injectable() decorator ؟
ايه هو فايدته ؟

ال @Injectable decorator بيعرف ال class
كـ service من ال angular وبيسمح لل angular
يـ inject ال service دية من اى component بالتالى
ال component يقدر يـ access ال service.

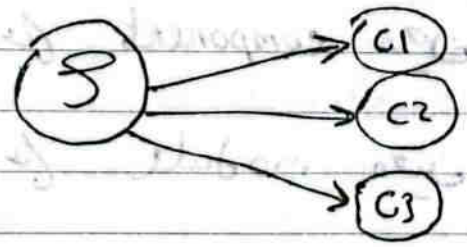
ايه هى الاماكن اللى نقدر نعمل فيها provide لل service ؟

---

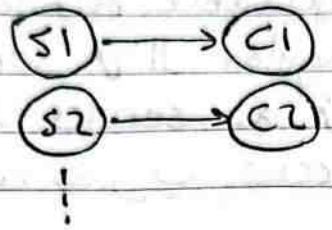
* اولاً ان احنا نعمل provide لل service يعنى نحدد
مين يقدر يـ access ال service / ونحدد ال scope
بتاع ال service.

* عندنا ٣ اماكن / 3 decorators نقدر نعمل فيهم
providing لل service يعنى نحدد ال scope
بتاع ال service.

① بداخل ال @Injectable() عن طريق {providedIn: 'root'}
وده بخلى ال service accessable من كل ال components
وبتبقى نسخة واحدة مشتركة ما بين ال components ، يعنى اى
تغيير من ال service هيسمع من كل ال component.

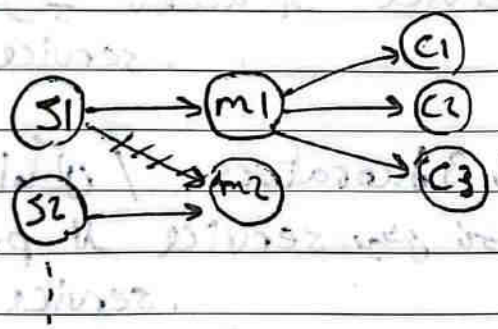


② بداخل الـ @Component عن طريق property اسمه providers: [اسم الـ service] ، وده هيخلي الـ component ليه النسخة الخاصة بيه من الـ service ، يعني أي تغيير في الـ service مش هيسمع في باقي الـ components .



كل component ليه النسخة بتاعته .

③ بداخل الـ @NgModule عن طريق property اسمه providers: [اسم الـ service] . وده هيخلي الـ service مشترك لل components اللي تبع الـ module ده فقط .



كل module ليه النسخة بتاعته

الخلاصة : عندنا 3 scope لل service .

① على الـ root / بال project كله .

② على component معين / على مستوى الـ component .

③ على module معين / على مستوى الـ module .

الممسوحة ضوئيا بـ CamScanner

(ex)

① >> ng g s users ↵

② at users.service.ts

```
@Injectable({ providedIn: 'root' })
export class UsersService {
    users: object[] = [{ id: 1, name: 'Ahmed', age: 30 }];
}
```

③ at home.component.ts

```
export class HomeComponent {
    homeUsers: object[] = [];
    constructor() {
        let x = new UsersService();
        this.homeUsers = x.users;
    }
}
```

اتعود دايماً اي service عايز تستخدمها ، اعمل منها نسخة بالـ dependency injection بدل اعمل نسخة من غير ما تعمل call للـ constructor عن طريق استخدام الـ dependency injection .

```
constructor(private _s1: S1) {}
constructor(private x: UsersService){}
this.homeUsers = this.x.users;
```


الممسوحة ضوئيا بـ CamScanner

④ dependency injection

ما هو ال design patterns ؟

اولاً ، pattern يعني نمط ، نمط يعني طريقة أو اسلوب
design يعني تصميم .
design pattern يعني انماط تصميم أو طرق مختلفة للتصميم ،
والمقصود بالتصميم هو تصميم ال architecture الخاص
بال project ، ازاي هنحط ال folders ؟ ازاي ال classes
وال component هيتكلموا مع بعض ؟

ازاي ال design patterns اتعملت ؟

ال design pattern هي حلول لمشاكل معينة ظهرت
لل developers اللي قبلينا .

ال developers اللي قبلينا واجهوا مشاكل معينة في
ال software ، فعملوا لكل مشكلة pattern
نمشي عليه عشان نتجنب المشاكل دي .

كل ال design pattern هي حلول لمشاكل معينة .

* من ضمن ال design pattern اللي ال angular
بتستخدمها هي ال dependency injection .

الممسوحة ضوئيا بـ CamScanner

• اعمل نسخة من ال service
من غير ما تعرب مال constructor

dependency injection

ايه هو ال dependency injection pattern ؟

دة عبارة عن design pattern بيحل مشكلة الاعتمادية بين ال classes ، عن طريق انه يخلي ال class يبقى مستقل عن ال classes اللي بيعتمد عليها .

هو programming technique بيحل مشكلة الاعتمادية بين ال objects وال classes .

ليه بنستخدمه في ال angular ؟ ايه المشكلة ؟ ايه الحل ؟

المشكلة ان احنا لما بنعمل inject ل service على أكثر من component عن طريق ان احنا بنعمل object من ال service في كل component ، بالتالي كل ال components بقت معتمدة على ال service ، يعني ان اي تغيير في ال constructor هيسبب مشاكل في كل ال components اللي بتعتمد عليه .

الحل ان احنا نعمل inject ال dependency اللي ال component معتمد عليها / محتاجها ، من غير ما نعمل calling ل constructor بشكل مباشر . ازاي ؟

في ال constructor بتاع ال component هنعمل object من ال service من غير ما نعمل call ل constructor بتاع ال service .

```
constructor( obj : UserService ){
    this.compo1 = obj.users;
}
```


الممسوحة ضوئيا بـ CamScanner

(ex)

```
export class HomeComponent {
    let X:object[] = [];

① constructor ( ){
        let _UserService = new UserService ( );
        this.X = _UserService.users;
    }

② using dependency injection
يعني منعمل حقن للحاجة اللي ال component
محتاجها من برة ، يعني من غير منعمل الـ
ال constructor بتاعه بشكل مباشر

    constructor (_UserService : UserService) {
        this.X = _UserService.users
    }
}
```

ملحوظة: ال angular بتعمل implementation
ال dependency injection ، يعني لو
استخدمت ال design pattern
خارج ال angular ، مش هيشتغل

ال dependency injection هو اسمه مجلص ومرتب
كلن هو اصلا شغلة وفكرة تافهة
مخافش
قوى قلبك

الممسوحة ضوئيا بـ CamScanner

* يفضل نعمل نسخة من ال service عن طريق ال dependency injection.

* يفضل النسخة بتاعة ال service تكون private لأننا مش هنستخدمها خارج ال class.

```
constructor( private_s1 : S1,
             private_s2 : S2 ) { }
```

ملحوظة مهمة جداً:

أي access modifier بنحطه لأي parameter في ال constructor دة معناه إن ال parameter دة هيبقى ليه variable متعرف خارج ال constructor وهيتعامل assign بقيمة ال parameter داخل ال constructor وكل دة behind the scene في ال TS.

(ex)

```
public x:string;
constructor(x:string){
  this.x = x;
}
```

⇒

```
constructor(public x:string){ }
```

بس كدة ☺، دي كدة إختصر عليا إن انا أعرف ال x واعمل initialize بإيدي.

تذكر: لما تستخدم أي service في ال angular اعمل منه نسخة بال dependency injection

⇐ ال service وال component لها في الآخر عبارة عن class.

APIs

Session (5)

1- Observables.

2- HttpClient.

3- template reference variable

4- pipes

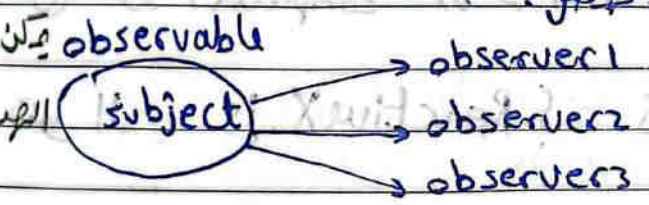
الممسوحة ضوئيا بـ CamScanner

① Observables

* ايه هو الـ observable ؟

الـ observable هو بديل الـ promise بنستخدمه
عشان نهندل الـ asynchronous code ، يعني نحول
الـ asynch code لـ synch code .

الـ observable بيتبع الـ observer design pattern
وهو باختصار عبارة عن علاقة one to many بين الـ objects
يعني فيه object واحد بيعتمد عليه عدة objects ، بالتالي
الـ object لما حالته تتغير كل الـ objects اللي معتمدة عليه
هتتحس بالتغيير اللي حصل .



يعني باختصار الـ object هيراقب الـ object اللي معتمد عليه .

امثلة على الـ asynchronous code :
api call
ajax call
fetch call

الخلاصة الـ observable هنستخدمه لمراقبة الـ data
اللي بتيجي من asynchronous code زي الـ http client ،
عشان نستخدم الـ observable لازم نشترك
فيه يعني نعمله subscribe .

* لما بتتكلم عن observable يبقى بتتكلم عن:
① حاجة لمراقبة ال data
② real time تابع ل event زي ال http request

* لاستخدام ال observable لازم نعمل subscribe/إشتراك.

اولا بننشاء observable لل data اللي عايزين نراقبها.
ثانيا لما نحتاج نستخدم ال observable بنشترك فيه.

* ال subscribe ليه functions زي:
① next() function لعرض المعلومات / ال data
② error() لما يكون فيه error
③ complete() دة شبه ال default في ال switch.

أمتى هو ال ReactiveX.js ؟ RxJS.

دي عبارة عن javascript library بتتعامل مع ال api باستخدام ال observable، بتتعامل مع ال asynchronous code باستخدام ال observable، ال library دي بتتبع ال observer design pattern.

ال observable هو مراقبة ال data في زمن.

٧ ايه هو الفرق بين ال promise وال observable؟

observable	promise
ال observable زي ال event يقدر يعمل call مع اي تغيير يعني ال function في ال subscribe() بتاعة ال observable، ممكن تتنفذ اكثر من مرة.	ال promise بتنفذ مرة واحدة فقط، بتنفذ ال callback function في ال then() مرة واحدة فقط.
observable.subscribe( )	promise.then( )
ال observable نقدر نوقفه عن طريق ال unsubscribe()	ال promise مش نقدر نوقفه لما يتنفذ.
ال observable بيوفر operators نقدر نستخدمه مع ال data اللي راجعة.	ال promise مبيوفرش اي operators
ال observable هو lazy، يعني مش هيتعمل execute الا لما نعمل subscribe باستخدام ال subscribe() method	not lazy يعني بيتعمل execute في طول لما بيتعمل creation

اهم ميزة في ال observable انه:
① cancellable يعني نقدر نوقفه
② lazy يعني مبيشتغلش الا لما نعمل subscribe

الممسوحة ضوئيا بـ CamScanner

② Http Client

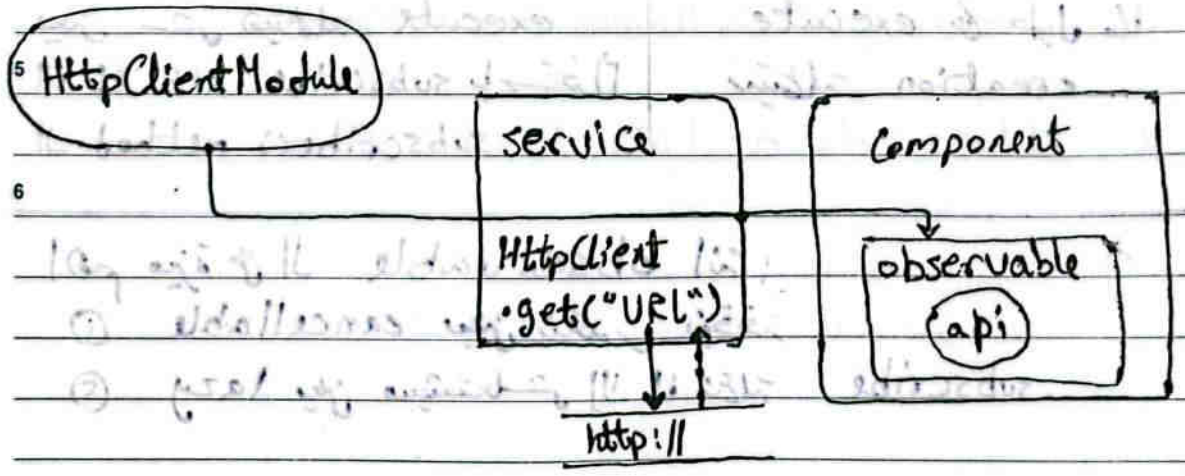
١- الـ angular بتوفرلنا module للتعامل مع الـ APIs
اسمه (HttpClientModule) عشان نستخدمه
لازم نعمله import في الـ app.module.

الـ HttpClientModule دي الـ module المسؤول
عن التعامل مع الـ api في الـ project.

٢- يفضل نعمل service نستلم فيه الـ api، بحيث
نقدر نستخدمه مع أكثر من component ونعمل
http request واحد فقط.

٣- لي module في الـ angular بيوفر لنا class
اقدر استخدمه (الـ HttpClientModule) لنا class اسمه (HttpClient)

٤- الـ HttpClient class بيرجع الـ api من نوع Observable
والـ observable هو بديل للـ promise للتعامل مع الـ
asynchronous code.



الممسوحة ضوئيا بـ CamScanner

① app.module.ts

```
import {HttpClientModule} from "@angular/common/http";
≡
imports: [—, —, HttpClientModule]
```

② test.service.ts

```
import {HttpClient} from "@angular/common/http";
import {Observable} from "rxjs";
≡
export class testService {
    constructor (private _httpClient : HttpClient ) {}
    getAPI () : Observable <any> {
        return this._httpClient.get(' ');
}. }
```

③ home.component.ts

```
import {testService} from "../test.service"
export class HomeComponent implements —, —, — {
    constructor (private _test : testService) { }
    ngOnInit () { this.getAllData () }
    getAllData () {
        this._test.getAPI().subscribe(
            (res) => console.log (res),
            (err) => console.log (err),
            () => console.log ("completed")
    } }
```

الـ data بترجع لنا

```
getAllData() {
    this._test.getAPI().subscribe(
        (res) => {
            this.x = res.result;
        },
        (e) => { },
        () => { }
    )
}
```

callback functions

```
ngOnDestroy() {
    this._test.getAPI().unsubscribe()
}
```

ممكن اختصاراً ، نخزن ال observable اللي بيحتوي على ال api في property / ~~variable~~.

(ex)
```
x : any;
x = this._test.getAPI();
this.x.subscribe((r) => {}, () => {}, (e) => {});
this.x.unsubscribe();
```

property بدل variable

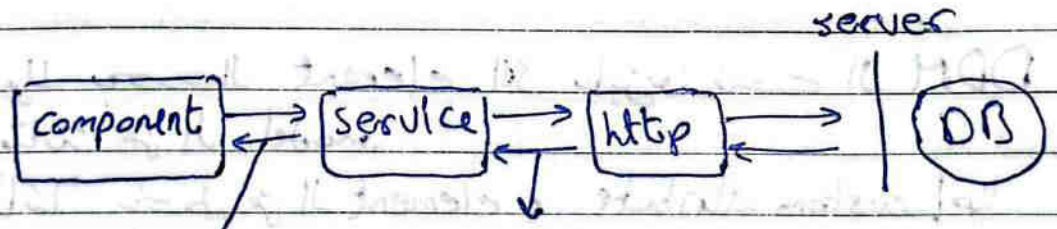
① اول خطوة اعمل import لل HttpClientModule من ال module.
② ثاني خطوة اعمل import لل HttpClient من ال service
~ ~ ~ ~ لل Observable
اعمل method بيستقبل ال observable اللي
بيحمل ال api
③ ثالث خطوة استخدم ال service في ال component.

الممسوحة ضوئيا بـ CamScanner

الداتا اللى راجعة بتكون من نوع observable

Keyword to remember

1- HttpClientModule
2- HttpClient
3- Observable
4- .get('URL')
5- .subscribe()
6- .unsubscribe()



response the data as observable

to use the observable that is returned from the service, we've to (subscribe)

الممسوحة ضوئيا بـ CamScanner

(3) template reference variable

أية هو ال template reference variable ؟

دة عبارة عن feature من ال angular
بيسهلنا نبعت ال DOM element ل element
معين متبعتة من ال html لل model (ts).

بالتالي نقدر نستخدم ال properties وال methods
الخاصة بال DOM في ملف ال ts.

ال DOM هو تمثيل ال html element كـ object في js.

ازاي نستخدم ال template reference variable ؟

اولا بنحدد ال element اللي عايزين نبعت ال DOM
بتاعته في ال model.
ثانيا بنحط في ال element ، custom attribute إحنا
اللي بنسميه وبنكتب قبليه رمز ال (#) ،

```
<input type="text" #anyname >
```

template ref_variable هو ال
reference variable هو
ال template اللي هو خية.

الممسوحة ضوئيا بـ CamScanner

(EX)

```
<input type="text" #txtInput>

<button (click)="m1(txtInput)">click here
</button>
```

in ts file

```
msg = " ";
m1(input){
    console.log(input);
    console.log(input.value);
    this.msg = input.value;
}
```

```
<p #hampozo></p>
```

hampozo هو variable يمثل ال dom الخاص بال <p>

* الفايدة الاساسية منه ان إحنا نبعت value من ال view لل model.

④ pipes

* دية feature من ال angular بتسمحلنا نعمل transformation ، ال data ، يعني نطلع ال data بشكل معين ، (نحول ال data)

* عندنا نوعين من ال pipes :

① pipes تبع ال angular (defined pipes)
② pipes إنتة بتعملة (custom pipes)

* ال pipe من الأمر دو عبارة عن function بيستقبل ال value ، ويحلل return بعد تعديل معين .

```
{{ x | lowercase }}
{{ x | uppercase }}
{{ x | titlecase }}

{{ x | slice : 3 }}
{{ x | slice : 4 : 7 }}

{{ 3.657 | number : '1.2-3' }} → 3.657
{{ 3.657 | number : '4.4-5' }} → 0,003.6570
{{ 3.657 | number : '4.1-2' }} → 0003.65
```


الممسوحة ضوئيا بـ CamScanner

ازاى نعمل custom pipe

1- ng g pipe square لـ

2- square.pipe.ts

اى pipe بنعمله create بيحتوى على function اسمه transfor() اللى بننفذ فيه الـ logic بتاع الـ pipe.

```
@Pipe({
  name: 'square'
})
class
export class SquarePipe implements PipeTransform {

    transform (value: number): unknown {
        return value * value;
    }
}
```

3- in html

```
{{ 10 | square }} → 100
```

(ملحوظة) اسم الـ custom pipe اللى انت بتعمله لازم يكون unique ومش متكرر من defined pipes الخاصين بالـ angular) وإلا مش هيشتغل.

Q) ازاى ال pipe بياخد parameter ؟

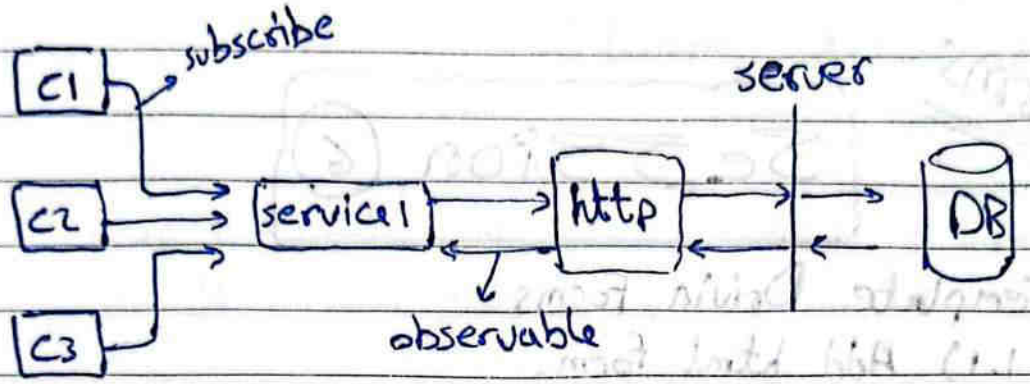
```
transform (value:number, v2:number):number {
    return value * v2;
}

{{ 10 | square : 5 }} → 50
```

parameter ال الاول

parameter ال التانى

الممسوحة ضوئيا بـ CamScanner



١- ال component بتكلم ال service
٢- ال service بتبعت http request
٣- ال http request هيكلم ال server
٤- ال server بيشوف ال DB
٥- ال server يرد بال http
٦- ال http هيرجع بال data على شكل observable object.

ال observable عشان نستخدمه لازم نعمله subscribe.

الممسوحة ضوئيا بـ CamScanner

forms

Session (6)

1- Template Drivin Forms
- 1.1) Add html form.
- 1.2) Bind data.
- 1.3) track state & validaty.
- 1.4) provide visual feedback.
- 1.5) display error messages
- 1.6) posting ~~data~~ to the server.

2- Reactive Forms
- 2.1) add html form.
- 2.2) create form model.
- 2.3) manage form control values.
- 2.4) FormBuilder service
- 2.5) validation → simple
  - → custom
  - → cross-field
  - → dynamic
- 2.6) add dynamic form controls.
- 2.7) submit form data.

الممسوحة ضوئيا بـ CamScanner

ايه هي اهمية الـ form ؟
امتى نحتاج form ؟

الـ form هي طريقة بتسهلنا ناخد data من الـ user .
هنستخدم الـ form لو عايزين نستقبل data من الـ user .

ازاي نتعامل مع الـ forms في الـ Angular ؟

الـ angular عندها طريقتين للتعامل مع الـ forms :
① Template Driven Forms
② Model Drivin Forms (Reactive Forms)

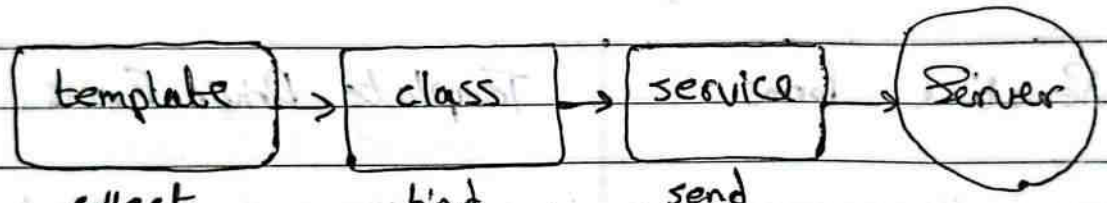
Reactive forms	Template Drivin Forms
→ heavy on the component class الـ logic اغلبه في الـ ts .	heavy on the component html يعني الـ logic بتاعنا اغلبه في الـ html .
١- الـ logic موجود اغلبه في الـ ts .	١- الكود موجود في الـ html
٢- بنستخدمها لما نتعامل مع الـ complex senarios ,	٢- easy to use
٣- نقدر نعمل custom validation .	٣- بيستخدم الـ two way binding [ngModel]
٤-	٤- نقدر نتتبع الـ state والـ validation .
	٥- بنستخدمه مع الـ simple forms .
	٦- بنستخدمه لو الـ unit test مش مهمة .

Angular Forms

① > the component template (html) contains HTML to collect the user data

② > the component class handling data binding, which means it takes the data from the view.

③ > the collected data will be sent from the component class to the server via service.



Q) ايه الفرق الجوهري بين ال Template Drivin Forms & Reactive Forms

* في ال template drivin forms الحل بيكون على ال html .
* في ال Model drivin forms . الحل بيكون في ال TS ..

بنستخدم ال template drivin forms مع ال Forms البسيطة .. زي ال login وال register ، وعلشان يكون ال units testing بسيط ،

الممسوحة ضوئيا بـ CamScanner

① اعمل form ،
② خد منه data ،
③ ابعت الـ data للسيرفر .

① adding the html form :

(ex)

```
<input type="text" >
<input type="email" >
<input type="tel" >
```

② binding the data [(ngModel)] , ngForm

طالما بنستخدم طريقة الـ template drivin forms يبقى لازم نعمل import الـ (FormsModule) بالتالي نقدر نعمل 2-way binding.

template drivin forms $\xrightarrow[\text{use}]{\text{need}}$ 2-way binding [(ngModel)]

اول خطوة | في اي وقت بنستخدم الـ form ، الـ angular بيـ attach directive جديد اسمه (ngForm) بيحتوي على معلومات زيادة عن الـ form زي مثلاً الـ value بتاعة كل input وهل هي valid ولا لا ،

- عشان نوصل لمعلومات عن اي html element هنستخدم الـ template reference variable.

```
<form #userform = "ngForm" >
=
</form>
```

الـ variable دة هتبقى جواه ،
معلومات عن الـ form element
+ معلومات عن الـ inputs اللي جواه .

الممسوحة ضوئيا بـ CamScanner

ثاني خطوة

عشان الـ ngForm يـ track الـ value بتاعة inputs معين ، لازم الـ input ده يتوفر فيه شرطين : ① يحتوي على name ، ② يحتوي على ngModel .

```
<form #userForm = "ngForm">
    <input type="text" name = "Name" ngModel >
    <input type="email" name = "Email" ngModel >
    <input type="tel" name = "phone" ngModel >
</form>
```

عشان الـ ngForm يتمكن ال track لازم يكون ليه name

عشان يحصل binding

ثالث خطوة

عشان نعمل group of inputs ، بحيث نعرضهم في object لوحدهم في الـ ngForm ، هنستخدم ngModelGroup = " " .

```
<form #userForm = "ngForm">
    <input type = "text" name = "Name" ngModel>
    <div ngModelGroup = "address">
        <input type = "text" name = "street" ngModel>
        <input type = "text" name = "state" ngModel>
    </div>
</form>
```

User
- Name = "Ali"
- address
  - street = " "
  - state = " "

الممسوحة ضوئيا بـ CamScanner

رابع خطوة: هنعمل class نجمع فيه كل الـ fields اللي في الـ form، بحيث نعمل منه object ونعمل فيه binding.

>> ng g class User لـ

```
export class User {
  ~~public~~
  constructor ( public name:string,
                public email:string,
                public phone:string,
                public address:string,
                ) {}
}
```

كل field منهم عبارة عن مكان هتخزن فيه الـ value التابعة لـ input معين

خامس خطوة: نعرّف object من الـ class في الـ component اللي شايل الـ form.

```
userModel = new User('', '', '', ...);
```

سادساً: اعمل bind بين الـ inputs والـ object باستخدام الـ ngModel.

```
<input type="text" name="email" [ngModel]="userModel.email">
```


الممسوحة ضوئيا بـ CamScanner

③ Tracking state & validity

بيرتبط input / form controll كل ، by default
بيه مجموعة directives / classes بناءا على الـ state الـ هو فيه .

state	if true	if not
control has been visited.	ng-touched	ng-untouched
control value has been changed	ng-dirty	ng-pristine
control value is valid.	ng-valid	ng-invalid

لو الـ control متلمسش ، مش محولة focus هياخد ng-untouched .
لو الـ control الـ value بتاعته فاضية متغيرتش ← ng-pristine
لو الـ control مفهوش validation هياخد ← ng-valid

دول عبارة عن classes بيتضافوا للـ form control ، وكل class منهم بيتضاف عند حالة معينة للـ input

(ex)

```
<input type="text" #name class="test" />
<p> {{name.class}} <p>
```

→ "test ng-untouched ng-pristine ng-valid"

الممسوحة ضوئيا بـ CamScanner

ايه هو البديل لل classes دية؟
لو مش عايز استخدم ال classes دية ايه البديل؟

عن طريق ال ngModel.
ف ال ngModel بتوفر properties بديلة لل classes دية
وبنفس الاسم التالي..

ng-touched ⟹ touched property
ng-untouched ⟹ untouched ~
ng-dirty ⟹ dirty ~
ng-pristine ⟹ pristine ~
ng-valid ⟹ valid ~
ng-invalid ⟹ invalid ~

عشان نقدر نوصل ال properties دية، هنخزنها جوا v-r-t.

(ex)

```
<form #userForm = "ngForm" >
  <input type="text" #name = "ngModel" />
```

ال ngModel بتحتوي على properties
بتلاقي ال classes الخاصة بال control.

```
<p>{{name.dirty}}</p>
<p>{{name.touched}}</p>
</form>
```

} → الناتج هيكون true او false

(notes) عشان نوصل لأي معلومات من ال html element
لازم نستخدم ال template reference variable،
وهي عبارة عن variable بيحتوي على معلومات عن
ال element.

④ providing visual feedback ادى style

← الـ visual feedback هو ان إحنا ندى style معان تبين الـ validation.

← الـ display msg for feedback هو ان احنا نعرض رسالة تبين الـ validation.

الخلاصة: لو ندى style معان توضح الـ validation يبقى اسمه visual feedback

لو هنعرض رسالة معان توضح الـ validation يبقى اسمه display message.

ex) لو الـ value كانت invalid value ، ادى style للـ input خلية لونه احمر.

```
ex <input ... [class.t.] = "name.invalid" #name="ngModel" >
.t{
    color: red;
    border: ... red;
}
```

بنوصل المعلومات / التفاصيل الـ element من خلال t-r-v

```
ex <input ... [class.t.] = "name.invalid && name.touched" ... >
```

ضيف class اسمه (t) لو

الممسوحة ضوئيا بـ CamScanner

امتى ال touch. يرجع بـ true؟ لو عملت focuse على input
امتى ال valid. ~ ~ ؟ لو ال value حققت ال validation
اللى انا محددة لل input.
امتى ال dirty. ~ ~ ؟ لو ال input استقبل قيمة
يعني اتوسخ ☺.

" ازاى نعمل validation rules على ال input في ال html؟
امتى نتحقق من ال validation في ال html مش ال ts؟
" ازاى نعمل validation في ال html مش ال ts؟

" عن طريق استخدام ال Regular Expression،
بنعمل regex في ال html من خلال html property
اسمه pattern="".

(ex) <input ---- pattern="^[0-9]{11}$" ---- />
RegEx

الممسوحة ضوئيا بـ CamScanner

(5) display error message

عشان نتحكم في ظهور element معين بناءاً على condition، هنستخدم (*ngIf).

```
<input ——— #name="ngModel" —— />
<p *ngIf="name.invalid || name.untouched">
  there is error here
</p>
```

or

```
<p [class.d-none]="name.invalid">
</p>
```

ازاي نعمل مجموعة error messages مختلفة لنفس الـ element، تعتمد على الـ error اللي حصل؟

```
<input —— #name="ngModel" —— />
<div *ngIf="name.errors && (name.invalid || name.touched)">
    <p [class.d-none]="name.valid">—</p>
    <p [class.d-none]="name.dirty">—</p>
    <p *ngIf="name.errors.required">—</p>
    <p *ngIf="name.errors.pattern">—</p>
</div>
```


الممسوحة ضوئيا بـ CamScanner

ازاى نخلى ال button يكون disabled ،
حلول ما فيه حاجة invalid ؟

```
<form #userForm = "ngForm" >
  =
  =
  <input type="submit" [disabled] = "userForm.form.invalid" >
</form>
```

ازاى نلغى اى validation تبع ال browser ؟
ازاى نلغى تأثير ال browser على form ؟

```
<form #userForm = "ngForm" novalidate >
  =
</form>
```

ده property بتميز من ال validation الخاصة بال browser ، لما ال validation بتاع ال browser ميخشش على ال validation بتاعى .

⑥ send the data in form to server

عن طريق ال event binding .
فيه event خاص بال form اسمه (ngSubmit)

① <form ——— (ngSubmit)="sendData()">

</form>

② userModel = new User('', '', '', '');
sendData() {
this.userService.enroll(this.userModel).subscribe();
}

③ enroll(user:User) {
return this._http.Client.post<any>("http——", user);
}

ده بيرجع ب observable ، عشان نستخدم
ال observable لازم نعمل subscribe .

الممسوحة ضوئيا بـ CamScanner

ملحوظة

١- ال form بيكون من نوع (FormGroup)
٢- ال input " " " (FormControl)

٣- ال angular بيتعامل مع ال template drivin forms
من طريق ال Reactive form بشكل internally
يعني behind the scene.

ال angular بيحول ال template form اللي
انتي عاملة له reactive form في behind the scene
وراء الكواليس

Developer → template drivin form → Angular → reactive form

CamScanner
الممسوحة ضوئيا بـ CamScanner

Reactive Forms

الـ reactive forms هنستخدمها مع الـ complex senarios.
أغلب الـ logic هيكون في الـ component class.

① Add HTML form

```
<form>  ---> formGroup in angular
  <input type="text">  ---> formcontrol in angular
  <input type="email">  ---> FC
  <input type="password">  ---> FC
</form>
```

كل input بنقول عليه instance من الـ FormControl.
كل input بنسميه FormControl.
مجموعة الـ input بنقول عليه FormGroup.

```
name: FormControl
email: FormControl
pass: FormControl
```

Form Group

هشان نستخدم الـ Reactive form لازم نعمل
import لـ (ReactiveFormsModule).

CamScanner
الممسوحة ضوئيا بـ CamScanner

⇐ ال form بالنسبة لل angular ،
أى form هو formGroup و كل
item جواه هو formControl

② create Form model

ال form model هو عبارة عن FormGroup بيحتوى على مجموعة FormControls.

ال form model هو محاكاة لل form جوا ال typescript.
نبدأ عملية انشاء form من خلال ts code.

```
registerForm = new FormGroup({
    userName: new FormControl(' '),
    email : new FormControl(' '),
    pass1 : new FormControl(' '),
    pass2 : new FormControl(' ')
});
```

? ممكن بالدفولت

كده إحنا انشأنا form model اسمه (registerForm)

③ binding b/w html form & form model :-

```
<form [FormGroup]="registerForm">
    <input type="text" formControlName="userName" />
    <input type="email" ———————— ="email" />
    <input type="password" ———————— ="pass1" />
    <input type="password" ———————— ="pass2" />
</form>
```

August
CamScanner
الممسوحة ضوئيا بـ CamScanner

nesting Form Groups :-

← هي عبارة عن FormGroup بداخل FormGroup .
← كل FormControl بيبقى تبع الـ FormGroup المباشر ليه .

(ex)

```
registerForm = new FormGroup({
    userName : new FormControl(''),
    email : new FormControl(''),
    address : new FormGroup({
        state : new FormControl(''),
        city : new FormControl('')
    })
})

<form [formGroup]= "registerForm" >
    <input type="text" formControlName = "userName" />
    <input type="email" ———————— = "email" />

    <div formGroupName = "address" >
        <input formControlName = "state" >
        <input ———————— = "city" >
    </div>
</form>

{{ registerForm.value | json }}
```


CamScanner
الممسوحة ضوئيا بـ CamScanner

Q) ازاى نشغل ازرار ال edit ؟
Q) ازاى نظهر data فى ال form من API ؟
Q) ازاى نجيب data من API ونحطها فى ال form ؟

عن طريق built-in method اسمه setValue() بياخد object جواه: اسم ال formControl وال value بتاعته.

```
<button type="submit" (click)="getdatafromAPI()">
</button>

getdatafromAPI(){
    this.registerForm.setValue({
        userName: "Amir",
        email : "@-",
        address :{
            state: 'egypt',
            city : 'egy',
        }
    });
}
```

⇐ ال setValue() بنستخدمه لما نعمل initialization لل form model كامل.

لأجزاء معينة فقط

⇐ مش نعمل initialization لل inputs معينة فقط، مش كل ال inputs فى ال form model، بالتالى هنستخدم patchValue() بدلاً من setValue()

CamScanner
الممسوحة ضوئيا بـ CamScanner

④ FormBuilder service

الـ service دي بتخلي الـ form model بشكل:
ابسط / بكود أقل.

ما هي إلا service بتبسط عملية انشاء الـ form model.

(ex)

لي service بنطلب من dependency injection عن طريق الـ constructor.

```
constructor( private fb: FormBuilder){}

registerForm = this.fb.group({
    name : [''],
    email : [''],
    address : this.fb.group({
        state : [''],
        city : [''] 
    })
});
```

من الآخر:
بدلا من new FormGroup() ← this.fb.group()
new FormControl ← []

دي عبارة عن تبسيط في الكتابة لكن internally الـ group() هترجع instance من الـ FormGroup، الـ [] هترجع FormControl

CamScanner
الممسوحة ضوئيا بـ CamScanner

(5) Validation

(5.1) simple validation :-

هنستخدم class اسمه Validators بيحتوى على
مجموعة من static properties .

(ex)
```
registerForm = this.fb.group ({
    name: [ '' , Validators.required ]
});
```

visual feedback

formGroup formControl

(1)
```
<input [class.t] = "registerform.get('name').invalid
    && registerform.get('name').touched" >
```

```
.t{
    border: 1px solid red;
}
```

message error

(2)
```
<p [class.d-none] = "registerform.get('name').invalid" >
    there is an error
</p>
```

الـ parameter الـ FormControl
الاول هو الـ default value .
الثاني هو الـ validation .
الثالث هو الـ
لو حابب تكتب أكثر من validation بنعملهم في array .
```
name: [ '' , [Validators.required , Validators.minlength(16)] ]
```


CamScanner
الممسوحة ضوئيا بـ CamScanner

conditions

* عشان نـ check الـ state بتاع الـ input ،

```
registerForm.get('name').valid
                        .invalid
                        .touched
                        .untouched
                        .dirty
```

* عشان نـ check validation معين .

```
registerForm.get('name').required
                        .minlength
```

اسم الـ form model — اسم الـ form control — اسم الـ validator

لازم ننادي على property اسمه errors . اولاً

```
registerform.get('name').errors.required
registerform.get('name').errors.minlength
```

* عشان نتضمن إن الـ property اللي اسمه errors لو مش موجود ، أو لو رجع ب null ، أو أي مشكلة حصلت ، يفضل نخليه optional

```
registerform.get('name').errors?.required
```

optional

يعني لو الـ errors موجود ، اعمل access لـ required .

CamScanner
الممسوحة ضوئيا بـ CamScanner

(5.2) custom validation

يفضل الـ logic بتاع الـ validation تتكتب في ملف ts منفصل لوحده، عشان الـ reusability.

(ex) userName.validator.ts

الـ validator في النهاية ما هو إلا function وبالتالي لازم ياخد parameter من نوع AbstractControl.

```
export function forbiddenValidator(control : AbstractControl){
    const x = /^[0-9]{15}$/.test(control.value);
    return x ? {"forbidden" : {value: control.value}} : null;
}
```

عشان نقدر نعمل import للـ validator

true / false — if — object — else

```
registerForm = this.fb.group({
  name: ['', forbiddenValidator]
})
```

اسم الـ key الـ راجع هو الـ error

```
<p *ngIf="registerform.get('name').errors?.forbidden"></p>
```

١- الـ validator ما هو إلا function واخد parameter من نوع AbstractControl.

٢- بيرجع error عبارة عن {key : value}، اسم الـ key هو اسم الـ error الـ راجع.

٣- يفضل أي custom validator يكون في ملف ts منفصل عشان نقدر نستخدمه في أي component.

CamScanner
الممسوحة ضوئيا بـ CamScanner

ازاى نعمل validator بيستقبل regex
كـ parameter ؟

هنعمل function بـ return function.

```
export function V(x:RegExp):ValidatorFn{
    return (control:AbstractControl)=>{
        const f = x.test(control.value);
        return f?{"e":" "}:null;
    }
}
```

```
registerForm = this.fb.group({
    name:['', V(/^011$/)]
})
```

```
<p *ngIf="registerForm.get('name').errors?.e" > </p>
```

(ملحوظة)

① الـ formGroup بيورث من الـ AbstractControl
الـ formControl بيورث من الـ AbstractControl.

② الـ validator اللى بنكتبه فى الـ formControl بيبقى على مستوى الـ control اللى اتكتب فيه خطأ،
لكن لو كتبنا الـ validator فى الـ formGroup هيبقى على مستوى الـ form كله.

CamScanner
الممسوحة ضوئيا بـ CamScanner

(5.3) cross-field validation

ما هو إلا custom validator ، لكن بيكون على مستوى الـ formGroup كلها .

الـ cross field validator بيتنفذ على مستوى الـ form group ، وليس على مستوى الـ controller الواحد .

```
export function ConfirmValidator(controls:AbstractControl){
    const pass1 = controls.get('pass1');
    const pass2 = controls.get('pass2');

    return (pass1 && pass2 && pass1.value !== pass2.value)
            ? {'misMatch' : true}
            : null ;
}
```

```
registerForm = this.fb.group({
    pass1 : [''],
    pass2 : ['']

}, {validators:[ConfirmValidator]});
```

```
<p *ngIf = "registerForm.errors?.misMatch" > </p>
```


CamScanner
الممسوحة ضوئيا بـ CamScanner

(5.4) dynamic validation

* يعني نضيف ال validation أو نشيلها بناءاً على condition معين في ال runtime.

(مثال) لو ضغط على checkbox اسمه subscribe يبقى نضيف required Validator ال email control.

```
<Input type="email" formControlName="email">
<input type="checkbox" formControlName="subscribe"
  (click)="setemailValidation">
```

```
registerForm = this.fb.group({
    email: [''],
    subscribe: [false]
})

setemailValidation(){
    this.registerform.get('subscribe').valueChanges.subscribe(
      (value)=>{
        if(value)
        { this.registerform.get('email').setValidators(
            validators.required)
        } else{
            this.registerform.get('email').clearValidators();
        }
      }  this.registerform.get('email').updateValueAndValidity();
}
```


CamScanner
الممسوحة ضوئيا بـ CamScanner

٧ إيه هي أهمية الـ updateValueAndValidity()؟

٨ لو عملنا تغيير الـ validator في الـ runtime مع
٩ الـ controller، لازم نعمل update عشان يحس
بالـ effect الجديد

١٠

١١

١٢

١

CamScanner
الممسوحة ضوئيا بـ CamScanner

⑥ Add dynamic form controls

يعني نقدر نضيف input / formControl جديد في الـ runtime.

مثال: ان احنا نعمل زرار للـ teacher يضيف سؤال جديد الـ quiz، فيطلعلة form جديدة يملى عليها.

اوكي؟ عن طريق الـ FormArray ← this.fb.array()
الـ FormArray بيسمحلنا نضيف input في الـ runtime.

① ex) registerForm = this.fb.group({
    email : [''],
    addemailfield : this.fb.array([])
});

② get addemail() {
    return this.registerform.get('addemailfield') as FormArray;
}
addNewEmail() { this.addemail.push(this.fb.control('')) }

③ <button (click) = " addNewEmail()"> add </button>

<div formArrayName = "~~addemailfield~~" "addemailfield"
    "ngFor = "let email of addemail.controls ; index as i" >

    <input type="email" [formControlName] = "i" >

CamScanner
الممسوحة ضوئيا بـ CamScanner

لية عملنا ! [formControlName] = "1"
مش ، [formControlName] = "email"

عشان نقدر نمسكه بسهولة عن طريق ال index.
ex: addEmailField [0]
[1]
وهكذا...

امتى نستخدم ال FormArray ؟

لما نحتاج نعمل repeat لل control معين ، أو لل formGroup .

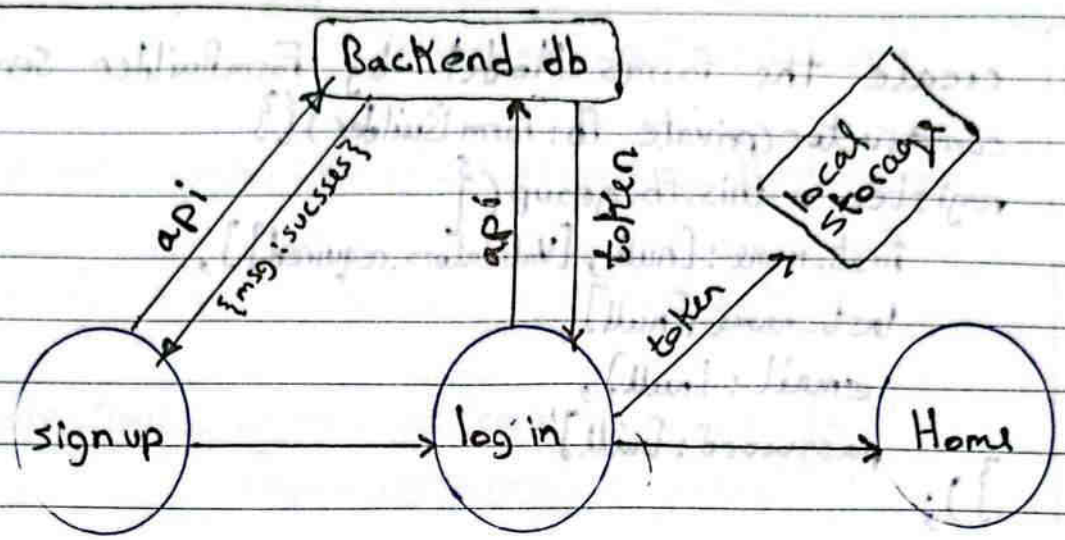
CamScanner
الممسوحة ضوئيا بـ CamScanner

Session (7)

1- Authentication & Authorization
2- Jwt
3- routing guard
4- BehaviorSubject
5- Route parameter
6- search pipe
7- lazy loading
8- interceptor
9- production

CamScanner
الممسوحة ضوئيا بـ CamScanner

① Authentication



يفضل نستخدم الـ Reactive forms مع الـ complex forms
اللي بتحتاج business logic معقد زي الـ registeration .

create sign up & login forms steps

1- add html form in html.
2- create the form model in ts.
3- bind b/w html form and form model.
4- add the validation
5- provide the visual feedback and error msgs.
6- post the data to the Server via the APIs.
7- → at registration → navigate to login page.
→ at login → navigate to home page and provide the authorization by using the Token in the localStorage.

CamScanner
الممسوحة ضوئيا بـ CamScanner

registeration form

① create the form's Model by FormBuilder service.

```
constructor(private fb: FormBuilder){}
register = this.fb.group({
    first_name : [null, [Validators.required]],
    last_name :[null],
    email : [null],
    password : [null]
});
```

② add the html form

```
<form>
    <input type="text">
    <input type="email">
    =
</form>
```

③ bind b/w html form & form Model.

```
<form [formGroup]="register">
<input formControlName="first_name">
<button (click)="sendData()" —>
(or)
<form (ngSubmit)="sendData" —>
```


CamScanner
الممسوحة ضوئيا بـ CamScanner

(4) add Validation

```
password: [null, [Validators.required,
                  Validators.pattern('^[A-Z]\d{10}')]]

email: [null, [Validators.required, Validators.email]]
```

(5) provide visual and message feedback

```
<div *ngIf="register.get('email')?.errors &&
            register.get('email')?.touched">

    <p *ngIf="register.get('email')?.errors.required"></p>
    <p *ngIf="register.get('email')?.errors.email"></p>
</div>
```

(6) post user data / integrate with auth API

(at) auth.service.ts

```
constructor(private h:HttpClient){}
register(data:any):Observable<any>{
    return this.h.post('URL', data);
}
```

(at) register.component.ts

```
sendData(){
    if(this.register.invalid){return;}
    else{this.authSevice.register(this.register.value).subscribe(
        (data)=>{if(data.message == 'success'){}
                 else{}
        });
}
```

navigate to

error

CamScanner
الممسوحة ضوئيا بـ CamScanner

② JWT → json web token

Q) ايه هو ال authentication وال authorization؟

ال authentication هو عملية تسجيل ال user في الموقع عن طريق ال registration وال login.

ال authorization هو ما بعد ال login، هو ال administration اللي بتحدد نية إيه اللي مسموح لل user إنه يشوفه وايه لأ.

- Authentication
- login
- who are you?

- Authorization
- what permissions do you have?

ال authentication هو اني اتأكد إن ال user متسجل عندي.
ال authorization هو الصلاحيات اللي ال user يقدر يعملها.

Q) ازاي نتأكد ال user عمل login؟

عن طريق ال token اللي بيرجع في ال http response الخاص بال login request.
وال token هو عبارة عن معلومات ال user اللي عمل login لكن متشفرة، والمعلومات دي بتكون معلومات insensetive يعني مفيهاش معلومات تضر بأمن ال user.

CamScanner
الممسوحة ضوئيا بـ CamScanner

Q) ايه هو الـ token؟ وبيتكون من ايه؟

١- الـ token هو عبارة عن معلومات معينة عن الـ user اللي عمل login، لكن المعلومات دي بتكون متشفرة.

٢- فايدة الـ token إنه بيعرفني هل الـ user عمل login ولا لا، ومن خلال معلومات الـ user في الـ token، أقدر أحدد صلاحيات الـ user / الـ authorization.

٣- الـ token بيتكون من ٣ أجزاء:

١- أول جزء هو الـ Header وده عبارة عن الـ algorithm اللي اتعمل بيه التشفير، ونوع الـ token.

٢- ثاني جزء هو الـ payload وده عبارة عن الـ data بتاعة الـ user.

٣- ثالث جزء هو الـ signature وده عبارة ناتج تشفير الـ Header والـ payload بالـ base64، ومضاف إليه secret key، قيمة الـ secret بتكون موجودة في الـ server فقط، بنستخدم الـ secret لما نعمل create للـ token ونبعته للـ user ولما نعمل decode للـ data اللي جاية من الـ user.

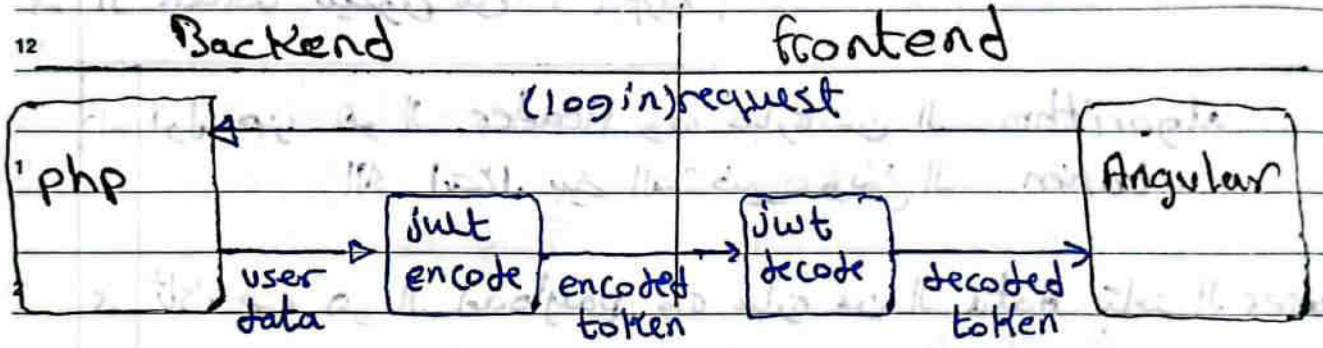
signature = (Header + payload)base64 + secret

CamScanner
الممسوحة ضوئيا بـ CamScanner

@) ايه هو الـ jwt ؟

الـ jwt عبارة عن library ، الـ backend بيستخدمها عشان يعملوا تشفير للـ data بتاعة الـ user .
واحنا كـ frontend هنستخدمها عشان نعمل ذلك تشفير للـ token اللي اتبعت من الـ backend .

* الـ jwt هي مكتبة بنستخدمها عشان نعمل encode و decode لـ data معينة



* >> npm i jwt-decode ↵

```
* from import jwtDecode from "jwt-decode";
* userData:any = null;
* saveUserData(){
    let encodedToken = JSON.Stringify(localStorage.getItem('token'));
    let decodedToken = jwtDecode(encodedToken);
    this.userData = decodedToken;
  }
```

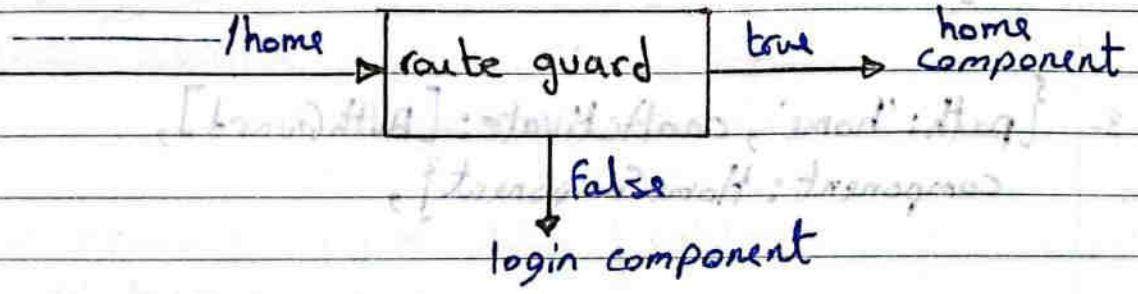

CamScanner
الممسوحة ضوئيا بـ CamScanner

③ guard ng g g auth

الـ angular route guards هما عبارة عن interfaces
بيسمحلنا نتحكم في امكانية الوصول للـ route بناءا"
على conditions بنكتبها في الـ class اللي بيـ implement
الـ interface.

الـ guard هو built-in interface من خلاله نقدر نتحكم
في الـ accessibility بتاعة الـ route.

انواع الـ guard :
CanActivate interface
CanDeactivate ~
CanLoad ~
CanActivateChild ~
Resolve ~



الـ guard هو boolean function بيـ return
إما true أو false.

CamScanner
الممسوحة ضوئيا بـ CamScanner

Example

1- npm >> ng g g auth ↵

2- auth.guard.ts

```
export class AuthGuard implements CanActivate {
  - constructor(private a:AuthService,
                private r:Router){}
  canActivate(——):——{
    if(this.a.userData.getValue()){
      return true;
    }else{
      this.r.navigateByUrl('/login');
      return false;
    }
  }
```

3- {path:'home', canActivate:[AuthGuard],
component:HomeComponent},

CamScanner
الممسوحة ضوئيا بـ CamScanner

variables ولكن ال observable
انها ال methods ال observable

④ BehaviorSubject

آية هو ال subject؟

ال subject هو observable و observer في نفس الوقت، يعني نقدر نشغلة subscribe ونقدر نستعمله كـ object نشيل فيه data.

الفرق بين ال subject وال observable، إن ال subject شغال multi casting وال observable شغال uni casting

uni casting يعني كل subscribe بياخد نسخة خاصة بيه observable بيشتغل عليها.

multi casting يعني ال subject بتشغل execution ببدأ ال subscribers يعمل listen التغيير حال data بالترتيب بتاعهم، يعني كل subscribe بيشتغل على ال observable بتاع اللي قبله.

ليه نستخدم ال subject؟ مثلاً نحول ال observable من uni لـ multi casting

ايه هو ال BehaviorSubject؟
هو نوع من انواع ال subject، ليه ميزتين:
① بيبعت معاه initial value.
② بيخزن أحدث value دايماً.

CamScanner
الممسوحة ضوئيا بـ CamScanner

ملحوظة:

١- الـ property اللي من نوع BehaviorSubject
بقدر اجيب الـ value بتاعته من خلال getValue() فقط.
وأقدر اخزن قيمة value من خلال next() فقط.

```
ex: userData = new BehaviorSubject(null);

    userData = 20;  ✗

    userData = next(20); ✓

    console.log(userData); ✗

    console.log(userData.getValue()); ✓
```

٢- عشان نحس بأي تغيير لازم نعمل subscribe.

```
userData.subscribe(() => {
    if(userData.getValue()){
    } else{
    }
}
```


CamScanner
الممسوحة ضوئيا بـ CamScanner

(5) Route parameter,

routerLink = "/details" ← الـ route العادي

[routerLink] = ["/details", p1, ...] ← الـ route + parameter

(ex) (1) {path: "details/:x/:y", ____}

(2)
```
<div *ngFor="let m of movies">
    <a [routerLink] = ['/details', m.id, m.name] >
    ____
    </a>
</div>
```

(3)
```
import {ActivatedRoute} from '@angular/router'
constructor(private _ar: ActivatedRoute){}
currentId: number = 0;
currentName: string = '';
itemDetail: any = {};
ngOnInit(): void {
    this.currentId = this._ar.snapshot.params.x;
    this.currentName = this._ar.snapshot.params.y;
}
```

* الـ ActivatedRoute هو عبارة عن class بيشتغل مع الـ route اللي محولة active، بياخد snapshot من الـ url الحالي في الـ browserURL يعني الـ route اللي شغال حالياً، وبيطلع object اسمه params، فيه الـ parameter اللي اتبعتت في الـ path.

CamScanner
الممسوحة ضوئيا بـ CamScanner

للتجربة

```
constructor (private _ar: ActivatedRoute){}
x:any = this._ar.snapshot.params
ngOnInit():void {
    console.log (this._ar)
    ———— (this._ar.snapshot)
    ———— (this._ar.snapshot.params)
}
```

الـ ActivatedRoute بياخد snapshot من الـ current URL
اللي شغال في الـ browser engine وبيطلع منه object
اسمه params بيحتوي على الـ parameters اللي موجودة في الـ URL
على شكل object property.

(ex) www.example.com? x=1 & y=50

```
console.log (this._ar.snapshot.params);
{ x: 1,
  y: 50
}
```


CamScanner
الممسوحة ضوئيا بـ CamScanner

(6) search pipe

i/p → pipe → o/p
i/p + edit

ال pipe عبارة عن function بياخد input وبترجعه كـ output بعد تعديل معين عليه.

1-) ng g p search

2) search.pipe.ts

```
.transform( movielist : any[] , term : string ) any[] {
        return movielist.filter( (item) => item.title.includes (term))
}
```

جسم الـ object

3) <input [ngModel] = "term" type="search" —>

```
<div *ngFor = "let m of movielist | search: term" >
</div
```

4) term : string = '';

→ return movielist.filter((item) => {
    return item.title.toLowerCase().includes(term.toLowerCase())
})

CamScanner
الممسوحة ضوئيا بـ CamScanner

⑦ lazy loading

١- هل ممكن نستخدم أكثر من module؟ آه ينفع.
٢- الـ module هو عبارة عن set of components.
٣- ازاي نعمل module باستخدام الـ cmd؟
>> ng g m setting --routing ←
اسم الـ module.

٤- ازاي نعمل component تبع module معين؟
>> ng g c setting / profile ←
اسم الـ c ↖ اسم الـ m

① اعمل الـ module والـ components بتوعه بالـ cmd.
>> ng g m setting --routing
>> ng g c setting / profile
>> ng g c setting / password

② اعمل routing path يـ navigate عالـ module.
هنعمل الـ path في ملف الـ routing الرئيسي بتاع الـ app
app-routing.module.ts

{path: 'setting', loadChildren: () => import('./setting/setting.module')}
← الـ import method بتـ promise

∴ {path: 'setting', loadChildren:
() => import('./setting/setting.module').then(
(r) => r.SettingModule)}
(result)

CamScanner
الممسوحة ضوئيا بـ CamScanner

- السطر اللي فات {path: 'setting', loadChildren: ....}
معناه متحملش load للـ setting module اللي ما افتح
الـ routing اللي اسمه (/setting)..

- الـ concept دي اسمه ← lazy loading.
- lazy loading يعني مش هيتحمل loading اللي ما أدخل عليه.
- الـ lazy loading بيشتغل على مستوى الـ module مش الـ component.

Q) ايه هو الفرق بين: ng g c c1 @ ؟
ng g c m1/c1 @

الاول هيعمل generate للـ component في أقرب module منه.
الثاني ← ← في الـ m1 module.

- الـ app module هو الـ root module الأساسي
وباقي الـ modules بتتفرع منه، يعني اي module غير
الـ root، هيكون child module.

Q) يعني ايه lazy loading concept ؟

يعني متحملش loading للـ module اللي ما تحتاج request.

ازاي؟ عن طريق إن إحنا نعمل import للـ module في الـ
routing path من خلال property اسمه loadChildren.
بيستقبل promise method اسمه import().

CamScanner
الممسوحة ضوئيا بـ CamScanner

Q) امتى نستخدم الـ lazy loading ؟

لو الـ scale بتاع الـ project بقى أكبر ، واحتجنا نقسم الـ project لأكثر من module .

Q) ليه نستخدم الـ lazy loading ؟

عشان نحسن الـ performance عن طريق إن الـ lazy loading مش هيحمل الـ module معين إلا لما نحتاجه / request / لما نطلبه .

lazy loading هو concept معناه تحميل الـ module لما اطلبه / احتاجه .

الـ SPA بيحمل loading لكل الـ modules تلقائياً أول ما الـ application يقوم ، بالتالي مع المشاريع الكبيرة اللي فيها routes كتير ، الـ loading time هيزيد ، يعني الـ performance هيقل ، بالتالي هنحتاج الـ lazy loading يعني اعمل load لما احتاجه وبس .

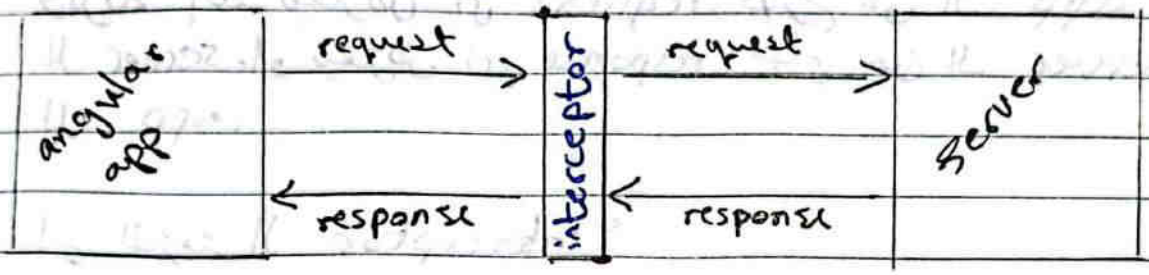
Q) ازاي نعمل loading للـ module لما نحتاجه ؟

عن طريق الـ dynamic import من خلال ان إحنا نعمل loadChildren في الـ routing .
الـ loadChildren هو property خاص بالـ route ، بياخد arrow function بيرجع import method فيه الـ path بتاع الـ module ثم بياخد الـ class بتاع الـ module .

CamScanner
الممسوحة ضوئيا بـ CamScanner

⑧ interceptor إعتراض / المعترض

- ال interceptor هو عبارة عن layer بين ال client side وال server side ، بيعترض طريق ال request من ال angular app لل server، لية؟ عشان نعمل بية ال request قبل ما يتبعت لل server.



ازاى نعمل interceptor بال cli؟

>> ng g interceptor apiKey

* ال interceptor هو layer كل ال http request اللى طالع من ال angular app بيعدى عليه قبل ما يتبعت لل server.

* عن طريق ال interceptor نقدر نعمل تعديل على كل ال request قبل ما تتبعت لل server.

* ال interceptor مبيقدرش يعدل على ال request مباشرة، لكن بياخد clone من ال request ويعمل عليه.

* عشان ال interceptor يشتغل، لازم نربطه بال app من خلال ال app module.

CamScanner
الممسوحة ضوئيا بـ CamScanner

Q) ايه هو ال Interceptor ؟

ال Interceptor هو concept عبارة عن layer بين ال client side وال server side.

Q) ايه فكرة ال interceptor ؟

فكرته إنه يعترض اى request خارج من ال app لل server، او يعترض اى response جاى من ال server لل app.

Q) ايه لازمة ال Interceptor ؟

من خلال ال interceptor نقدر نعمل أى تعديل على اى http request طالع من ال app بتاعنا، قبل ما يوصل لل server.

(مثال) ١- نقدر نضيف apiKey لاى request طالع.
٢- نقدر نحط ال token فى اى request طالع
٣- نقدر نعدل أى حاجة فى ال request
٤- نقدر نعدل أى حاجة فى ال header

Q) ايه هو ال intercept() function ؟
هو دى ال layer / ال interceptor، بياخد حاجتين:
① ال request: سواء اللى طالع أو اللى جاى.
② ال next: دى التعديل اللى هيتعمل فى ال request.

CamScanner
الممسوحة ضوئيا بـ CamScanner

1- ng g interceptor apikey ↵

2- apikey.interceptor.ts

```
intercept(request —, next —): — {
    let newRequest = request.clone({
        url: request.url.replace('key=', 'key=12345...')
    });
    return next.handle(newRequest);
}
```

3- app.module.ts

```
providers: [
    { provide: HTTP_INTERCEPTORS,
      useClass: interceptor اسم الـ class تبع الـ ,
      multi: true
    },
],
```


CamScanner
الممسوحة ضوئيا بـ CamScanner

⑤ production

٣ خطوات:

اول خطوة: هنروح لملف الـ index.html من الـ <base> tag
هنضيف dot قبل الـ slash في الـ "href="./"
وده عشان نظبط الـ relative pattes

① <base href="./">

تاني خطوة: هنروح لـ app routing الاساسي اللي هو ملف
الـ app-routing.module.ts، هنضيف الـ useHash
عشان يضيف # hash قبل كل route path.

② @NgModule({
imports: [RouteModule.forRoot(routes, {useHash: true})],
=
})

ثالث خطوة: استخدم الـ webpack وده اهم من الـ
فيه الـ configuration الخاصة بتحويل / بتجهيز الـ
angular للعملية الـ deployment من خلال:
١- تجميع ملفات الـ css في ملف واحد
٢- " " " " html " " "
٣- تحويل الـ ts لـ js والـ sass لـ css etc...

③ ng build ← لو فيه errors في المشروع،
امر الـ build مش هيشتغل.

CamScanner
الممسوحة ضوئيا بـ CamScanner

ملحوظات

أ) امر الـ build مش هيشتغل لو فيه اى error فى المشروع.

ب) ناتج امر الـ build انه هيكون folder جديد اسمه (dist) إختصار لـ distribution ، وده هو الـ folder اللى هنجهزه deployment.

CamScanner
الممسوحة ضوئيا بـ CamScanner

Angular Material Design

Session (8)

1- introduction
2- getting started
3- generic material module
4- Typography
5- Components

CamScanner
الممسوحة ضوئيا بـ CamScanner

① introduction

ايه هو ال Angular Material ؟
هي عبارة عن library بتوفرلك UI components جاهزة،
ميزتها إنك اقدر اعمل implementation في المشروع بشكل سريع.

هي library خاصة بال angular، شبه ال bootstrap
لكن الفرق إن ال Angular Material محول لل angular خصوصاً.

② Getting started

ازاي نضيف ال angular Material في المشروع؟

```
>> ng add @angular/material
```

١- هيسألك ١ ايه شكل ال theme اللي انت عايزه؟
هل هتحتاج typography ؟
هل هتحتاج browser animation ؟
٢- بعدكده هيعدل ال components :
٢- ايه الملفات اللي حصلها تغيير:

app.module.ts	angular.json
style.scss	package.json
	index.html

③ عشان نشتغل على أي material component بنعمل ٣ خطوات.
١- بنعمل import لل component module في ال app module.
٢- نكتب اسم ال module في ال imports array.
٣- نكتب ال html code في المكان اللي احنا عايزين نظهره.

CamScanner
الممسوحة ضوئيا بـ CamScanner

③ generic material module

* يدخل في المشاريع الكبيرة نعمل module نـ import فيه كل الـ material modules، وبعد كده نقدر نستخدمه في الـ app module.

① >> ng g m material --routing

② add exports array in @NgModule

(ex)

```
const MaterialComponents = [ ];
@NgModule({
    imports : [ ]
    exports : [
        MaterialComponents
    ]
})
```

هنا هنكتب الـ module اللي بتاع الـ component اللي هنستخدمه

③ import material module in app module.

* الهدف من كل ده هدف تنظيمي مش أكثر، بحيث نخلي الـ module يكون generic يعني نقدر نركب فيه الـ components اللي احنا محتاجينها

CamScanner
الممسوحة ضوئيا بـ CamScanner

④ Typography

ما هي إلا classes · css تبع الـ material design بتخلينا نتحكم في شكل الـ text ستة الـ h1 ، h2 ، ... h6 ، p ، i .

ex) —
 —
 —
mat-body
mat-caption

(ملحوظة) ١- دايماً نبدأ بكلمة (mat-).
٢- الـ classes دول ممكن تعملها بايدك واحدة واحدة بال font-size ، لكن دة بتوفرلك وقتك ☺

⑤ Components

أيوة هي أهم الـ components اللي دايماً هتحتاجها من الـ angular material.

Button	list	basic autocomplete
Button togglе	gridlist	filter autocomplete
Icon	expansion pannel	
Badge	card	
process spinner	tabs	checkbox, radiobox
navbar	stepper	datepicker
sidenav	input	Tooltip
menu	select	snackbar
Dialog		

CamScanner
الممسوحة ضوئيا بـ CamScanner

ملحوظة

① اي حاجة تبع الـ material design بتبدأ بكلمة (mat)

② خطوات ثابتة للتعامل مع اي component تبع الـ material design :

① أعلى import في الـ app module مباشرة ،
او أعلى import في module خارجي يكون
للـ generic واعمل الـ module ده import في الـ
app module .

② استخدمه في الـ html .

that's it

CamScanner
الممسوحة ضوئيا بـ CamScanner

ايه اللي محتاج تذاكره بعد كده؟
ايه الجديد اللي يفضل تتعلمه؟

1- Angular Material Design
2- Angular primeNG
3- RxJS (reactive programming)
4- NgRx (state management)
5- Unit Testing
6- Angular + Firebase

CamScanner
الممسوحة ضوئيا بـ CamScanner

RKJS

٨

٩

١٠

١١

١٢

١

٢

٣

٤

٥

٦

٧

٨

٩

CamScanner
الممسوحة ضوئيا بـ CamScanner

١- ايه هي الـ RXJS ؟

- هي اختصار لـ Reactive Extentions for Js
- هي library جاهزة بتسهل التعامل مع الـ asynch-code
والـ RXJS بتوفر tools عشان كده.

٢- ايه الغرض الاساسي من الـ RXJS ؟

التعامل مع الـ asynchronous code بشكل ابسط
من خلال الـ tools اللي الـ RXJS بتوفرها.

٣- هل الـ Js هي synch ولا asynch ؟

الـ Js متمثلة في الـ V8 اللي بتنفذ الـ ecmascript
هي synchronous code.
ولكن الـ browser بيضيف features زيادة على الـ V8
عشان تحل asynchronous code.

٤- يعني ايه asynchronous ؟

يعني اكثر من task / code بيتعملوا execute في
نفس الوقت مع بعض.

مهم الـ function في الـ Js هي نوع معين من الـ objects.

Name Tel. Email

يونيه ٢٠١٦

السبت	٤	١١	١٨	٢٥
الأحد	٥	١٢	١٩	٢٦
الاثنين	٦	١٣	٢٠	٢٧
الثلاثاء	٧	١٤	٢١	٢٨
الأربعاء	١	٨	١٥	٢٢	٢٩
الخميس	٢	٩	١٦	٢٣	٣٠
الجمعة	٣	١٠	١٧	٢٤

CamScanner
الممسوحة ضوئيا بـ CamScanner

٥- اشرح اهم مصطلحات ، اهم elements اهم مكونات الـ RXJS ؟

-اولاً اهم elements في الـ RXJS هما : ١- الـ observable
٢- الـ observer
٣- الـ subscribtion

① - الـ observable هي data generator هي عاجة بتـ emit الـ data اللي بـ subscribe عليهم .

أو هي function بتشغل generate لشوية الـ values بتشغل subscribe عليهم .

② - الـ subscribe هي function بتـ run الـ observable ، يعني بتخلي الـ observable يـ emit شوية values يـ generate شوية values / data .

ومن خلال الـ subscribe methode نقدر نهندل الـ data اللي الـ observable عاملها generate او emit .

* عشان نشغل الـ observable لازم نشغل subscribe .
* كل ما نشغل subscribe الـ observable هيـ run .
* الـ observable نقدر نشغل فيه asynchronous task يعني مثلاً نطلبه يـ emit داتا جاية من API .

③ الـ subscribe() بيستقبل object بيحتوي على properties معينة وكل property منهم بيهندل جزء معين من الـ observable .

ex observableObj.subscribe({
next: بيهندل كل value بيتعامل emit .
error: بيهندل الـ error لو حصل .
});
↳ notifications

٩- افاهم ايه من ال observable وال subscribe؟

- ال observable هي function بتحتوي على شوية logic جواها، الفرق بينها وبين ال function العادية إنها تقدر تعمل emit لأي عدد من ال data مع مرور الوقت.

- ال observable لوحدها مبتقدرش تعمل حاجة، لازم نعمل subscribe عليها عشان نتعامل.

- فنقدر نتعامل مع ال data اللي مبعوتة من ال observable أو حتى ال errors اللي جاية من خلال handler أو observer object.

الشرح اللي فات يعتبر Quick start عن ال RXJS
الشرح اللي جاي هيكون تفصيلي أكثر عن:

1. observables
2. subscribtions
3. observers
4. creation functions
5. pipeable operators
6. subjects

CamScanner
الممسوحة ضوئيا بـ CamScanner

Observables (1)

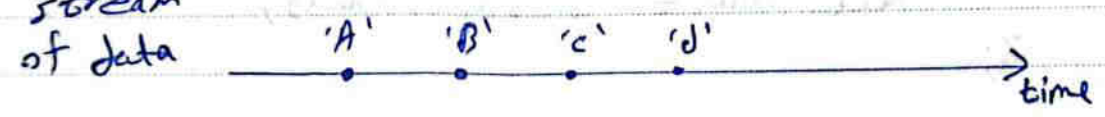
١- ايه هو الـ stream ؟

الـ observable بيعتمد على فكرة الـ streams ، يعني الـ observable يقدر يـ emit أي عدد من الـ values عند نقط زمنية مختلفة .

الـ stream هو عبارة عن مجموعة من الـ data بيظهروا عند نقط زمنية مختلفة مع مرور الزمن ، ومقدرش access الـ data أون ريكوست عليها من الوقت / الزمن اللي بتظهر فيه فقط ، يعني لو الوقت عدى وظهرت data بعدها مش هنقدر نـ access غير الـ data اللي ظهرت من الوقت اللي بنـ access فيه .

٢- ايه الفرق بين الـ array والـ stream ؟

الـ array هي مجموعة data موجودة طول الوقت ، يعني نقدر نعملهم / نعمل access لأي data منهم من أي وقت .
أما الـ stream هي مجموعة data بتظهر في أزمنة مختلفة ، ومنقدرش نـ access غير الـ data اللي بتظهر في الوقت اللي بنـ access فيه .



٣- ايه الـ approach اللي الـ stream بيتبعه ؟

الـ approach دي اسمه الـ reactive programming وديه الطريقة اللي الـ observable بيعتمد عليها .

CamScanner
الممسوحة ضوئيا بـ CamScanner

٤- إيه علاقة ال stream بال observable؟

← ال stream بيعمل emitting لل data من أكتر نقط زمنية مختلفة، وتقدر ال react ال data ما بتظهر.

← ال observable بيعمل emitting لل data من أكتر نقط زمنية مختلفة، وتقدر ال react ال data ما بتظهر، يعني لازم نستنى ال data تظهر عشان نقدر نتعامل معاها.

٥- ازاى نعمل observable؟

باستخدام constructor new Observable() اللى بيستقبل جواه ال logic بتاع ال observable كـ callback function.

```
(ex) const obs1 = new Observable(subscriber=>{
        subscriber.next('A');
        subscriber.next('B');
    });
```

٦) ازاى نعمل observer؟

هو مجرد object بيحتوى على properties وكل property منهم بيمثل notification معين ال observable بيبعته.

```
(ex) const observer = {
        next: value => console.log(value);
    };
```


CamScanner
الممسوحة ضوئيا بـ CamScanner

Q) ايه الفرق بين ال observable وال observer؟

ج: ال observable هو مجرد object بيحتوي على logic معين ويقدر يبعت emit لل data، وكل data تقدر بتمثل ال notification معين،
ال observable يقدر يبعت 3 diff notifications
1. next notification
2. error notification
3. complet

ج: ال observer هو object بيوصف ال reaction اللي هتعمله على ال data اللي اتعمل emit.

ج: ال subscribtion هي الكلمة اللي بتـ execute ال observable، باختصار هي بتـ run ال callback function اللي جوا ال observable وبتستقبل ال observer جواها.

Q) ازاي تعمل subscribtion؟

من خلال method تبع ال observable اسمها subscribe()،
ex: observable1.subscribe(observer);

Q) اشرح ال process اللي بيتم:
1. هتعمل subscribe على ال observable وده هيعمل subscribtion جديد هيـ execute ال observable وال observer.
2. كل مرة ال next notification بتعمل emit من خلال ال observable، ال next callback هيـ run من خلال ال observer، والعمليتين دول هيستمروا طول ما ال subscribtion شغال.

CamScanner
الممسوحة ضوئيا بـ CamScanner

subscribtion

observable —next()→ observer

subscribe(observer)

Q) امتى الـ subscribtion بيقف؟

من خلال طريقتين:
① باستخدام الـ methode unsubscribe()
② او هتتقفل بشكل تلقائي من خلال الـ logic اللي جوا الـ observable نفسه عن طريق ارسال error notification او complete notification.

Q) ليه لازم نتأكد إن الـ subscribtion هيقف؟

عشان ميحصلش اي side effect أو memory leak.

راجع الخطوات اللي هتتعمل:

① هنعمل observable جديد باستخدام new Observable.
② هنعمل observer object يستقبل الـ next notification.
③ هنعمل subscribtion بالـ observable من خلال الـ subscribe method.
④ هنقفل الـ subscribtion باستخدام الـ unsubscribe method.

Name Tel. Email

CamScanner
الممسوحة ضوئيا بـ CamScanner

Q) ازاى نعمل emit للـ data من الـ observable ؟

باستخدام الـ next() method ودى method
تبع object اسمه subscriber بنستقبله من
الـ callback اللى بنبعته من الـ observable.

```
const observable = new Observable(
    subscriber => {
    });
```

الـ logic اللى هيـ run لما
نعمل subscribe على الـ observable

Q) ايه اللى هيحصل لو عملنا أكتر من subscribe
على نفس الـ observable ؟!!

مهم جدا

كل subscribtion هيعمل execute
للـ logic اللى فى الـ observable بشكل
منفصل عن باقى الـ subscribtions.

الـ subscribing هو مجرد ان احنا بنعمل invoke
للـ callback function اللى جوا الـ observable.

each subscribtion is a separate independent
observable's execution.

CamScanner
الممسوحة ضوئيا بـ CamScanner

Q) ايه الفرق بين ال next ، error ، complete ؟

① next notification بيعني فيه data اتعمل emit
② complete بيعني ال observable خلص ومفيش حاجة ثانية هتتبعت ،

③ error بيعني فيه حاجة خطأ حصلت ، بالتالي هتوقف ال subscribtion بيعني هيعمل unsubscribe .

Q) ايه الفرق بين ال error وال complete ؟

الاثنين بيعملوا unsubscribe لل observable
الاثنين بيوقفوا عملية ال subscribtion .
ولكن الفرق إن ال error بيقدر يشيل data / payload معاه ، وال complete مبيعرفش .

next notification	بنستخدمه عشان نعمل emit داتا . تقدر تعمل emit لأي عدد من ال values .
error notification	بتعرفنا إن فيه error أو fail . تقدر تشيل payload . بيحصل مرة واحدة بس . بتوقف عملية ال subscribtion .

Name Tel. Email

يونيه ٢٠١٦

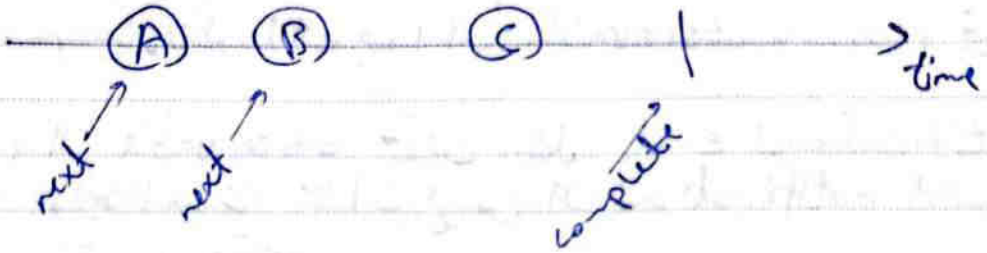
السبت	٤	١١	١٨	٢٥
الأحد	٥	١٢	١٩	٢٦
الاثنين	٦	١٣	٢٠	٢٧
الثلاثاء	٧	١٤	٢١	٢٨
الأربعاء	١	٨	١٥	٢٢	٢٩
الخميس	٢	٩	١٦	٢٣	٣٠
الجمعة	٣	١٠	١٧	٢٤

CamScanner
الممسوحة ضوئيا بـ CamScanner

Q) ايه هو ال marbles diagram ؟

هي رسمة بتوضح او بتمثل سلوك ال observable والسيناريوهات المختلفة اللي ممكن تحصل اثناء ال subscribtion .

مكونات ال marble diagram :
① بتمثل ال next notification كدايرة . Ⓐ .
② ~ ~ ~ error كعلامة X .
③ ~ ~ ~ complete كعلامة | .
④ ~ ال observable ك time line



② Subscriptions :-

Q) اشرح دورة حياة الـ subscription ؟

اولاً: بيكون عندنا observable ، والـ observable ميقدرش يشغل run بنفسه ، هو بس جوا شوية logic متخزنين جواه .

ثانياً: بنشغل subscribe بال observable وده هيشغل execute للكود اللي جوا الـ observable .

ثالثاً: لما بنشغل subscribe بيتم انشاء subscribition جديد والكود اللي جوا الـ observable بيـ run قيمة .

رابعاً: الـ observable تقدر يشغل emit notifications وايضاً عندنا ٣ أنواع من الـ notifications اللي نقدر نستخدم .

1) كل next notification جديد بيتبعت من الـ observable ، القيمة اللي قيمة هتتبعت لـ observer's handler الخاص بالـ next notifications ، وبعد ما يتم معالجة الـ next notification observable's code ال running هيكمل .

Name Tel. Email

CamScanner
الممسوحة ضوئيا بـ CamScanner

١) لو ال observable بعت error notification
هيتم عمل invoke لل error handler من ال observer،
بعد كده هيتم إغلاق ال subscribtion يعني عملية
ال execution observable متوقف، بعد كده
هيحصل running كام تبع ال observable اسمها
Teardown، ومن خلال ال cleanup ال observable.

٢) لو ال observable بعت complete notification
هيحصل نفس الحاجة مع ال ~ error.

خامساً: لو مفيش error أو complete اتبعت
توقف ال subscribtion من خلال ال
unsubscribe()، بعد كده ال observable
هيشغل ال teardown عشان يعمل cleanup.

Q) فيه كام طريقة نقدر نوقف بيها ال subscribtion؟

---

٣ طرق: ① نبعت error notification.
② ~ complete ~.
③ نستخدم ال unsubscribe() method.

Q) ايه هو ال teardown وازاي نعمله؟

هو مجرد كود / logic بنكتبه عشان نعمل
cleanup زي مثلاً توقف ال settimeout() مثلاً
او نمسح data من ال memory ملوش لازمة خلاص.

ال teardown هي function بيتعمل return من ال
callback function الخاص بال observable.

→ observable's callback returns teardown logic/function.

CamScanner
الممسوحة ضوئيا بـ CamScanner

③ Types of Observables :

* عندنا نوعين من ال observable وهما :
① hot observable ،
② cold ،

Q) امتى نقول عال observable إنه cold ؟
و " " " " " hot ؟
وايه أصلاً الفرق بينهم

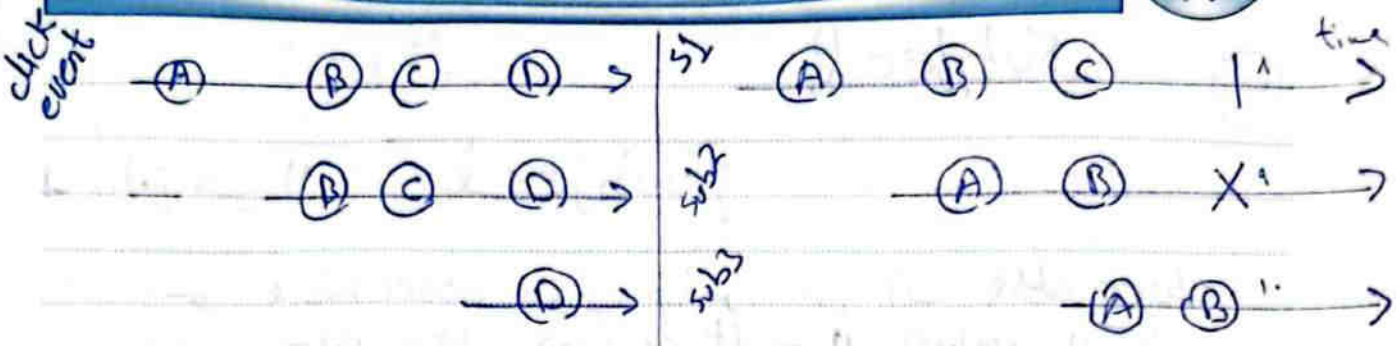
HOT	COLD
كلمة "hot" يعني كل ال subscripers هيستقبلوا نفس ال stream of values ، يعني هيشاركوا نفس ال value اللي بيبعت ال source .	كلمة "cold" يعني كل subscription بتعمل عال observable هتعمل execution جديد مستقل منفصل عن باقي ال subscriptions لنفس ال observable .
زي مثلاً : ال dom events state subject	يعني كل subscriper هيستقبل stream of values ~~منفصلة~~ جديدة خاصة بيه ومنفصلة عن باقي ال subscriber .
زي مثلاً لما تبعت API ، لا ، نعمل ajax .	

Name Tel. Email

يونيه ٢٠١٦

السبت	٤	١١	١٨	٢٥
الأحد	٥	١٢	١٩	٢٦
الاثنين	٦	١٣	٢٠	٢٧
الثلاثاء	٧	١٤	٢١	٢٨
الأربعاء	١ ٨	١٥	٢٢	٢٩
الخميس	٢ ٩	١٦	٢٣	٣٠
الجمعة	٣ ١٠	١٧	٢٤	

CamScanner
الممسوحة ضوئيا بـ CamScanner



كل subscriper بيستقبل values خاصة بيه.

كل ال subscripers هيستقبلوا نفس ال stream of value، نفس ال source، وهيستقبلوا نفس ال value في نفس الوقت مهي وهما بيطلبوا execute بشكل منفصل.

الخلاصة

الفرق الاساسي مابين ال cold / Hot observables هو إن ال cold observable بيعمل stream of data منفصل لكل subscriber. انما ال hot observable بيعمل نفس ال stream of data لكل ال subscribers في نفس الوقت.

← لو حصل emit من داخل ال observable، اذاً ال observable دة هو cold.

← ~ ~ emit ~ خارج ال ~ ، اذاً ~ ~ ~ hot.

CamScanner
الممسوحة ضوئيا بـ CamScanner

(4) Subject

إيه هو الـ subject ؟

هو observable، نوع خاص من الـ observable بيسمحلنا نعمل multicasting للـ values لأكتر من observer، subscriper.

هي شبه الـ event emitter، وجه الشبه إن الـ event لما بنعمله emit بيقوم باعمله multicast لكل الـ listeners اللي عليه.

الـ subject هو observable و observer في نفس الوقت، بالتالي:

① تقدر تـ subscribe على الـ subject زي ما بتعمل مع الـ observable العادي.

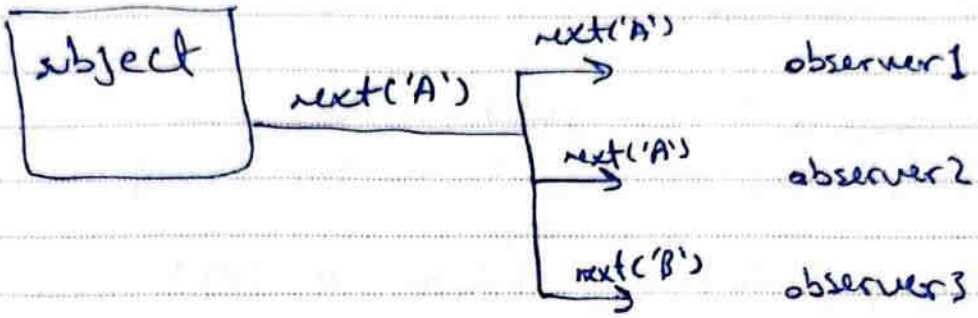
② تقدر تستخدم الـ next() والـ error() والـ complete() مع الـ subject زي ما بتستخدمهم مع الـ observer العادي.

subject = observable + observer

CamScanner
الممسوحة ضوئيا بـ CamScanner

٢- هل ال subject هو hot observable / cold ؟

ـ هو hot observable لأنه بيعمل multicasting
يعني بيعمل emit لل data وال notification
هيتبعت لكل ال subscripers اللي عليه وهيستقبلوا
نفس ال data في نفس الوقت .



٣- ايه الفرق بين ال subject وال observable ؟

ال subject هو عملية دمج ما بين ال observable
وال observer يعني نقدر نعمل عملية subscripe بالتالي
نتابع اي notification جاية منه ،
ونقدر نستخدم ال next() بشكل مباشر من خلال
ال subject وال next notification هيبعت multicasted
لكل ال subscripers .

٤- ايه الفرق بين ال subject وال behaviorsubject ؟

ال behaviorsubject هو نسخة statefull من ال
subject ، يعني بيخزن آخر value اتعملة في emit في
ال memory ، بالتالي لو عملنا subscripe جديد عال
behaviorsubject ، ال behaviorsubject هيعمل emit تلقائي
لآخر value اتعملها emit وهيبعتها لل subscriper الجديد
حيث يبقى متابع آخر update مع باقي ال subscripers .

CamScanner
الممسوحة ضوئيا بـ CamScanner