Question 1

ما هي الDirectives فى الAngular وما هي أنواعها؟
سؤال Interview مهم ولازم تتسأله مهما كان الposition اللي هتقدم عليه ان شاء الله.
-التعريف:
هي Functions تستخدم لمعالجة سلوك (behavior)الDom، ويتم تنفيذها فى الحال ما ان وجدها الCompiler.
-الأنواع:
-1الComponents directive:
هي نوع مميز من الdirectives، طب ليه؟
لأنها ببساطة بتيجي بال Templateبتاعها جاهز، المثال عليها هو ال Component@وده Class decorator و :Directiveوهو عبارة عن البنية الأساسية لبناء مشروع الAngular، وبيحوي ملف لل Styleوملف ال Tsاللي بيكون فيه Functions ومعلومات، وفي ملف تالت لل Htmlوده هو الخاص بعرض ال.Component
---
-2الstructural directives:
وده عبارة عن Directive بيغير ال Domكاملًا من حيث الشكل (Layout) والإضافة والحذف، والظهور والتكرار، من الاخر بيستبدل الDom، بناءًا عليه، وهيوضح دلوقتي لما نذكر أمثلة built in عليه وهم:
ال :ngIf*وده بيشيل جزء او يخليه ظاهر بناء على ال Booleanاللي راجعله، لو كان True يظهر لو كان False يختفي.
ال :ngFor*بيكرر حسب المطلوب منه لو في داتا راجعاله؛ فهو بينظمها حسب ما تحتاج منه؛ وده بيغير اجزاء كتير فى الDom، لاحظ النقطة دي.
ال :ngSwitchCase*برضو بتغير ال Domبناء على الحالة الي بترجعله وبيظهر حسب رغبتك Dom او يخفيه.
بإختصار ال structural directivesبتغير ال Domوبتتحكم فيه ككل، ولاحظ ان اولها.*

-3الattribute directives:
دي من اسمها تقدر تضيف Directive يتحكم فى Attribute معين، وهديك الأمثلة على طول اهو:
ال :[ngClass]ودي بتتحكم في إضافة وإزالة Class فى الHtml، حسب ال booleanاللي واصله.
ال :[ngStyle]ودي بتتحكم في إضافة وإزالة Style فى الHtml، حسب ال booleanاللي واصله.
ال :[ngModel]ودي بتتحكم في إستقبال data ب.one way data binding
ال :[(ngModel)]ودي بتتحكم في إستقبال وإرسال data ب.Two way data binding
---
نضيف سؤالين للمقال:
-1ما هي الcustom directives؟ (هكمل الإجابة فى مقال منفصل)
-2ايه الفرق بين اني اضيف Class واعمله display none وبين الngIf*؟

-1الcustom directives دي بتعملها بنفسك عن طريق انشاء directives وينفع تكون structural وينفع تكون attribute، وفي الحالتين بتستخدمهم بين [] لو attr وبتستخدم قبلهم * لو structural زي ال directivesالbuilt in. وان شاء الله فى مقال منفصل يكون شرح وافي لبناء.custom directive

-2الفرق انك لما تضيفClass ؛ فهيكون لسه موجود فى ال inspectكومنت يظهر ال domالمخفي بكل ما في داخله، لكن لما بتعمل ngIf* وال ngIf*تحذف ال domوتبدله بكومنت من عند الAngular يوضح سبب الحذف وميظهرش حاجة من ال.Dom

Question 2

ما هي الDecorators فى الAngular؟

(سؤال انترفيو مهم).

-التعريف:

الDecorators هي مفهوم أساسي فى الTypescript؛ ولأن الAngular معتمده على الTypeScript؛ فوضعته كحجر أساس فى الAngular.

هي نوع مميز من الDeclaration؛ غير انها الDesign pattern الأساسي الي بتعتمد عليه الAngular في تعريف خواص عديدة.

-خصائص:

1- بتعرفك النوع الي بتتعامل معاه، مثال:

@ Component

2- هي Functions بترجع Functions

3- بتحتوي على Metadata وهو الobject الي بيرجع فيه البيانات الخاصة بالdecorator.

4- بيشتغل الdecorator مع الruntime.

5- تساعدك انك تشغل من خلالها Function او تبعت Function او انك تبعت داتا وترجع داتا وإلخ...

ما هي أنواع الDecorators فى الAngular؟

هما 5 أنواع، فى المنشور ده هنحصر مهمتهم بشكل سريع ونذكر الأنواع؛ لأن شرح كل واحدة فيهم تدخلنا في نقاش وجدال ومقارنة، وده الي هيحصل بإذن الله فى المقالات القادمة.

1- الClass Decorator:

هو أهم نوع والأكثر إستخدامًا فيهم وهو (highest-level decorator)؛ لأنه بيوضح الغرض من الDecorator وبيوضحلك الClass الي انت شغال فيه اذا كان Component ولا Module، نذكر بعض الأمثلة سريعًا:

-الPipe، -الComponent، -الNgModule، -الDirective، -الInjectable.

(ملحوظة قبل اي decorator بنحط علامة @ مهما كان نوعه فى الخمسة).

2- الProperty decorator:

وهو تاني أشهر نوع فى الdecorators وبيساعدك تبعت Property من Component لComponent والعكس او من Directive وهكذا من الأمثلة، وأشهر اتنين فى الفئة دي Input وOutput ودول الي بينقلوا data او Functions من الاب لابن والعكس، وهديك كام مثال على الفئة دي:

-الViewChild، -والViewChildren، -والContentChild، والContentChildren.

كلهم مميزين وليهم أدوار قوية في نقل الخواص، لكن مش هيكفيهم حتى منشور لكل واحدة فيهم.

3- الaccessor decorator:

ودول بيستقبلوا parameters زي: name او target.

واشهر مثالين الGetter والSetter وبيتكتبوا كدة قبل الfunction:

بيتكتب قبل الget والset الDecorator بالشكل التالي:

@LogAccessor

} get text(): string

return this._text;

{

} set text(value: string)

this._text = value;

{

4- الMethod decorator:

ودي decorators بتعمل Method، أمثلة:

HostBinding و HostListener

واحدة منهم بتعمل بيها Function والتانية بتعمل bind لكود جاهز، وبتستخدم فى الdirective والComponent بنفس الطريقة.

5- الParameter decorator:

في مثال عليها بس قليل لما بيتم استخدامه وهو الInject؛ لأنه اتعوض بطرق آخرى، ومهمته انه يعمل inject للService داخل الconstructor بتا الComponent، بالطريقة دي:

} constructor(@Inject(MyService) myService)

console.log(myService); // MyService

{

وطبعًا الأفضل استخدام الAccess Modifiers زي الPrivate والPuplic والProtected لما ليهم من خصائص اكبر من مجرد الInject.

Question 3

كيف يعمل الAngular؟
بعد تثبيت ملف ال Angularبيتم انشاء العديد من ال Filesوال Foldersالمهمة، في المقال ده هتعرف أهمية أغلب الملفات دي (ان شاء الله).

اولًا ملف ال package jsonوده مهمته بإختصار يقولك الاسم الخاص بالبروجيكت والأوامر الأساسية + الملفات المثبتة او سواء ك devاو مثبتة بشكل عام (dev & build) وده كله تقدر تعدله، والملف ده بيعرف اي حد شغال معاك المكاتب المثبتة إلى جانب الAngular؛ فعند التحميل يكتب أمر npm i في التيرمنال، والتيرمنال يروح يقرأ ملف ال packageويحمل بناءًا عليه.

ثانيًا:Life cycle of angular project
هتسألني ايه المصطلح ده؟
-المصطلح ده بيقولك خريطة تشغيل الموقع، طبعًا الخطوات دي بعد ملف الpackage json، هتقولي ليه محطتهوش الخطوة الأولى فى الlife cycle؟
-هقولك لأن مهم ال Package jsonبتقف لحد تثبيت الملفات وبس، لكن الملف بعدين فى ال Real projectمش هتبقى محتاجه.
نبدأ بقى فى ال:life cycle
1- Angular json:
ده بقى الملف الي بيقول للبروجيكت يبدأ منين؟ ويروح منين؟ ومين يجيب مين؟ والملفات الخارجية المستخدمة وهتلاقي فيه سطرين حلوين، الي هما دول:
"index": "src/index.html",
"main": "src/main.ts",
الاول وده عشان ال Angularبتعتمد على الSingle page؛ فالأول بيقول لل Angular jsonهي دي ال Single pageبتاعتك.
2- Main ts:
وده السطر التاني لو انت مصحصح معايا، وده بيقوله نقطة البداية للبروجيكت، عن طريق Function اسمها : bootstrapModule.
وبيديله الخطوة الجاية الي معانا.AppModule :
3- AppModule:
وده ال Moduleالرئيسي فى البروجيكت، بعيد عن انك ينفع تغيره (مع ان مفيش داعي وده احسن نظام والناس الي معاك هتدعي عليك لو عملت كدة)، بس ال AppModuleبتاعنا في كام حاجة مهمة كدة:
-declarations
-imports
-providers
-bootstrap
ال Importsدي بتحط فيها ال modulesالمستخدمة، وال providersبتحط فيها ال Servicesالي عايزها تشتغل ف الmodule، بس دول مش موضوعنا.
المهم عندنا بالنسبة لل cycleال declarationsوال bootstrapودول ببساطة:
ال :declarationsبتحط فيها ال componentsالتابعين للmodules، ولما بتكون لسه بادئ البروجيكت بيكون فيها AppComponentوبس.
ال :bootstrapبتحط فيها ال componentالي هيبدأ يشغلها الmodule، ودي مهمة ال bootstrapبتاع ال Angular مش التاني بتاع ال cssالاتنين مختلفين تمامًا.

4- AppComponent:
ودي لما بتدخل عليها بتلاقي فيها دول:
selector: 'app-root',
templateUrl: './app.component.html',
styleUrls: ['./app.component.css']
الاول ده المهم فى قصتنا، بس هقولك كل واحد فيهم بيعمل ايه؟
الاول ده اسم ال Componentالي اقدر من خلاله اعملها.call
التاني وده ال Htmlبتاع ال.component
التالت ايوة شاطر ال cssبتاع ال.component
5- index html:
فاكر صاحبنا الي ف اول رحلتنا فى البروجيكت قولتلك هو ال single pageبتاعنا؟
اهو الحمدلله بعد الرحلة دي كلها وصلناله، هتلاقيه صفحة Html عادية خالص بس فيها سطر غريب كدة:
<app-root></app-root>
معنى كدة ايه؟
ايوة شاطر، هو عمل call لل AppComponentبتاعتنا الي ال Selectorبتاعها اسمهapp-root ، وبكدة هيبدأ من ال AppComponent، هتروح انت بقى لملف الAppComponent html، وتحذف الكلام الي فيه وهو ده نقطة البداية الي اول ما تنشأ بروجيكت Angular بتكون متحضرالك.
وكدة يبقى انت عرفت دورة حياة بروجيكت ال Angularمن الصفر، لحد ما وصل انه بقى بروجيكت قدامك لما بيتنشأ.

Question 4

ايه الفرق بين ال Pure pipeو الImpure pipe؟
السؤال ده لا بد تتسأله فى اي.Angular interview
وبرغم سهولة السؤال كتير بيغلطوا فيه.
هديك إجابة مفهومة ومختصرة بتوضح الفروق الجوهرية وابقى اتعمق مع نفسك:
-1ال Impure pipeبتشتغل لما يحصل تحديث(change detection) ، يعني مثلًا انا عندي:
Array = [1,2,3]
وعملت function تعمل push رقم 4 بعد ثواني.
وقولت لل pipeبتاعتي احسبي الناتج.
الpure pipe، بعد مرور الثواني هتطلعلك الناتج ب6.
الImpure pipe، بعد مرور الثواني هتطلعلك الناتج ب10.
وده لأن ال Impure pipeبتشتغل بطريقة Async وبتعرف ال change detectionالي حصل) وده عند اي تغيير، ممكن click او زرار كيبورد مش شرط المثال ده بس.(
على عكس ال pureمش بتعمل detection لل change الي حصل.
-2طيب ايه الفرق فى الكود بين ال pureوالimpure؟
فى حالة ال impureهتضيف متغير لل decorator pipeالي عندك اسمه:
pureوتديهFalse ، بالشكل ده:

```
@Pipe({
name: 'sumPipe',
pure: false
})
```

وفى حالة ال pureيكون true او متضيفهوش؛ لأنه by default بيكون.pure pipe
-3من حيث المشاركة:
ال pure pipeينفع تشاركها؛ لأنها بتأدي دور ثابت مفيهوش متغيرات، على عكس ال impureمينفعش تشاركها؛ لأنك ممكن تأثر عليها، وبالتالي بتغير من قيمتها.
-4أمثلة لل pureوال impureفى ال:bulit in pipes
-1ال:pure
TitleCasePipe
DatePipe
UpperCasePipe
-2ال:impure
JsonPipe
AsyncPipe
SlicePipe
وبكدة اكيد عرفت ان ال impure pipesمهمتها الأساسية هي معرفة أخر التغييرات الي تمت، وده يعني انها بتتبع استراتيجية اسمها:
#ChangeDetectionStrategy
والاستراتيجية دي تعني: اي تغيير يحدث اتفاعل معاه.

Question 5

ما هي الفروق بين ال Promisesوال Observablesفى الAngular؟
)سؤال متكرر فى انترفيو ال.(Angular
-المصدر:
ال Promiseجات مع ال.Es06
ال :Observableمن مكتبة ال.Rxjs

المقارنة:
-1ال:Call back
ال promiseليها حالتين اما Success واما يحصل.Error
ال observableليها 3 حالات: اما Success او يحصل Error وبعدهم.Complete
---
-2ال:Lazy loading
ال :promiseبتكون not lazy والجملة دي مساوية لجملة.(Eager loading)
ال :observableبتكون.lazy

-3ال:Emits
ال :Lpromiseبتعمل Emit ل Valueواحدة.
ال :observableبتعمل Emit لأكثر من:Value
وده بيحثل عن طريق الnext، مثال:
observer.next(1)
observer.next(2)

-4ال) cancellableالإغلاق:(
ال :promiseبتكون.not cancellable
ال :observableقابلة للإغلاق(cancellable) ، وده عن طريق ال.()unsubscribe
---
-5التزامن:
ال :promiseدايمًا) ASYNCHRONOUS غير متزامن.(
ال :observableممكن تكون ASYNCHRONOUS وممكن SYNCHRONOUS يعني متزامن وغير متزامن.

-6ال:rx oberators
ال :promiseلا تقبل ال.rx oberators
ال :observableيقبل الrx oberators، عن طريق ال pipeوبالشكل التالي:
Obs.pipe(map(value) => value * 2);

Question 6

ترتيب ال Life cycle componentفى الAngular؟
من الاسئلة المشهورة فى الانترفيو، ولكن بالتأكيد هيسألك على الاقل عن نقطة من نقاطه، والي غالبًا بيكون عن الأربعة دول:
AfterViewInit, OnDestroy, Constructor, Oninit، بس ده مش موضوع مقالنا، موضوع مقالنا هو الترتيب، وبإذن الله هنشرح كل hook منهم فى مقال لوحدها، مقالنا هو مقدمة عن الlife cycle، ونبذة بسيطة عن كل hook.
الترتيب بيكون كالتالي:
1-ال:constructor
وده بيشتغل اول حاجة، وبنحتاجه؛ عشان ن injectمن خلاله ال servicesونعطي قيم أساسية بداخله بدون الحاجة لعلامة ال.(!)non null assertion
---
2-ال:ngOnChanges
وده اول حاجة بتشتغل بعد الconstructor، وبيشتغل لما يحصل تغير فى ال propertiesالخاصة بال.@Input
---
3-ال:ngOnInit
وده بيشتغل مرة واحدة، وبيهيء ال componentوبيعمل Set لكل الأساسيات الي بتحتاجها من الcomponent؛ فمثلًا لو عندك data راجعة منService ؛ فبيتعملها Set فى ال hookده وبتساوي من خلالها Object او Array بال dataدي.

4-ال:ngDoCheck
وده بيتشغل لما يحس بأي تغيير فى الcomponent، وتقدر من خلاله تستخدم خوارزمية ال.change detection
---
5-ال:ngAfterContentInit
وده بيشتغل بعد ما ال contentيبدأ يشتغل فى ال viewيعني يتعمله render فى ال viewوليها تعبير تاني اسمه (projected).

6-ال:ngAfterContentChecked
وده بيشتغل فى حالتين:
-الأولى: لما ال contentيحصل فيه تغييرات.
-الثانية: لما ال ngDoCheckيشتغل.
وممكن تقول ان الحالتين حالة واحدة؛ لأنه لما بيحصل تغيير فى الcontent؛ فالأول بيشتغل الngDoCheck، وبعدها يشتغل ال hookالي بنتكلم عنه.

7-ال:ngAfterViewInit
وده بيشتغل لما ال Viewيتعملهbuilt ، او لما تعمل built ل Child componentخاصة بال Componentبتاعتك.

8-ال:ngAfterViewChecked
وده مقارب لل hookرقم 6 وبيشتغل لما ال viewيحصل فيه تغييرات وبعد ال.ngDoCheck
---

9- ال:ngOnDestroy
وده بيشتغل لما تسيب الcomponent، وترجع أهميته لأنه بيعمل destroy للداتا لما تروح ل componentتانية، وبالتالي الابليكشن هيكون أسرع بكتير، وأخف.

ملحوظة1) :ده الترتيب للlife cycle، لكن مش بيشتغلوا فى سلسلة متتاليةStatic ، بيشتغلوا بطريقة مختلفة، هحط صورة ليها فى الكومنتس.(
ملحوظة2) :وقت الإجابة يفضل تبدل مصطلح البناء او التهيئة بكلمةinitialize ، عشان إجابتك تكون فورمال اكتر.(

Question 7

ما هو الفرق بين ال Template driven formsوال Reactive Formsفى الAngular؟
الاول لازم تعرف ان في انواع من ال Formsفى ال:Angular
-1ال basic forms.
-2ال Template driven forms.
-3ال Reactive Froms.
-4ال Dynamic Forms.
وفى المقال ده مقارنة بين الTemplate driven، والReactive، والأربعة هيتشرحوا بالتفصيل فى مقالات منفصلة ان شاء الله، ندخل فى المقارنة على طول:
--
-1ال Setup:
-ال Template: بتم من اسمها بتم من خلال التمبلت Html عن طريق ال directive.
-ال Reactive: بتم من خلال ال Class componentفى ملف ال ts.

-2الإستخدام:
-ال template: سهل الإستخدام.
-ال Reactive: محتاج تدريب.

-3ال) Use caseضرورة الإستخدام:(
-ال template: للسيناريوهات الصغيرة.
-ال Reactive: للسيناريوهات المعقدة.

-4ال Data flow:
-ال Template: بيكون Asynchronous.
-ال Reactive: على العكس بيكون Synchronous.

-5ال Validation:
-ال Template: من خلال ال Directives.
-ال Reactive: من خلال ال Functionsفى ملف ال ts.

-6شكل الكود:
-ال Template: ال Html اكتر وقليل Ts.
-ال Reactive: ال Html قليل ال Ts كتير.

-7ال testing:
-ال Template: بيكون أصعب.
-ال Reactive: بيكون أسهل.

-8ال Data binding:

-ال Template: بيكون Two way data binding بيعتمد على ال[(NgModel)]، يعني بيكون.(mutable)
-ال Reactive: مفيش data binding يعني.(immutable)

Question 8

-ما هو ال view encapsulationفى ال Angularوبيعمل encapsulate لايه وازاي وانواعه؟
-ما هو التغليف فى ال Angularوبيغلف ايه وازاي وانواعه؟
)من أسهل اسئلة الانترفيو بس يبان صعب، وموجود فى اغلب الانترفيوهات.(
التغليف او view encapsulation ده عبارة عن specific attribute بيتضاف؛ عشان لو هتعمل style لحاجة معينة؛ بحيث انها متأثرش ف مكان تاني يعني مثلًا لو انت عندك component وفيها عنصر او class فأديته Style مينفعش تلاقي مثلًا نفس العنصر فى component تانية واخدة نفس الStyle؛ فال Angularتلقائي بتعمل عملية التغليف فمثلًا بدل ما العنصر يبقى h3 فى ال stylesيبقى[_ngcontent_yax-c5]h3 ، ونفس التغليف يكون على عنصر ال Htmlبالشكل ده<h3 _ngcontent_yax-c5> ؛ فبيخليه معزول ومش بيشارك ال styleبتاعه مع حد.

وال view encapsulationله 3 انواع:
-1وده ال defaultوالي اتشرح فوق:(Emulated)
شوف شرحه فوق + طريقة كتابته بتكون ف ال decoratorاللي اسمه @Component بتضيف property او metadata:
encapsulation: ViewEncapsulation.Emulated

-2ال:None
وده بيتكتب كدة:
encapsulation: ViewEncapsulation.None
وده معناه انك متعملش تغليف لأي حاجة فى الComponent؛ فمعنى كدة لو عندك مثلًا h3 مديه لون؛ فأي h3 هياخد اللون ده ك defaultطالما مفيش تأثير عليه من ال compnentالتانية.

-3ال:ShadowDom
وبيتكتب كدة:
encapsulation: ViewEncapsulation. ShadowDom
بيعملShadowRoot ، وده بيغلف ال componentبالكامل؛ فبالتالي مش بتتأثر بالglobal style، يعني مثلًا لو عندك style فى الملف ال(gloabal (style.css؛ فمش هيأثر فيه ولكن لو في component تانية ال ViewEncapsulationبتاعها بNone؛ فهتأثر فيها عادي.

Question 9

ايه الفرق بين ال Ahead Of Time(AOT)وال Just In Time(JIT)؟
)سؤال انترفيو(
-هما الاتنين عبارة عن compiler بيحولوا الكود من شكل لشكل او من حالة لحالة.
المقارنة:
-1ال:compile
ال :AOTبتعمل compile وقت بناء البروجيكت داخل الIDE، مش بتحتاج انها تعمل compile فى ال.runtime
ال :JITبتعمل compile لما البروجيكت يروح لل browserقبل ما يتعرض، وبتحول ال TypeScriptلJavaScript؛ لأن ال browserمبيفهمش ال.TypeScript
---
-2السرعة:
ال :AOTاسرع؛ لأنه بيعمل compile وقت كتابة الكود؛ فمش بيحتاج وقت.
ال :JITابطأ؛ لأنه بيعمل compile وقت ال.runtime

-3ال modeالمفضل:
ال :AOTيفضل في ال.Production mode
ال :JITيفضل في ال.development mode

-4ال:command to run
ال:AOT
ng build --aot OR ng serve --aot
ال:JIT
ng build OR ng serve

-5اكتشاف الاخطاء:
ال:AOT
-بمجرد الحفظ، داخل الIDE؛ وبيوقف البروجيكت لما يلاقي خطأ ويطلعلك ايرور فى ال Command lineوال IDE وال.browser
ال:JIT
-عند استخدام الخطأ فى الbrowser، يكشف الخطأ فى ال.console
======
ال Angularبالوضع الإفتراضي يستخدم الاثنين:
ال Aotبيعمل compile للكود داخل ال IDEوبعدين يبعته لل JITيعمله compile ويحول ال TypeScript لـ.javaScript
ممكن تعمل لل Aotداخل ملف ال Angular jsonإيقاف، ووقتها لو عندك ايرور فى البروجيكت هيسيب البروجيكت يشتغل عادي، لكن لما تستعمل حاجة مش شغالة هيطلعلك ايرور فى ال.console

Question 10

ما هو ال lazy loading فى الAngular؟
)سؤال انترفيو(
-1المصطلح:
ال Lazy loadingهو طريقة(Pattern) ، بتستخدمه ال Angularوالعديد من ال Frameworksالأخرى وأغلب لغات البرمجة، الغرض منه هو زيادة سرعة ال Applicationوالوصول ل performanceأفضل.

-2مع ال:Angular
ال Angularبتستخدمه بطريقة ما للحد من عدد ال modulesالي بتتحمل لما ال userيحتاجها، بالتالي الابليكشن يكون اسرع وال performanceأسرع وال Intial bundle sizeيكون أقل؛ فميحصلش بطأ لما تفتح الابليكشن.

-3ال:Eger loading
وال lazy loadingعلى عكس نظام ال Eger loadingتمامًا الي بيحمل كل ال modulesال Egerبمجرد دخولك للابليكشن؛ عشان كدة ال lazy loadingجاه حل المشكلة دي بأنه يحمل ال modulesال lazyعند طلبها.

-4طريقة عمله (ككود):
بيستخدمloadChildren ، وال loadChildrenبتستخدم توقع يرجع Function فيهاPromise ؛ بحيث انه يحمل ال moduleفقط عند الدخول على component من ال componentsالخاصة بيه، الكود للتوضيح:
{ path: 'profile', loadChildren: () => import('./profile/profile.module').then(m => m.ProfileModule) },
وال :ProfileModuleبيحتوى على ال componentsالخاصة به مستدعاه بداخله وبداخل ال Router module الخاص بيه.

-5التأكد من عمل ال:lazy loading
عند فتح الapplication، لن تجد اي من محتويات ال componentsالتابعة لل lazy moduleوجود فى ال network الخاصة بك، لكن عند الدخول الى ال pathالخاص بالlazy module؛ فمثلًا في مثالنا عند الدخول إلى: yourPort/profile؛ ستجد ان الملفات الخاصة بال moduleتسارع ف الظهور.

-6لم تصلك الفكرة؟
لديك الحق؛ لأننا هنا نتحدث عن module واحد فقط، لكن تخيل ان لديك ابليكشن يتسع لاكثر من 20 module او حتى module4 ؛ فبالتأكيد ستكون بحاجة ماسة، لمثل هذا الحل الرائع؛ فبتحويل جميع ال modulesإلىlazy ؛ سيكون الأمر أكثر سرعة وأقل حجمًا في ال.initial bundle size

Question 11

ما الفرق بين ال Observableوال subject؟
)سؤال انترفيو(
-1الاختلاف الرئيسي:
ال observable:
بياخد observer وبيتعامل معاه كأنه هو، يعني مثلًا:

```
let myObservable = new Observable<any>(observer => {
observer.next('Something');
});
```

ال subject:
بيتعامل بشكل مباشر يعني مثلًا:

```
let mySubject = new Subject();
mySubject.next("something");
```

-2الاختلاف الثاني:(Hot vs cold)
ال Observableهو من النوع:cold
يعني لازم يتعملهsubscribe ؛ عشان كل الي اتكتب فيه قبل كدة يشغله الابليكشن، مثال:

```
let observable = Observable.create(observer => {
observer.next('first');
observer.next('second');
...
});
observable.map(x => ...).filter(x => ...).subscribe(x => ...)
```

لاحظ ال subscribeجات فى الاخر.
--
ال Subjectمن نوعhot ؛ فهو المطلوب انك تعمل subscribe فى الاول، واي جديد هيتضافله عادي، مثال:

```
let source = new Subject();
source.map(x => ...).filter(x => ...).subscribe(x => ...)
source.next('first')
source.next('second')
```

لاحظ انك ضيفت لل Subjectعن طريق ال nextبعد ما اتعمله subscribe عادي.

-3الاختلاف الثالث:(UNICAST VS MULTICAST)
ال Observable:
بتكونUnicast ، يعني ايه؟
يعني مثلًا لو انت عندك متغيرين وعايز تبعتلهم ارقام عشوائية؛ فلما تعمل subscribe لكل واحدة بحيث انه يغير قيمة الاتنين؛ فهو بيرجع قيمة مختلفة لكل متغير، مثلًا: 9|5.
ال Subject:
على العكس تمامًا بيكون:multitask
يعني لو معاك متغيرات كتير هيرجعلهم هما الاتنين نفس القيمة مثلًا:

8|8|8|8
لحد هنا تمام، بس لو عايز تعرف السبب؛ فده لأن برضو ال Subjectبيكون hot وبيعمل عملية ال nextبعد ال subscribe، على عكس ال observableبيبعت الداتا قبل ما يعمل.subscribe
---

https://www.youtube.com/watch?v=RcHeXE1OqLs

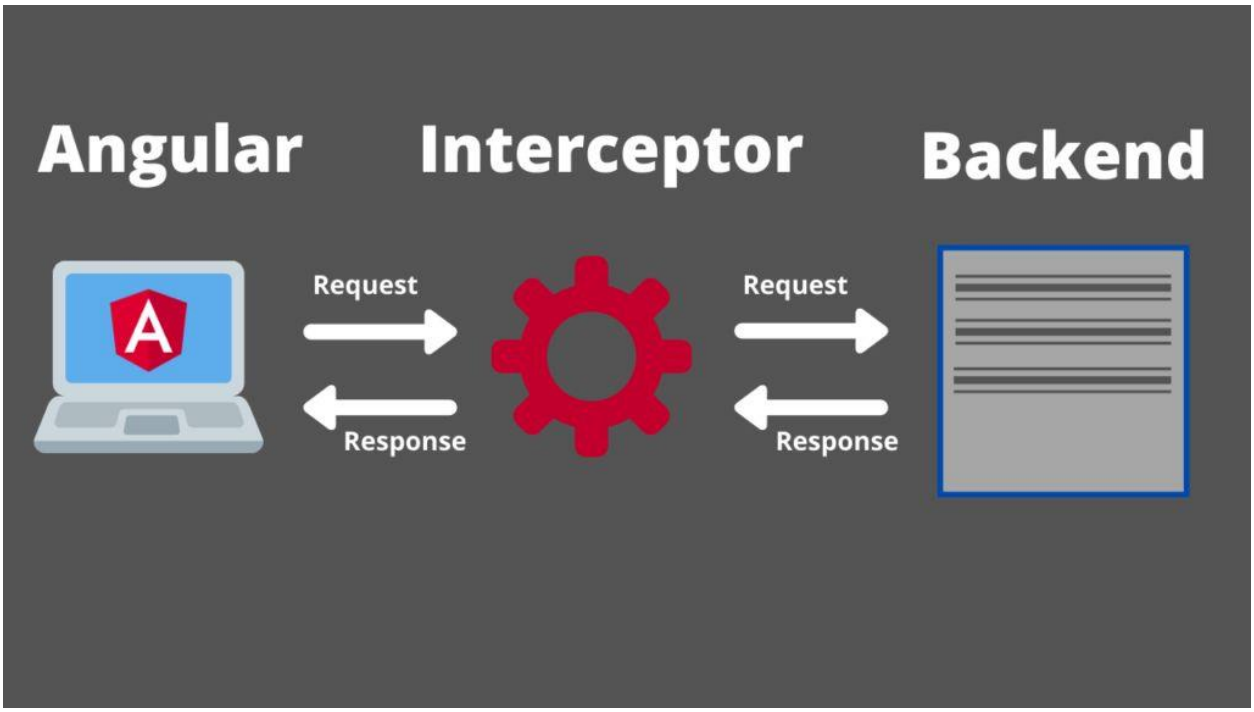
Question 12

ما هو الAngular interceptor؟
)سؤال انترفيو.(
قبل ما اقولك ايه هو ال interceptorومعناه العربي (المعترض)، خليني اذكرك بمعلومة ان ال frontendعندنا مجردUi ؛ فهو لازم يعمل عمليات تخليه يتواصل مع الbackend؛ عشان يقدر يعرض الداتا والعمليات دي كلها تتلخص فى عمليتين Request) و(Responseهو ان ال Requestاو ال Frontهو ان ال Uiايطلب من ال backendالداتا، ال Responseهو ان ال backendيجاوب على الطلب بتاع ال frontendويبعتله المطلوب.

فال Angularقالت يعني ايه يتواصلوا مع بعض كدة على طول؟ احنا Framework محترم مفيش ظابط مفيش رابط؟ راحوا جابوا ال Interceptorاو المعترض، وده يا سيدي قاطع الطريق فى الAngular، رضيته هتعدي وتكمل مهمتك، مرضيتهوش وحصلت مشاكل، طلبك هيترفض) ال.(Request
---
نجاوب سؤال الانترفيو بقى:
لو المنصب الي مقدم عليه جونيور؛ فممكن تقول التعريف فقط، انه:
بيكون medium بين ال backendوالfrontend؛ فعند عمل request بيعترضه ويتعامل معاه ويبعته لل backend ويرجع بال.response
لكن لو المنصب أعلى لازم تذكر شوية أمثلة حلوين، ولازم تبقى عارف ان ال interceptorمش بيهندل او بيعمل التواصل بنفسه؛ هو بس بيعملهperform ؛ عشان يعمل المطلوب منه بالظبط.

تعالى نشوف إستخدامات صاحبنا ده، هو له العديد من الإستخدامات، بس أشهرهم 10:
-1ال :loadersالي بيحمل قبل الصفحة.
--
-2تعمل.Changing the URLs
--
-3تعمل check للErrors؛ عشان يعرف ال APIشغالة ولا لا) يعني نفس طريقة ال (Ajaxالي هي 400 و500 و200، هارش؟
--
-4تعمل Notifications ويبعتهالك فى ال consoleولا بأي طريقة انت تقوله يبعتهالك؛ عشان مثلًا يقولك يا باشا ال Apiاشتغلت ولا ال Apiمش لاقيها.
--
-5تعملFake backend ، يعني انت مثلًا معندكش backend تفضل كدة من غير ما تجرب ال connectingوالكود بتاعك؟ يرضيك ده مثلًا؟ فأنت تقدر تعمل جواه Fake backend وتعمل تيست براحتك للصبح.
--
-6ال :Authenticationتعمل check من خلاله على ال tokenعلى الUser، عيش براحتك.
--
-7تعمل.Headers
--
-8تقدر تعمل بيه Converting من حاجة لحاجة، مثلًا تعمل convert لملف XML وتخليه.json
--
-9تقدر تعمل ال :Profilingيطلع ايه ده؟ ركز معايا عشان ده مهم، ال Profilingمن الاخر يعني تزامن للعمليات،

يعني لو انت عملت اتنين request ورا بعض او سجلت بأكونتين لنفس الdatabase؛ فهو بسبب التزامن مش بيتأخر في انه يعرف ان دول طلبين لنفس الحاجة مش طلبين مختلفين، من الاخر.(without any time delay)
--
-10تعمل :Catching وده مختلف شوية عن الcheck errors؛ لأنه لما بتعمل Catching فأنت بتتأكد من عملية الGetنفسها وعلى أساسها بتهندل بال operatorsالمتاحة بتاعت الRxjs.



Question 13

ما هو الcomponentفى الAngular؟

--

-1التعريف:

الComponentهو اللبنة الأساسية لواجهة المستخدم (UI) في تطبيق الAngular، يحتوي تطبيق الAngular على شجرة (Tree) من الComponents.

--

-2النوع:

هي مجموعة فرعية من الDerctive، يعني نوع خاص منها؛ لأنه دايمًا مرتبط بقالب، والنوع ده فى الDirective اسمهComponent directive ،)هتلاقي عندي مقال بيشرح الdirectivesبأنواعها.(

--

-3متطلبات:

الComponentالعادية لازم تنتمي لNgModule؛ عشان يكون مسموح لComponentتاني انه يستخدمه؛ فلازم يكون موجود فى الdeclarationsبتاعتModule ، والنوع الغير عادي هو الStandalone componentوالنوع ده ظهر فى Angular 14 كFeature preview،)هتلاقي عندي مقال بيشرح الStandalone component.(

--

-4الLife cycle:

وطبعًا لإن الComponentهو فى الأصلDirective ؛ فهو له Lifecycle Hooks و Constructorالي بيتضاف فيها الServicesبتاعتك،)هتلاقي عندي مقال بيشرح الLife cycle.(

--

-5مكونات الComponent:

بيتكون من 4 ملفات (ممكن تختصرهم فى ملف واحد):

-1الTs fileوده اهم ملف فى الComponent، وده لإنه بيتكتب فيه الdecoratorالخاص بيه@Component ، وبداخله بيتكتب شوية meta data بتوضح اسم الComponentوالملفات الباقية المعتمد عليها، وكمان هو الControllerبتاع الComponentيتكتب بداخله الFunctionsوالLife cycleوالConstructorالي بيتضافلها الServicesالمساعدة، وفي اشياء اكثر زي الchangeDetectionوغيرها لكن المقال مش هيكفي لذكر الكل.

-2الHTML file:

وده الViewاو الTemplate، وبيتضافله كود الHTMLوالDirectivesالمساعدة زي الngForوngIf، ويتعمل فيه Bindingلل dataالموجودة فى الTs fileغير طبعًا الPipes، والملف ده بيتعرف داخل الTs fileبtemplateUrl، وبيتحط الpathبتاعه او يمكن الاستغناء عن الملف بالكامل وتعمله داخل الtsعن طريق كلمةtemplate ، وده لا يفضل إلا في حالة ان الcomponentصغيرة، بحيث انك تضيف حاجتك داخلها.

-3الStyle file:

بيكون موجود فيه كود Css او Scss او Sass او Less حسب ما انت اخترت تشتغل بيه، ومش محتاج انك تعمل Compileلأي نوع؛ لأن الFrameworkبيعمل الcompileبنفسه، والملف ده بيتعرف داخل الTs fileبstyleUrls وخلي بالك انك تقدر تضيف اكتر من ملفStyle ، وبيتحط الpathبتاعه او يمكن الاستغناء عن الملف بالكامل زي الHtmlبس هيبقى style مش.Template

-4ملف الTest:

وده بيكون ملف Test داخل الAngularوبيشتغل بkarmaوJasminفى الوضع الإفتراضي.

لحد هنا الكلام مخلصش بخصوص الAngular component، يعني المقال ده فكرة توضيحية سريعة عنه.

Question 14

ما هي الPipes فى الAngular؟
)المقال يتضمن سؤال انترفيو(
-1التعريف:
الPipes يعني انابيب ومن الاخر دي مسؤولة عن انها تعمل Format للdata داخل التمبلت بمساعدة الtemplate binding، وهي بتحول الdata من صورة داتا عادية للصورة الي احنا عايزينها، من الاخر دخول الpipe مش زي خروجه.

-2طريقة الكتابة:
زي ما قولنا فى التعريف بمساعدة الtemplate binding؛ فبتكون بالشكل ده:
{{ data | myPipe }}

-3بعض المصطلحات وشرحها:
-ال:built-in pipes
دي الpipes الموجودة فى الAngular(مش بتدخل في بناءها)؛ وهتلاقيها كلها فى الDocumentation بتاع الAngular من اللينك ده https://lnkd.in/dnEqiVwZ :
--
-ال:custom pipe
دي الpipe الي بتبنيها بنفسك من الصفر وبتعمل عملية مش موجودة فى الbuilt in pipes مثلًا لو حابب تعمل pipe للخصم او كدة.
--
-ال:impure pipes
الpipes دي شرحتها فى مقال قبل كدة مقارنة بينها وبين الpure pipes، بس كشرح سريع لفكرتها؛ فهي Pipes غير متزامنة(Async) ، يعني بتأدي دورها بعد ما كل العمليات المتزامنة على الdata ما تخلص، وفي منها built-in pipes وcustom pipes، المقال هتلاقي فيه شرح اكتر.
--
-Parameterized Pipes:
بإختصار دي pipes بتاخد params ، مثال:
{{ today | date:'fullDate' }}.
--
-Chaining Pipes:
وده مصطلح معناه انك تقدر تعمل اكتر من pipe لنفس الحاجة، مثلًا:
{{ today | date:'fullDate' | uppercase}}.

Question 15

ما هو الAgile؟
-التعريف:
ال Agileهي منهجية لتطوير البرمجيات، تنبع من فكرة تقسيم كميات كبيرة من العمل إلى اجزاء صغيرة، يمنح هذا مديري المنتجات والمطورين وأي صاحب مصلحة فهمًا أفضل للعمل المراد ويزيد من قيمته، يختص في بناء المشاريع المبهمة والتي يوجد بها مخاطر.
)ال Agileهي.(Mindset
-سبب وجود المنهجية:
قديمًا، كان تطوير البرمجيات عملية بطيئة؛ حيث يمكن للتغييرات الرئيسية في المتطلبات أن تضع ضغوطًا ومخاطر كبيرة على الفرق.
-الحل الذي قدمته:Agile
عند اتباع منهجيةAgile ، تساعد الأجزاء الأصغر من العمل الفرق على أن تصبح أكثر مرونة؛ فهي تساعدهم في هذه العملية على تقديم الميزات بشكل أسرع والاستجابة للتغييرات بشكل أسرع.

أفكار ال Agileاتقسمت لاكتر من نوع من أطر العمل(frameworks) ، لكن أشهرهم اتنين Scrum :وKanban.
وبإختصار عمل أطر العمل بيعتمد على مهام متكررة مبنية على منهجية ال.Agile
---
وبشرح سريع لواحد منهم هتفهم عملية ال Agileبتم ازاي؛ فتعالى نشوف ال Scrumبيشتغل ازاي، وقبله لازم تكون عارف مصطلح ال.Sprint
ال:Sprint
فترة زمنية قصيرة ومحددة زمنيًا عندما يعمل الفريق لإكمال قدر معين من العمل،وهي صميم منهجيات Scrum ورشيقة، وبتساعد ال Sprintsالصحيحة فريقك المرن على شحن برامج أفضل مع عدد أقل من الصداع.
ولازم طبعًا الاول تتأكد إن الفريق محدد وفاهم هدف ال sprintوازاي يقيس نجاحه، ده هو الاساس للحفاظ على توافق الجميع والمضي قدمًا نحو وجهة مشتركة.

ال:Scrum
هو إطار عمل يمكن للناس من خلاله معالجة مشاكل التكيف المعقدة، مع تقديم منتجات ذات أعلى قيمة ممكنة بشكل منتج وخلاق.
وبيخضع لأربع عمليات رئيسية:
-1ال Product ownerيبعت طلب التحسينات في صورة.Backlog
-2ال Teamياخد ال Backlogويحولها ل Sprintsاو يرجعها فى صورة.Sprints
-3يقوم ال Teamوصاحب المصلحة بفحص النتائج ويعرفوا اذا كانت متوافقة مع بعضها او لا، وبعدها يتعمل Sprintsتانية.
-4تعاد العملية لحين انتهاء العمل بالكامل.
والعملية بأكملها بتخضع ل Scrum masterبيراجع العملية وبيشرح للكل دوره والقواعد الي يمشي عليها، وهو ال Software leaderل Teamال.Scrum
وطبعًا بعد ده كله يخضع البروجيكت لتجربة المستخدمين والشكاوى ويتم تطوير البروجيكت بناء عليه.
وده Framework من ال Frameworksالمطبقة لنظرية ال.Agile

طبعًا المقال مش أحسن حاجة لشرح عملية الAgile، لازم تدخل فيها بشكل عملي؛ عشان تفهمها وتتعمق فيها أكتر وتتعود عليها، غير انك ضروري بالتأكيد هتحتاج لكورس يشرح العملية بشكل أفضل.

Question 16

ما هي الAngular building blocks؟
)يتضمن العديد من اسئلة الانترفيو(
المقال ده كوكتيل، بديك ملخص سريع عن دور الbuilding blocks فى الAngular، وخلي بالك ده دورهم مش تعريفهم، التعريف حاجة تانية وهعمل مقال منفصل عنه بإذن الله.

ال:building blocks
يعني ال blocksالمكونة للبروجيكت بتاعنا، لو هنفترض ان البروجيكت عباراة عن ماتش كورة فكل واحد من ال building blocksبيأدي دور مختلف عن التاني يساعد في تكوين البروجيكت، وفى المقال ده جمعتلك اهم ال blocksالمهمة لعمل بروجيكت.Angular
---
1-ال:Modules
ال moduleهو كتلة من التعليمات البرمجية، لها مسار عمل واحد ومحدد، اي تطبيق Angular لازم يكون في moduleواحد على الاقل وهو الroot module، بس أغلب البروجيكتس ليها اكتر من.module
--
2-ال:Components
هو ال building blockالرئيسي ومش بس فى الAngular، لا في اي Frontend Frameworks ؛ لإن كل ال Frontend Frameworksانشأت لبناء SPA appliction ، والي بيخدم العملية دي هو الComponent، ومهمته بتكون هي العرض(View) ؛ بحيث ان كل عرض بيعرضه بيكون جزء من الشاشة، وطبعًا اي تطبيق Angular لازم يكون فيه على الاقل component وهو ال app rootوبيتعمله bootstrap فى الroot module، ال Component قبل كدة شرحت مكوناته وبإختصار هما تلاتة مهمين فى عملية ال:dev
ال classوده بيكون فيه كل العمليات.
ال Viewوده بيكون التمبلت ال.Ui
ال Styleوبيكون فيه اكواد ال.Style
--
3-ال:Data binding
نعم، هو جزء من ال Componentلكنه Building block ؛ لأنه من أساسيات ال Angularان يحصل Data binding بين ال Tsوال Html، وال blockده معناه الربط بينهم مش اكتر.
--
4-ال:Directives
بتعدل سلوك ال Domبالإعتماد على طبيعتها الديناميكية وتم شرحها في مقال مسبق، وليها اكثر من نوع، يفضل تشوف المقال بنفسك https://lnkd.in/dNUHZ6Zx :
--
5-ال:Metadata
ال Classesبيكون ليها Metadata يعني بيانات وصفية توضح الغرض منها؛ عشان تساعد ال Decoratorفي فهم الغرض من الClass، لإتمام عملية ال.Processing
--
6-ال:Dependency injection
ده عبارة عن Design pattern بتستخدمه ال Angularعشان يقدملك ال dependenciesالضرورية للإستخدام مع الComponent؛ فمثلًا بتستدعي فى ال Constructorبتاع ال componentبتاعتك ال Servicesوال Modules

المطلوبة، زي كدة مثلًا:
constructor(private router:Router, private myService: MyService)
وبالشكل ده تقدر تستخدم istance من ال moduleاو ال serviceداخل ال classبالكامل، يعني داخل وخارج ال constructor، لكن لو من غير كلمة private يبقى داخل ال constructorبس.
--
7-ال:Services
بستغرب من الخلط الكبير الي بيحصل بين وبين الDependency injection؛ وده لأنه بيشتغل فى ال component بالDI، بس ال Servicesعبارة عن شوية Functions بتلعب دور مهم فى ال Applicationوتقدر تقدر تاخد instanceمنها عن طريق الDi، واكثرية تعامل ال Servicesبيكون مع ال APIs.
--
8-ال:Routing
وده Topic كبير جدًا ممكن تختصره فى جملة) بيختصر روابط ال Urlالشكل انت تحدده(، بس طبعًا له حوارات وتفاصيل، واتكلمت عن شوية تفاصيل فى الاكونت بتاعي.
--
9-ال:Template
هو جزء من ال Componentومرتبط بال componentدايمًا، وبيساعدك في بناء ال Uiالخاص بيه، وبتستخدم بداخله ال Directivesوال Pipeلمعالجة سلوك ال Domوالبيانات (Data) بداخله.

Question 17

ازاي تعمل Share ل dataبين اتنين components او directives فى الAngular؟
)سؤال انترفيو مهم.(
السؤال ده من الاسئلة الي بتوضح الفهم؛ فلازم تركز وتبقى عارف تبدأ الإجابة منين وبأحسن شكل.

اول حاجة لازم تعرف مفهوم ال Parentوال Childفى الAngular، وهو ان التمبلت بتاع ال parentبيكون جزء من مكوناته الchild، يعني افترض ان الكود ده داخل ال:parent

```
<h1>
I am parent component
</h1>
<app-child></app-child>
```

الإجابة بشكل مباشر:
-ال shareمن ال parentلل:child
-1له طريقة واحدة وهي بال Input@وده.property decorator

-ال shareمن ال childلل parentوده له طريقتين:
-2الأولى: وهي من خلال ال Output@وده property decorator مع طبعًا ال.EventEmitter
-3التانية: عن طريق ال ViewChid@وده برضو.property decorator

-4ال Shareبين اتنين component ملهمش علاقة ببعض:
وده بيتم عن طريق ال.Services

والأربعة دول هم إجابة السؤال، وإن شاء الله إجابة كافية، لكن لازم تكون اتدربت عليهم؛ فأنصحك تشوف Tutorialsعنهم لإن ممكن تتسأل فى شوية تفاصيل أكتر.
وطبعًا ممكن تعمل share لل dataعن طريق ال state managementزي ال ngrxبس ده حوار لوحده.

Question 18

ما هو الMulti-routerفى الAngular؟
ازاي تعمل اكتر من router-outlet فى نفس الcomponentوشكله بيكون ازاي ومنهم واحد بيعرض componentsمختلفة بس داخل الcomponentاللي بيعرضها الاول؟
)ده واحد من أهم اسئلة الانترفيو ل2022، بل وبيكون فاصل فى القبول، رغم انه ان شاء الله هيكون سهل، لما نجاوبه سوا.(

1-الPrimary router outlet:
تعالى معايا فى جولة بسيطة داخل الrouter-outletيعني(behind the scenes) ، الrouter outletفى الوضع الإفتراضي بيتخلقله الnameده بيكون اسمه.primary
ايه لازمة المعلومة دي؟ هقولك.

2-الSecondary router outlet:
بيكون router outlet بس انت بتحدد الاسم بتاعه، بالطريقة دي:
<router-outlet name="sidebar"></router-outlet>
وتيجي داخل الroutingبتاعك وتضيفه بالشكل ده للcomponentsاللي عايزه يعرضها:
{ path: 'chat', component: ChatComponent, outlet: 'sidebar' }
لاحظت كلمةoutlet ؛ اهي دي بتقوله ان الrouter-outletاللي تسمعله هو الي اسمه.sidebar
---
3-شكل اللينك Url عشان توصل لcomponentمن الSecondary:
localhost:4200/home(sidebar:chat)
تعالى نفصصه:
الlocalhostبالأرقام 4200 ده الdomainاللي انت شغال عليه.
الhomeدي الcomponentاللي فيها الSecondary.
لاحظت حاجة غريبة صح؟(sidebar:chat) ، دي بقى عشان ده Secondary router ؛ او انه routerمعمول customللإسم بتاعه؛ فلازم يظهر كدة.

4-طريقة كتابته كrouter-link:
طبعًا انت كواحد عاملpath ؛ فلازم تعمله button بيودي عليه، وبيتكتب بالطريقة دي مع الbutton:
<a [routerLink]="[{ outlets: { primary: 'home', sidebar: 'chat' } }]">Chat
لاحظ انه اداله الprimaryبتاعه، والSecondary، وطبعًا هتلاقي اكتر من شكل لكتابة الrouterLink، بس كتبتلك الأفضل بالنسبالي.

Question 19

ما هي الفروق بين ال Interfacesوال Model classes؟
)سؤال انترفيو(

بصراحة في المقال ده، انا جاي اقولك كبداية ازاي تبطل تستخدم any او تقلل إستخدامها بنسبة 90% وبعدين نبدأ في مقالنا، ال Apiالي انت هتتعامل معاها هتكون مبعوتالكJson ، ازاي تحولها ل InterfaceاوModel ؟
بص في طريقتين والطريقتين ساهلين:
الاولى انك تعملها يدوي، والتانية انك تدخل ع موقع Transform tools وتختار json to typescript وتديه json هيطلعلك.Interface
---
تعالى نبدأ في سؤال الانترفيو، ال Interfaceمقابل ال:Model class
-1الإجابة البسيطة:
ال Interfaceوال Classمشتركين في اني اعمل Typing لل Dataبتاعتي، بس الفرق ال Interfaceف الوضع الافتراضي لما اطلبه عشان اتأكد من نوع الداتا، ارجعله كل الداتا المطلوبة، لكن ال Classبيفرض اني اقوله المتغير ده مطلوب ولاoptional ، تعالى اضربلك مثالين:
ال:Interface
```
export interface Root {
id: number
title: string
}
```
لازم ارجعله الاتنين والا مش هيشتغل او ال Compilerهيوقف البروجيكت.

لكن ال:Model Class
```
export class Root {
id!: number
title?: string
}
```
الاولى معناها لازم، التانيةoptional ، يعني ممكن مترجعش data للمتغير ده.

-2ال compileوال:Run time
-ال:Interface
وقت ال Compileبس بيكون موجود، ولا يعتبر جزء من الكود، بيسمحلك تتأكد بس من البيانات بتاعتك وان البنية صح، ولما يتأكد ان كل حاجة صح، يسمحلك ان البيانات توصل للواجهة؛ فهي اتعملت للراحة والسهولة، ومش بتاخد مساحة ولا بتكون موجودة وقت ال.Run time
-ال:Class
الكود بيفضل فى ال Run timeيعني مش بيتمسح، بيكون موجود ف الحالتين، ال Compileوال Run time؛ اتعملت للراحة والسهولة، بس بتاخد مساحة.

-3امتى استخدمها، وامتى لا؟:
--ال:Interface

-تستخدمها لما تكون عايز شكل مكون لل Dataالمطلوبة فقط، وكلها تكون تعريفية للمطلوب، مش بتعمل عملية معينة، وخاصة لما تكون data هتستخدمها في اكتر من Component او.Service
-متقربلهاش لو عايز يكون في default value للمتغير بتاعك، او انك عايز تعمل implement لاي حاجة، او انك محتاج ال Constructorاو محتاج انه يعملكfunction ، يعني مثلًا لو الداتا الراجعة يتغير من شكلها تضيف كلمة او معادلة، إلخ.
--ال:Model class
-تستخدمها لما تحب تطلع instance من ال Dataالي رجعالك، او انك تعملdefault value ، بتدعم انك تستخدم ال Constructorووجود.Functions
-لا يفضل استخدامها لو ال Dataقليلة وملهاشdefault value ، وخلي بالك ان الاتنين بيعملوا Type checking ؛ فلو مش هتستخدم حاجة خاصة من خواص الClass، يفضل تستخدم ال.Interface
---
-4نقاط إضافية:
-ال:export keyword
ال :Interfaceبتاخد.interface keyword
-ال :Classبتاخد.class keyword
-ال:Real time use
ال :Interfaceمرتبط بال.architecture
ال :Classهو Design pattern مرتبط بال.project structure

Question 20

ما هي ال Reusable componentsفي الAngular، وأمثلة عليها؟

)سؤال انترفيو(

لما ال Angularاتعمل، اتعمل لتعزيز إمكانية إستخدام الكود، معنى كدة انك تحاول تستفاد من الكود اكتر من مرة بدل ما تنشئ جديد؛ فال resuable componentsهي components قابلة للإستخدام اكتر من مرة وفي مختلف الصفحات، والعملية دي بتم عن طريق انك تعمل Child components ل Parent Componentاو page ومن خلال اكتر منDecorator ، أشهرهم ال Inputوال ViewChildودول Property decorators يعني بيتعاملوا مع الخواص.

ال architectureالمتبع عند بناء ال Angularعند غالبية الأشخاص الي عندهم خبرة بيكون في Shared module ومن ضمن ملفات الSharedملف اسمهpartials ، او مش شرط يكون مع ال Sharedبس ده مش موضوعنا.

ملف ال partialsده بيكون فيه Reusable components و componentsزي ال Headerوال Footer.

زي ما قولتلك ال Resuable componentsهي قابلة للإستخدام اكثر من مرة؛ فهديك أمثلة عليها:

1-ال Title:

وده أبسط مثال فيهم.

في مواقع كتير بتلاقي ال Titleالي في الصفحة بيتغير، مثلًا Cart - Blog ، إلخ...؛ فأنت ممكن تعمل Reusable componentsفي الحتة دي بالشكل ده:

هتيجي في ال Resuableوهي هنا Child component وتكتب الكود ده في ال View:

```
<h1>{{title}}</h1>
```

والكود ده في ال Component Ts:

```
@Input() title!: string;
```

وهتيجي في ال Viewعند الParent component، وتكتب الكود ده:

```
<app-title title="Blog"></app-title>
```

فتلقائي الجزء ده في صفحة ال Blogهياخد كود ال Htmlوال Css.

2-ال Card.

3-ال Buttons.

وأمثلة كتير مش بتخلص وكمان عندك مكتبة زي ال Angular materialخير مثال على فلسفة ال Resuable components، والموضوع مش بينتهي عند نقطة مشاركة ال Htmlوال Cssبس، في كمان مشاركة الTypeScript، وهديك مثال:

كنت شغال قبل كدة في بروجيكت Angular مع زميل؛ فكان مهم بالنسبالي ان يكون الكود ممتاز والشخص ده يكتسب خبرة، وكان عندنا 3 صفحات فيهمReactive Forms ؛ فلقيت اني ارشحله مقالة فيها الكود النهائي لل Inputوال Labelبالشكل ده:

```
<text-input
[control]="fc.email"
[showErrorsWhen]="isSubmitted"
type="email"
label="Email">
</text-input>
```

والكود ده كان بيطلعلك ال Errorsوال Inputله Label و Typeوال Controlsالمتحكمة بيه؛ الكود ده مكون من 3 Reusable components.

وكل Component تحتوي على تفاصيل Typescript واسطر من الHtml؛ فكان من الطبيعي يسألني ليه منكتبش فى ال Componentبشكل مباشر؛ فجاوبت بالإجابة دي:

لو لاحظت ان ال3 Forms الي عندنا معمولين بشكل أفضل وكإن Component ال loginمفيهاش غير 2 div بس وده بيدينا 3 حاجات:

-1الوضوح وتحقيق المطلوب بشكل سريع.

-2لو حصلت مشكلة هيبقى سهل اننا نحلها.

-3مش هنتعب في بناء 20 form بالطريقة دي.

وبالمناسبة ده مثال بسيط جدًا جدًا بالنسبة للReusable component؛ لأنه حرفيًا ملوش حدود وأفتكر اني عملت قبل كدة 4 profile page لنفس البروجيكت بالطريقة دي بتعديلات بسيطة جدًا من خلال الdecorator ()Input، وبدل ما اتعب نفسي في ال Testوالتعديل؛ فبقيت اعدل الاربعة من مصدر واحد.

Question 21

ما هي ال Observablesفي الAngular؟
-1التعريف:
-الترجمة الحرفية ليها هي المراقبات، وهي عبارة عن ممرات للdataتربط ال Applicationال SPAببعضه من خلال رسايل، وبيتم إستخدامها لعمل handling للasynchronous programming.
واحد يجي يسألني سؤال:
هي مش ال Promiseبتعمل كدة برضو؟
هقولك صح، بس في بينهم فروق كتير وأهمهم ان ال Observableبيوفرلك التعامل مع multiple values وكمان تستخدم كل ال Rx Operatorsعشان تعمل Handel للprocessesنفسه.

-2طبيعتها:
هي عبارة عن Design pattern بيتم إستخدامه للتعامل مع.asynchronous programming
بتستخدم في الHTTP module؛ عشان تعمل Handel للAJAX requestsوال.responses
وبتستخدم في ال Routerوال Forms؛ عشان تراقب ال reponsesبتاعت ال.user
---
-3الإستخدام الأساسي:
هو إنشاء Instance من ال dataالي معمولهاSubscribe ، وده بيسمح بتسجيلها في Subscription اقدر اعمله (Cancel)من خلال ال unsubscribeلما يحصل destroy للComponent.
وبالمناسبة ال Observableبيكونcold ، يعني مش بيشتغل غير لما اعمله.Subscribe
---
-4ال:Observer
هو handler بيساعد ال Observableعلى معالجة الإخطارات وإستقبالها، وبيتم التعامل مع الإخطار بثلاث طرق الأساسي فيهم واحدة وهي الnext، والتانية هي عند حدوث error واسمها error والتالتة هي لما يتم معالجة البيانات واسمها complete والاتنين دول اختياري، مش شرط تكتبهم لما تعمل.Observer
---
-5المميزات عن باقي الطرق:
-ال callbackله ثلاث حالات: اما ناجح او حصلت مشكلة او عملية ال.complete
-ال.Lazy load
-تقدر تعمل Emit لاكتر من.Value
-ال cancellableوده بيحصل عن طريق ال.unsubscribe
-ال rx operatorsوانها بتقدر تتعامل معاهم.

Question 22

ال #Agileباختصار شديد جدا هو طريقة في ادارة المشروعات بتعتمد علي تقسيم المشروع الكبير لاجزاء مستقلة والتعامل مع كل جزء بشكل مستقل

كل جزء بنسميهsprint

الفايدة الأساسية من كدة ان كل مجموعة في فريق العمل بتركز علي الجزء اللي يخصها وبس

كل جزء بيخلص بتعرضه علي العميل ولو فيه مشاكل او ملاحظات بيبأة سهل تحلها بعكس لو بتسلم المشروع علي بعضه وظهرلك مشاكل وعاوز تحلها علي مستوي المشروع كله مرة واحدة

تخيل كدة السيناريو الشهير والرخم دة واللي مفيش صاحب شغل مابيشتكيش منه واللسبب الرئيسي في الخلافات بين العميل واصحاب العمل

تخيل بأة جالك عميل طلب منك تعمله شغل معين وانت بذلت مجهود جبار عشان تنجز المشروع المطلوب منك وبعد تعب ومصاريف جامدة رايح توري العميل الشغل قالك بس مش دة التصور اللي انا كنت عاوزه

هتختلفوا علي تفاصيل كتير, الحل الوحيد اللي قدامك انك تعيد المشروع كله من جديد وبرضو هتدخل في نفس الدوامة دي تاني وتالت

شوفت المشكلة دي مكلفة أد اية .. مجهود ووقت وفلوس وبتخسر عملاء

أجايل بأة بتحل المشكلة العويصة دي

بكل بساطة هتقسم المشروع لأجزاء صغيرة وهتشتغل علي كل جزء بشكل مستقل

كل جزء بيخلص بتوريه للعميل ولو حصل مشاكل بينكم هتلاقي ان تكلفة اعادة تصميم الجزء الصغير دة ارخص بكتير من اعادة تصميم المشروع كله

لو قررت تعتمد Agile في شغلك ممكن تستخدم framework تساعدك وتسهل عليك الموضوع

في frameworks كتير لكن اشهرهم #Scrumو .#Kanban

البوست الجاي هنتكلم باستفاضة اكتر وهنتعمق اكتر في ال frameworks دي

Question 23

النهاردة معانا واحد من اهم اسئلة الانترفيو مع ال.Angular
-ما هو الAsync pipe؟
-اديني حل يخليني أ Avoidال memory leaks غير اني اعمل Subscription اوunsubscribe ؟
-عندنا data عايزينها تتعرض من غير subscribe او.then
)سؤال انترفيو(
-1التعريف والنوع:
ال Async pipeهيimpure pipe ؛ وال impure pipesعمومًا بيعملوا Async لحد ما تيجي اخر نتيجة، بمعنى ان لو حصل تغييرات اثناء وجود ال userعلى الموقع هيشوفها قدامه، وشرحت مثال عليها في مقابل سابق ليا.
فبإختصار هو بيوصل لأخر قيمة راجعة وبيتعامل معاها، ولما يحصل destroy لل componentهو أوتوماتيك بيعمل unsubscribe لل observablesالي انت عاملها بيه، وده بيحميك من ال.memory leaks
---
-2مميزات الAsync pipe:
- 1 بيسهل عملية ال Renderلل data(عرض البيانات)، الي بيتعملوا بال Promisesوال.Observables
- 2 بيقضي على ال.memory leaks
- 3 مع ال Promiseبيعمل then بشكل تلقائي.
- 4 مع ال Observableبيعمل subscribe و unsubscribeبشكل تلقائي.

-3مثال مع الpromise:
-بدون الasync pipe:
كود ال Tsبيكون بالشكل ده:
```
promiseData: string;
constructor() {
this.getPromise().then(v => this.promiseData = v);
}
```
وبيحصله عملية get ثابته.
وكود ال Viewبيكون بالشكل ده:
```
<p class="card-text">{{ promiseData }}</p>
```
-بالasync pipe:
كود ال Tsبيكون بالشكل ده) لاحظ لا يوجد:(then
```
promise: Promise<string>;
constructor() {
this.promise = this.getPromise();
}
```
وبيحصله عملية get ثابته.
وكود ال Viewبيكون كدة) لاحظ طريقة كتابة ال:(Async pipe
```
<p class="card-text">{{ promise | async }}</p>
```

-4مثال على الObservable:
في الحالة العادية بدون الAsync pipe، هتكون محتاج تكتب ده كله زيادة فوق عملية الget:
1 الاولى دي عملية الsubscribe:
```
subscribeObservable() {
```

this.subscription = this.getObservable()
.subscribe(v => this.observableData = v);
}
2️⃣التانية دي عملية الunsubscribe:
ngOnDestroy() {
if (this.subscription) {
this.subscription.unsubscribe();
}
}
ال Asyncببساطة سمحتلك تشيل العمليتين دول وتعوضهم بالشكل ده:
{{ observable | async }}
وتوصل لنفس النتيجة.

اظن كدة انت فهمت فايدة الAsync pipe، وانها بتسهل عمليات عرض البيانات وانها لو شافت حد شغال بال thenاو ال subscribeاو ال unsubscribeاو بيسجل البيانات في subscription بتقوله:
we do not do that here
غير انها بتعمل render لأخر data او Value وده بسبب طبيعتها كimpure pipe.
--
اخيرًا يمكنك متابعة كافة المواضيع التي اكتبها على هذا الهاشتاج #NgOmarRaبالتوفيق للجميع.

Question 24

ما هو ال trackBy وايه فايدته؟
)سؤال انترفيو سهل(
والإضافة دي مهمة في عمل render للdata وبيعتبروها من أسس ال clean codeفي ال.Angular
---
-1مقدمة:
في الوضع الإفتراضي مش بتحتاج تستخدم الtrackBy، بس بتستخدمها لغرض معين، ايه هو الغرض؟
معروف عن ال ngForانها structural directive ومعنى structural directive ان اي تغيير بيحصل بيأثر في ال domالي هي متحكم فيه بالكامل؛ فمثلًا لو انت عندك data كبيرة ومطلوب انك تعمل render او الموقع بيعمل render ل dataكبيرة وفى الخلفية في Crud operators للداتا الضخمة دي مستمرة، هل معقول كل ما يحصل إضافة للdata يحصل Render للdata كلها؟
اكيد لا، وده دور ال.trackBy

-2ما هي الtrack by؟
هي function بتضاف للngForك propertyك، وبتولد unique id مقابل لكل item بيقابله، وده شكلها ك function في جزء ال:Typescript
trackByFn(index, item) {
return item.id;
}
وده بيكون شكلها في الView، مع ال:ngFor directive
<li *ngFor="let item of items; trackBy: trackByFn">{{ item }}

-3بشتتغل امتى وايه فايدتها؟
بمجرد كتابتها بتكون شغالة، والفايدة منها انها بتحافظ على ال dataوبتخلي ال dataالجديدة بس هي الي يحصلها renderاما الي اتعمله render قبل كدة؛ فخلاص بقى محفوظ ومش محتاج يتعمله render تاني، لكن ال data الجديدة بس هي الي يحصلها.render

Question 25

-هي الAngularبتتبع معمارية الMVCولا الMVVMواثبتلي ده؟
)سؤال مهم جدًا جدًا جدًا(

مبدأيًا المقال ده هيكون مقال (مقصور) يعني مش هشرح المعمارية بتاعت الAngularكلها، لا انا هجاوب السؤال ده بس.
-1هما مين دول؟
الMVVMاو الMVCدول عبارة عن architectural pattern ينفع تسميهم architecture او design pattern او architectural patternزي ما انا بسميهم كدة بس في الواقع هما اكبر من كدة كمان، الMVVMده الشكل او الpatternالمتبع في بناء تطبيقات الويب مع الAngularوكمان في الاندرويد وغيرها من الplatforms والFrameworks.

-2ايه هي الarchitectural patterns؟
الarchitectural patternsاتوجدت عشان تقدم حلول فعالة وشكل منظم للتعامل مع الغرض من البروجيكت والعملاء الي بيتعاملوا مع البروجيكت وعندنا اكتر من architectural pattern عندك مثلًا الMVCوده كان المتبع مع الAngularjsوعندك الMVPوكمان الMVVMالمتبع مع الAngular.

-3هي الAngularبتتبع MVC ولاMVVM ؟
لو بصيت على إجابة السؤال التاني هتلاقي اني قايلكMVVM ، ولكن لما تروح تشوف كورس بيقولك ان الAngularjsوالAngularمتبعين نفس المعمارية وهي الMVCوبصراحة معاهم حق، فعلًا الAngularمتبعة الMVCبس ده فى حالة واحدة، لو اننا اعتبرنا ان الComponentفي الAngularتوازي الControllerفي الAngularjs؛ فبإختصار الAngularبتتبع معمارية الMVVM.

-4اثبات ان الAngularبتتبع:MVVM
وعشان الإجابة توضح اكتر لازم تعرف ان الMVCوالMVVMمشتركين في اول حرفين ومتشابهين في الباقي:
الMتمثل الModelودي بتوفرلك الbusiness logic.
والVتمثل الViewوده هو العرض بالنسبة للعميل.
شرحنا اول حرفين يبقى كدة فاضلنا الCمقابل الVMوالحقيقة ان الجزء المتبقي من الاتنين design pattern متشابهين مش مختلفين، وده لإن بإختصار شديد الي بيوفرهولك الCوهو الControllerاو المتحكم، يقدر يوفرهولك الVMاو الView Modelوده جزء مجرد من الViewبس بيدير كل حاجة بتظهر على الشاشة.
فين الإثبات؟
تعالى اقولك لو جيت عند اول حرف الي بيمثل الModelوقارنت الModelالموجود في الAngularjsمع الModelالموجود في الAngular؛ هتلاقي ان الModelفي الAngularjsبيمثل الbusiness logic، لكن الModelفي الAngularمعندوش اي Logic لكن الLogicكله بيحصل في الView Model.

حقيقي كان نفسي اكملك فوق المقال ده واشرحلك ايه هو الMVVMاو على الاقل الView Modelولكن المنشورات في لينكدان أخرها كدة؛ فإنتظرني في الجزء التاني وانا ان شاء الله هشرحلك الMVVMمع الAngular بالكامل.

فى الAngular لازم تذاكر:
Angular - Rxjs وممكن NgRx؛ عشان لو بتسأل عن الcategories الأساسية

Question 26

في البوست ده هنتكلم عن واحد من اهم اتنين Topics الي بتميز Angular عن اي Framework تاني، وهو الRxjs.
1-اي هي الRxjs؟
الRxjs هي مكتبة js تعني Reactive Extensions for JavaScript وخلي بالك من اسمها؛ فهي مش معمولة للAngular بس، ولكن الAngular بتتميز بيها؛ لأنها أساسية فى الangular، صعب تلاقي بروجيكت Angular مفيهوش Rxjs ؛ على عكس اي فريمورك تاني، ومن كلمة Reactive اقدر اقولك دور الRxjs واشرحلك هي ايه اصلًا الReactive Programming.

2-الReactive programming:
او البرمجة التفاعلية هي عبارة عن نموذج برمجي (design paradigm) يعتمد على منطق البرمجة الغير متزامن (asynchronous) وبتعتمد على الـ (stream) والstream هي مجموعة متماسكة بتعمل تدفق للبيانات داخل البروجيكت بشكل اسرع، الكلام حلو بس انا مش فاهم حاجة، بص من الاخر اعتبر ان الـ Reactive programming عبارة عن (مصعد)، والبرمجة العادية عبارة عن سلم.

3-ليه استخدمRxjs ؟ وهل لازم؟ ولا في بديل؟
زي ما قولتلك الReactive عبارة عن مصعد؛ فهي اسرع ووجدت الReactive programming عشان تحل مشاكل الـ memory leaks من بطأ وتسريب وبتضطر انك تكتب الكود اكتر من مرة بدل حفظه، غير انها منظمة اكتر وبتوفر عليك وقت ومجهود كبير ومختصرة، طبعًا موجود حل انك تشتغل بروجيكت كامل من غير الRxjs وانك تعتمد على الـ Promise والـ Higher order functions بتاعت الـ javaScript بدل من الـ Observables والـ Operators بتاعت الRxjs، بس انت كدة حرفيًا مستغلتش قوة الAngular وكدة البروجيكت بتاعك هيكون ابطأ.

4-هل تعلمها صعب؟ ومصادر للتعلم؟
هو سهل بس اتعلم صح، يعني مثلًا في ليست مقدمة من قناة Angular army حرفيًا الليست دي أبدعت في شرح الRxjs، بس الـ Rxjs مينفعش معاها مذاكرة المستعجلين، بمعنى انك متكترش عن درسين تلاتة فى اليوم وتراجع عليهم كويس وانت بتتفرج على الدرس لازم تكون فاتح الـ documentation وحاول تشوف مقالات متوسعة اكتر مع كل درس، وكمان اضرب بخيالك كل السيناريوهات الممكنة واكتب ملاحظاتك؛ عشان توصل لاكبر إفادة ممكنة، وممكن تتابع الهاشتاج ده #rxJsRadwan هنشر عليه ملخص ل Topics كتير فى الRxjs، وده لينك الليست بتاعت Angular army:
https://lnkd.in/d7xF74GX

واخيرًا لو انت متابعني على هاشتاج #NgOmarRa؛ فأكيد لاحظت اني اخر فترة قللت بوستات، وده مش كسل مني ولا اني مبقاش عندي مقالات، بالعكس انا كنت بحضر شرح لواحد من اهم المواضيع فى الـ Angular والي هناخد معاه رحلة طويلة شوية؛ عشان كدة بقاله هاشتاج مخصص #rxJsRadwan؛ كل الي طالبه منك تفاعل وكومنت ولو بكلمة(up) ؛ عشان المحتوى يوصل لاكبر عدد ممكن، ده هيشجعني اني اكمل بنفس قوة الكتابة فى المحتوى ده وأفضل بإذن الله، ومن المقال الجاي هنبدأ شرح واقعي للRxjs.

Question 27

الـMicro-Frontends

عند تطوير تطبيقات frontend الكبيرة ، يصبح development أكثر قسوة بسبب side-effects غير المتوقعة أو dependency .

يمكن لمفهوم Micro-Frontends ، الذي ظهر لأول مرة في 2016 ، حل بعض من هذه المشكلات وإنهاء عصر monoliths.

ما هيMicro-Frontend

يعد Micro-Frontend نمط architectural لتطوير تطبيقات frontend الكبيرة التي تستخدم فكرة تطوير أجزاء من APPمستقلة واستخدمها بشكل منفصل في اماكن اخري من المشروع وبالتالي ، يعمل التطبيق بأكمله كتكوين لحلول مختلفة ، فضلاً عن التقنيات التي يمكن تطويرها ونشرها بشكل منفصل دون تعطل اي منها.

الـ layout العام Application الويب

انظر للمثال في الكومنت

يتم تقسيم functionalities إلى features وتؤدي إلى technically مختلفة في (Micro-App) ، والتي ستعمل معًا ككل و يكون هذا بشكل أساسي غير مرئي لـ user وذلك باستخدام بعض تصميم الـ layout

يتم تحقيق ذلك بشكل عام باستخدامcontainer app ، والذي يحتفظ بالعناصر الـ core مثل Header أو Footer ويهتم بـ services الأساسية مثل authentication كما أنه يفي بمهمة دمج جميع الـ features الأخرى-Micro) (Applicationsفي تلك الصفحة.

فوائد (Micro-Frontends) وسبب استخدمه من الشركات لبناءsoftware

يمكن تقسيم فوائد Micro-Frontends إلى : الفوائد

organizationalوtechnical / architectural

الفوائدtechnical / architectural

أن Micro-Frontends يتم إنشاؤها عادةً في Run-Time وليس في Build-Time ، مما يميزها عن أساليب تجميع package .

ويقلل من side-effects في development و bugs علاوة على ذلك ، يمكن لكل team العمل بالتكنولوجي المفضلة له و إتاحة الفرصة لكل team للاختيار بحرية من بين Frameworks المختلفة مثل Vue.js أو React أو Vanilla Javascript، فهو مناسب أيضًا libraries و code styles و tooling الأخرى اثناء التطوير.

من المعروف ان الـ App الأصغر يسهل إدارتها بشكل طبيعي و يصبح الـ Appنفسه أكثر قوة وأقل عرضة bugs ولكن تواجه الشركات عند العمل علي التطبيقات الثقيلة التحدي المتمثل في الحفاظ على قواعد الـ code الهائلة وبتالي Micro-Frontends يساعدها علي ذلك بشكل مثالي.

فوائد الـ) Organisational التنظمية(

قد يتم استبدال فريق Frontend أو backend بـشكل سلسل.

يمكن لهذه teams تطوير ونشر application في environments مختلفة و التي تعمل عليها بشكل منفصل ، مما يؤدي إلى مزيد من flexibility و cycles تنفيذ أقصر.

تستفيد المؤسسات الأكبر بشكل خاص من السرعة التي يمكن أن تقدمها teams الصغيرة والمركزية.

بطبيعة الحال ، لكي يعمل هذا النهج ، يجب أن يكون لديك Backend services موجهة لـ feature المستخدمة و على الرغم من إظهار العديد من الفوائد ، قد لا يكون Micro-Frontend مناسبًا لأي نوع من المشاريع.

Question 28

المقال ده اعتبره رقم 1 في شرح ال Rxjsمع الAngular، وفي المقال ده هقولك ايه فايدة ال Rxjsوايه هي ال categoriesالمتاحة (والي ان شاء الله هنشرحهم سوا طول الليست دي).

-1ما هي فائدة ال:Rxjs

-تحويل البرمجة العادية لبرمجة تفاعلية غير متزامنة(async) ، وعناصر قابلة للملاحظة.(Observables)

-التكرار من خلال تحويل القيم لشكل.stream

-بتعمل Mapping لقيم مختلفة.

-فلتر لل.stream

-ربط اكتر من stream ببعض.

-2ماهي ال:Observables

هي مراقبات وملاحظات واتكلمت في مقال منفصل عنها، لكن يهمني دلوقتي تعرف عنها انها stream ومعنى كلمة stream هو تيار او تدفق، فبالتالي ال Observableعبارة عن Stream بيشيل الداتا او يحفظها في متغير وهو يمثل عملية الاتصال بين ال Backendوال Frontendتحت مراقبة وملاحظة وقابلية للتعديل، وبقول هنا كلمة مراقبة؛ لإن ال Obsesrvableله 3 حالات تقدر من خلالهم تراقب عملياتك، وقابلة للتعديل لانها بتتميز بإن ليها Operatorsبتتحط فى حاجة اسمها Pipe ودي مختلفة عن pipe ال.Angular

---

وبكدة اكون اتكلمت عن الRxjs، تعالى دلوقتي نتكلم عن العناصر الي هتتشرح على مدار الليست، وهتكون مقسمة لفئات؛ فده هيفيدك ان شاء الله في معرفة تقسيمة ال:Rxjs

-1عناصر ال Rxjsالاساسية:

بإختصار شديد هو عنصر واحد لازم يكون موجود وهو ال Observableوباقي العناصر الاساسية بتدور حواليه اعتبره شمس الRxjs، ولكن متصنف لاتنين ال Observableنفسه وده cold وال Subjectوده hot Observable ، بعدين هينزل مقال يشرحلك دي بس خلينا نتعرف على باقي العناصر وهما) ال Operatorsوال.(Scheduler

---

-2ال:Rxjs patterns

أكيد اي technology ليها patterns مشهورة؛ فبالتالي عندنا 4 باترن هنستخدمهم مع ال Rxjsدايمًا، وهم:

-ال.Observable

-ال.Functional

-ال.Observer

-ال.Iterator

ومش محتاج اقولك اننا ان شاء الله هنفهمهم سوا طول ما انا بكتب فى الليست دي.

-3ال:Subjects

هي عبارة عنhot Observables ، وعندنا كام نوع مشهورين هنشرح اغلبهم ان شاء وبالترتيب حسب الأهمية والإستخدام:

-ال Subject

-ال BehaviorSubject

-ال ReplaySubject

-ال AsyncSubject

-4ال:Operators

ودي عبارة عن Operators بيتم إستخدامها داخل ال Pipeللتحويل من شكل لشكل وال Topicده هو الي فيه اغلب الادوات الي هيتم شرحها في الليست دي، وفي انواع كتير جدًا من ال Operatorsبإذن الله هيتم شرح الانواع في مقال منفصل، وال operatorsهيكونلها مقالات متنوعة بين شرح ومقارنات.

وبتتبع ال operatrosال Marble Diagrams، ودي هيتم شرحها وتبسيطها طبعًا في مقال منفصل.

-5ال:Memory leaks

ودي طرق هنستخدمها لمنع ال Memory leaksابرزهم:

ال Async pipeوال Unsubscribeوشوية Operators.

-6ال:merging Observables

وهي انك تربط observable 2 مع بعض وال operatorsدي هتعرفها بإن أخرها كلمMap ، مثال:

ال.mergeMap

-7تحويل ال Observablesالمتجمعة ل:Array

وفي الحالة دي عندنا شوية functions ومقولتش Operators لان دول مش بيكونوا داخل Pipe ولكن بيتعاملوا بال.Marble Diagrams

وامثلة على ده combineLatest :و forkJoinواي حاجة تشبههم وان شاء الله هيتم الشرح منهم قدر المستطاع.

اخيرًا بطلب منك تكون صبور ومتستعجلش في تعلم الRxjs، وياريت تتفاعل وتعمل كومنت ولو بكلمة(up) ؛ عشان المنشور يوصل لاكبر عدد من الناس.

Question 29

بسم الله الرحمن الرحيم
ما هي الmarble diagrams؟
)سؤال مهم جدًا جدًا وواحد من اهم ال keysلفهم ال.(RxJs

-1مراجعة على ال:operators
دلوقتي احنا في اخر بوست اتكلمنا عن ال oberatorsوقولنا انها نوعين:
الاول:Creation Operators
وده بينشئ observable وهتلاقي صورته على ال Marble Diagramsفي الجهة اليمنى من الصورة.
التاني وهو ال Pipeable Operatorsودي قولنا انها بتدخلها observable وبتخرج observable زي الصورة الي على الشمال كدة.

-2تعريف ال:Marble diagrams
من الصور هتحس انك فهمت المعلومات الي اتذكرت فى النقطة رقم 1 بدون اي توضيح مني، وده هو غرض ال Marble diagramsوهي عبارة عن timeline بيوضح تأثير ال Operatorsعلى ال Observableاو ايه ال observableالي ممكن تخلقه ال.Operators
وعندنا نوعين من ال diagramsدي وهي ال diagramالي ليها arrow واحد، وال diagramsالي ليها اكتر من Arrow.

-3تسجيل الملاحظات على ال:(one arrow)
)لاحظ ان الرسومات الي في الصورة دي.(one arrow
لكل Operator فى ال Rxjsهتلاقي diagram بيوضح عملها فى الdocumentation؛ وده يعتبر رسم سريع او زي ما بنقول رسم كروكي بيوضح التأثيرات.
وتعالى هنا اوريك ازاي تسجل ملاحظاتك من مجرد رؤيتك لل.diagram
مثلًا من ال diagramاليمين انت هتطلع بالنتايج دي:
-1ان ال Sourceاو الكود ما قبل ال operatorغير موجود او موجود داخل ال Operatorنفسه؛ فده يعني انها (creation operator).
-2انك طلعت ب Observableعلى الشكل المطلوب وشكل منفصل) عملية.(flattening
-3دخلت ب Arrayطلعت بأرقام؛ فمعنى كدة انها بتقبل Array وبتعمله.flattening
---
ال diagramالشمال:
وده عبارة عن one arrow بس هما اتنين operators مش operator واحدة، واخترت المثال ده عشان يبان الفرق بينها وبين النقطة رقم 4.
-1يوجدsource code ، وهي بداخلها معادلة او ارقام غير الsource code؛ فمعنى كدة انها.Pipeable Operators
-2وضحت دور عملية ال mapفي انها حولت ال1 ل2، وال2 ل4.
-3ال operatorالاخير وضح التأخير بتاع الdelay، انه هيستنى شوية قبل ما يعمل التأثير.
طبعًا عالم ال diagramsمع ال rxjsلا ينتهي دي بس أمثلة سريعة.

-4ال diagramالي ليه اكتر من:arrow
هحطلك الصورة فى الكومنت، ومن الصورة هتفهم ان ال mergeبيدخلها data من اتنين source او من اكتر من Observableوساعتها هتفهم ان:
-1ال mergeهيcreation operators ؛ فهي ربطت اتنين Observable وطلعت بواحد جديد، يعني هي اكتر من

مجرد تعديل على القيمة او الشكل، هي طلعت ب new observableمن خلال اتنين.
-2انها بترص الداتا حسب ما بتجيلها، يعني مبتاخدش ال observableللاخر وبعدين تشوف التاني، لا الي يوصل الاول من اي observable بتدخله ال observableالجديدة.
-3على عكس ال concatالي بتستنى كل observable تخلص؛ عشان تقول يلا الي بعده.

Question 30

ما هوJWT

الـ JWT أو JSON Web Token ، هو معيار مفتوح يستخدم لمشاركة معلومات الأمان بين طرفين client - وserver

لماذا نستخدمJWT

و يستخدم لعملية الـ authentication بين Client و Server ولنكن واضحين اكثر من استخداماته الاساسية Authorizationو Information Exchange ويوجد العديد من السيناريوهات التي يستخدم فيها ايضا و يتم استخدام Kay خاص و سري بواسطة المُصدر الاساسي لهذه العملية

كيف يعملJWT

يتكون JWT من ثلاث اجزاء هم (Header , Payload , Signature) مثل هذا:

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJVc2VySWQiOjEyMywiVXNlck5hbWUiOiJhZG1pbiJ9.Qjw1epD5P6p4Yy2yju3-fkq28PddznqRj3ESfALQy_U

الـ Header هو جزء مخصص يقوم بتعريف نفسه للسيرفر, اظهار نوعه, الهاشنق اللي يستخدمه وغيرها من المعلومات التي تكون متاحة للقراءة من اي شخص

الـ Payload هو الجزء الذي يحتوي على المعلومات (claims) المأخوذة من الكلاينت للسيرفر و هو يحتوي على المحتوى الذي يتحقق منه السيرفر من أجل مصادقة هذه المعلومات.

الـ Signature جزء مهمته يقوم بأخذ (Header + Payload) ويقوم بتطبيق خوارزمية HMAC على الجزئين, لو قام المهاجم بتغيير Payload يتنافى مع Signature يرد له بالرفض التام ف هنا نفهم ان وظيفته هي التأكيد على الـ authentication

المزايا:

-بسبب شمولية json ، يتم دعم و استخدام JWT من قبل العديد من اللغات البرمجية مثل C # و JavaScript و NodeJSو PHP و غيرها.

-يمكن لـ JWT تخزين بعض المعلومات غير الحساسة اللازمة لمنطق الأعمال الأخرى في حد ذاته

-سهولة الإرسال ، تكوين jwt بسيط جدًا ، لذا فهي مريحة جدًا للإرسال

-لا يحتاج إلى حفظ معلومات session على server ، لذلك من السهل التعامل مع الجلسات

Question 31

السلام عليكم في البوست ده مش شرح؛ هتكلم بس عن مزايا وعيوب ال Ngrxوبناء عليه تقرر تتعلمها او لا.
مزايا وعيوب ال.Ngrx
مزايا ال:Ngrx
-برمجة تفاعلية(Reactive) ؛ لذلك فهو أكثر كفاءة من غيره.
-أسهل في الdebug، بيتتبع جميع تغيرات ال Stateويرجعها لأصلها بال.dev tools
-ال dev toolsأشبه بشرح لما يتم والتتبع من خلالها بيخليك واعي لكل خطوة.
-قابل للتوسع؛ لأنه بيعمل على تخزين الهياكل (data structure) المعقدة.
-أسهل في الصيانة؛ لأنه بيحفظ ال Statesفي مكان واحد.
-بيسهل ال communicateبين Component والتانية سواء.Parent, Child, Sibling
-بيوفرلك Reducers و Actionsو Selectorsو Effectsو Entitiesو router-storeودي كلها حاجات بتوفر معاك وقت ومجهود في كتابة الكود وكمان بتسهل عليك عملية إدارة المشروع.
-آمن جدًا؛ لأنه بيتتبع جميع تغيرات ال Stateويرجعها لأصلها.

عيوب ال:Ngrx
-أي مصدر خارجي بالنسبة لAngular؛ فهو بطبيعة الحال ممكن تعتبره عيب وبزيادة انه بيتطلب مكتبات إضافية ولكن لفائدته فهو مقبول في النقطة دي.
-صعب التعلم؛ لأنه بيتطلب فهم لمبادئ ال.Redux
-أوقات بتحصل مشاكل في ال debugوبتكون complex أكتر من الإعتيادية لو خبرتك حديثة بيه.
-البدايات بتكون صعبة في انك تعتمد عليه وبيستهلك الكثير من الوقت، ولكن مع التعود بيوفر معاك جامد في الوقت.
-أوقات بيكون صعب الفهم او مشتت بس مع تعلم الRedux pattern؛ هتلاقي الموضوع أسهل.
-بعض المطورين ممكن يكونوا مهووسين بال state managementويستخدموه ل Projectsصغيرة ودي حاجة سلبية جدًا وبرضو لو مشروع كبير والمطور اشتغله بشكل عشوائي هيصبح الأمر اشبه ب(الجحيم).
-المصادر التعليمية معقدة.