**1. Trình bày được ý nghĩa và khái niệm Observer**

- Observer : là một object có chứa các phương thức next, error và complete để xử lý dữ liệu tương ứng được gửi từ Observable.

- Có 4 loại Observer

* Observer
* SingleObserver
* MaybeObserver
* CompletableObserver

**2. Trình bày được ý nghĩa và khái niệm Observable**

- Đại diện cho khái niệm về một tập hợp các giá trị hoặc các sự kiện trong tương lai. Khi các giá trị hoặc sự kiện phát sinh trong tương lai, Observable sẽ điều phối nó đến Observer.

- Observable sẽ phát ra một sự kiện hoặc dữ liệu nào đó

- Có 5 loại Observable sau

* Observable
* Single
* Maybe
* Flowable
* Completable

**3. Phân biệt sự khác nhau giữa var, let, const**

|  |  |  |
| --- | --- | --- |
| **var** | **let** | **const** |
| - Được khai báo lại | - Không được khai báo lại | - Dùng để khai báo hằng số |
| - Được hoisted | - Không được hoisted | - 3Không hoisted |
| - Là function scope | - Là block scope | - Là block scope |
|  |  | (không thể gán lại ô nhớ mới, nhưng thay đổi được giá trị) |

**4. File angular.json và package.js sử dụng để làm gì?**

- **Package.json** là file cấu hình của npm, giúp cho npm hiểu nó cần phải cài đặt cái gì, thông tin về ứng dụng, phiên bản, … Được dùng để khai báo các thư viện trong project.

- File package.json được viết bằng json

- File package.json được đặt ở thư mục gốc của project.

- Thành phần file package.js gồm:

* Name
* Version
* Description
* Author
* Dependencies

**- angular.json:** cung cấp mặc định cấu hình cho toàn bộ không gian làm việc và dự án cụ thể cho các công cụ xây dựng và phát triển do Angular CLI cung cấp. Giá trị đường dẫn được cung cấp trong cấu hình có liên quan đến thư mục không gian làm việc gốc.

- Dùng để khai báo cấu hình CLI

- Thành phần file angular.json:

* Version
* newProjectRoot: Đường dẫn nơi các dự án mới được tạo
* defaultProject: Tên dự án mặc định để sử dụng trong các lệnh,
* schematics; Một tập hợp các sơ đồ tùy chỉnh tùy ng generatechọn lệnh phụ mặc định cho dự án
* projects: Chứa một tiểu mục cho mỗi dự án (thư viện hoặc ứng dụng) trong không gian làm việc.

**5. Kể tên các directive được hỗ trợ trong angular**

- Component directive: các selector.

- Attribute directive: ngClass, ngStyle, ngModel.

- Structural directive: ngIf, ngFor, ngSwitch.

- Custom directive: do người dùng tự định nghĩa

**6. Trình bày được ý nghĩa và khái niệm subscribe**

- Subscribe : là một phương thức xuất phát từ [rxjs](https://github.com/Reactive-Extensions/RxJS) thư viện mà Angular đang sử dụng nội bộ. Đây là một phương thức được đăng ký theo dõi. Bất cứ khi nào phương thức được gọi, một thực thi độc lập của quan sát được xảy ra.

- Khi chúng ta làm việc với Observable thì sẽ thường xuyên sử dụng phương thức subscribe để theo dõi các thay đổi của dữ liệu hoặc nhận về kết quả của một công việc nào đó.

- Khi API trả về kết quả thì dữ liệu đó sẽ được log ra qua phương thức **subscribe**.

**7. Trình bày khái niệm Single page application (SPA)**

- Single Page Application : ứng dụng trang đơn

- Là 1 ứng dụng web giúp nâng cao trải nghiệm người dùng bằng cách sử dụng HTML5 và AJAX

- **Cách làm việc:** Load lần đầu khi duyệt web, những lần sau client gởi những ajax request 🡪 trả về bằng JSON 🡪 update những DL cần thiết mà ko load lại từ đầu

**8. Trình bày về Data binding trong component**

- Databinding là kỹ thuật nơi dữ liệu được đồng bộ giữa component và tầng view .

*Ví dụ:* khi người dùng cập nhật data ở tầng view thì Angular cũng cập nhật giá trị đó ở component.

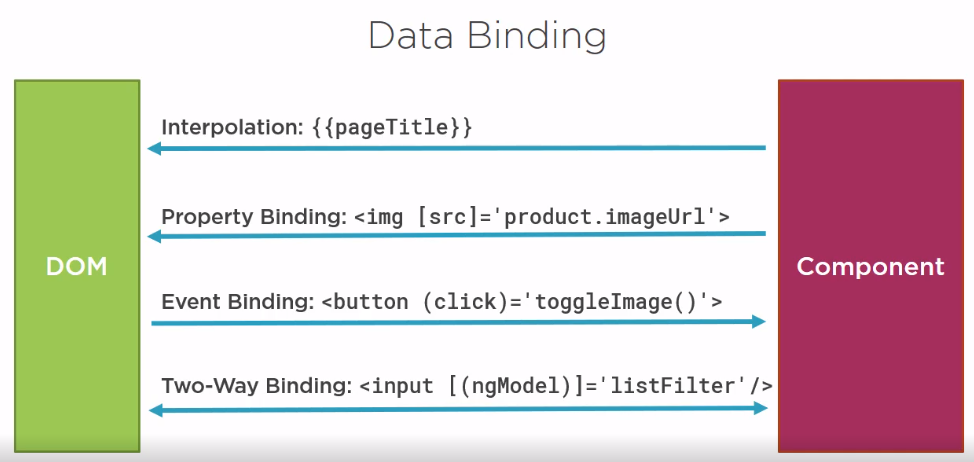
- Data binding trong Angular có thể chia ra làm 2 nhóm.

* **One way binding (binding 1 chiều):** dữ liệu được truyền 1 chiều. (Có thể từ view sang component hoặc ngược lại từ component sang view)
* Component 🡪 view:
* Interpolation : **{{ variable }}**
* Property Bingding : **<input [value] = ‘variable’>**
* View 🡪 Component:
* Property Binding
* Event Binding chúng ta sử dụng để bind các sự kiện như click chuột, sự kiện bàn phím …

**<input (event) = ‘tên function ($event.target.value)’>**

* **Two way binding (binding 2 chiều):** chúng ta thay đổi dữ liệu từ component qua view và ngược lại từ view chúng ta thay đổi dữ liệu, hữu dụng khi mình làm form. Chúng ta sử dụng ngModel để thực hiện việc binding 2 chiều.

**<input [(ngModule)] = ‘variable’>**



**9. Trình bày sự khác nhau giữa navigateByUrl và navigate**

- router.navigateByUrl :cung cấp URL mới “hoàn toàn”.

- router.navigatetạo một URL mới bằng cách áp dụng một loạt các lệnh được truyền vào, một bản vá, cho URL hiện tại.

**10. TypeScript là gì?**

- Là ngôn ngữ lập trình do Microsoft phát triển nhằm bổ trợ cho JS xử lý các vấn đề liên quan đến kiểu DL, hướng đối tượng…

- Code Typescript sẽ được transpile sang JS trước khi chạy 🡪 để chạy trên Browser

- Có thể nói Typescript là superset của JS

**11. Tại sao nên sử dụng Typescript**

- Hỗ trợ lập trình hướng đối tượng.

- Giúp dễ dàng phát triển và duy trì các ứng dụng phức tạp hơn.

- Để đơn giản hóa mối quan hệ giữa các thành phần khác nhau và cách khung được xây dựng sẵn.

- Cung cấp một hệ thống kiểu tùy chọn cho JavaScript.

- Cung cấp các tính năng được lên kế hoạch từ các phiên bản JavaScript trong tương lai cho các công cụ JavaScript hiện tại.

- Hỗ trợ định nghĩa kiểu. (Generic)

**12. Làm thế nào để lấy được dữ liệu từ component cha sang component con**

- Cơ chế truyền dữ liệu Component cha vào Component con bằng @Input

- Ở cha: khai báo biến : kiểu DL

- Đặt @input() ten: kiểu DL ở con 🡪 Tự mapping

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

*Component này hiện thị thông tin thuộc tính product của nó, và có dòng tiêu để titlebox, product chính là sản phẩm chọn của Component cha AppComponent::selectedProduct ...*

- Cần khai báo thuộc tính product với decorator là @Input(), nghĩa là thuộc tính này sẽ được Component cha thiết lập

- @Input() product:Product, Component cha sẽ thiết lập bằng thuộc tính product thông qua thuộc tính HTML cùng tên trong HTML [product]

- @Input('title') titlebox: string, thuộc tính titlebox được Component cha thiết lập thông qua thuộc tính HTML [title]

*Trong Component này cũng thiết lập khi bấm vào nút thì gọi đến phương thức yeucaudathang()*

*Giờ trong app.component.ts (Component AppComponent), sử dụng như sau (Trong thẻ app-product-detail để ý xem Component cha đã truyền giá trị vào Compnent con như [product] = "selectedProduct")*

**13. Làm thế nào để lấy được dữ liệu từ component con**

- Truyền DL từ component con 🡪 component cha: dùng @Output ()

- Khi dùng @Output () ta phải tạo sự kiện = EventEmitter

**- Luồng đi**:

* Đặt decorator @Output cho đối tượng event emitter
* Sử dụng event binding tại component cha
* Tạo một đối tượng EventEmitter tại component con
* Gọi phương thức emit của event emitter vào thời điểm cần tung event.

*Ở đây khi bấm vào đặt hàng sẽ gọi đến phương thức yeucaudathang() của Component con, giờ muốn là khi bấm vào đó thi thông tin cũng sẽ gửi tới Component cha. Nếu vậy, Component con sẽ cần phát đi một sự kiện, dữ liệu sẽ truyền đi theo sự kiến đó để giúp cho Component cha có thể bắt được.*

*Component con sẽ định nghĩa sự kiện như sau: Đặt tên sự kiện là dathangEvent, tham số truyền đi là đối tượng Product, vậy sẽ khai báo như sau trong Component con*

*@Output() dathangEvent = new EventEmitter<Product>();*

*Đã có sự kiện rồi, khi nào muốn phát sự kiện đó thì gọi của sự kiện emit(), ta sẽ gọi nó trong phương thức yeucaudathang (tương ứng bấm vào nút)*

*Như vậy, mỗi khi bấm vào nút Đặt hàng thì Component con đã phát ra sự kiện tên là dathangEvent, Component cha muốn bắt được sự kiện chỉ việc lắng nghe - hãy xem ví dụ trước bắt sự kiện dathangEvent thì cũng tương tự như bắt sự kiện click, keyup ...*

**14. Sự khác biệt giữa RouterModule.forChild và RouterModule.forRoot là gì?**

- Phương thức RouterModule.forRoot: sử dụng khi xác định cấu hình gốc cho ứng dụng trong mô-đun chính.

=> Điều này cho phép mô-đun chính truy cập vào tất cả các chỉ thị của bộ định tuyến cũng như xác định cấu hình chính.

- RouterModule.forChild khá giống với forRoot, có thể được sử dụng giống như mô-đun tính năng thay vì mô-đun chính (gốc).

**- Sử dụng mô-đun “gốc” forRoot, sử dụng mô-đun “con”forChild**

**15. Tại sao nên sử dụng Angular (điểm mạnh)?**

- Là công cụ giúp chúng ta xây dựng website, nó là 1 framework giúp cho việc phát triển front end nhanh chóng và tiện lợi hơn

**- Lợi ích:**

* Tiết kiệm chi phí
* Dễ dàng nâng cấp từ Angular JS
* Thống nhất về các tiêu chuẩn code, chạy tốt trên các trình duyệt khác
* Mã nguồn mở, miễn phí.
* Cộng đồng sử dụng đông.
* Được sự hậu thuẫn của Google.
* Hỗ trợ Single Page Application.
* Tăng tốc độ phát triển.

**16. Nhược điểm của Angular là gì?**

**- Hạn chế:**

* Các khái niệm khó hiểu, khó học, phức tạp
* Không bảo mật do đây là ngôn ngữ font-end.
* Phụ thuộc vào js nên nếu trình duyệt tắt js sẽ không hoạt động được
* TypeScript và Decorators
* Module API: các khái niệm import, export, bootstrap,…
* Multiple Modules: cấu hính phải đúng
* Các thông báo lỗi khó hiểu
* RxJS: xử lý các hoạt động bất đồng bộ thay thế cho promises nhưng có nhiều tính năng hơn; phức tạp hơn Promises

**17. Angular Cli là gì? Làm sao để sử dụng nó ?**

- Angular CLI là một công cụ được phát triển đề chạy trên giao diện cửa sổ ứng dụng nhằm giúp đỡ việc tạo dự án, quản lý tập tin trong dự án và thực hiện nhiều tác vụ khác nhau như test (kiểm thử), bundle và deploy dự án một cách nhanh chóng và hiệu quả.

- Lệnh tạo: **npm install -g @angular/cli**

**18. Directive trong angular là gì ?**

- Directive là một khái niệm trong Angular Framework, nó là những thành phần mở rộng cho các thẻ html dùng bổ trợ các thuộc tính nâng cao cho các thẻ html.

- Là thành phần của Angular thường dùng để tăng sức mạnh cho template.

- Directive có chức năng thay đổi thuộc tính, cấu trúc của HTML

- Gồm:

1. Component directive
2. Attribute directive : ngClass, ngStyle … *(Được dùng như một thuộc tính của đối tượng, cho nên khi build thì directive và các thuộc tính thông thường khác được build cùng một lúc dẫn đến dự thay đổi của directive là đồng thời khi render đối tượng đó)*
3. Structural directive: \*ngIf, \*ngFor…(thay đổi cấu trúc của HTML)
4. Custorm directive : directive do người dùng tự định nghĩa

**ng g directive tên của directive**

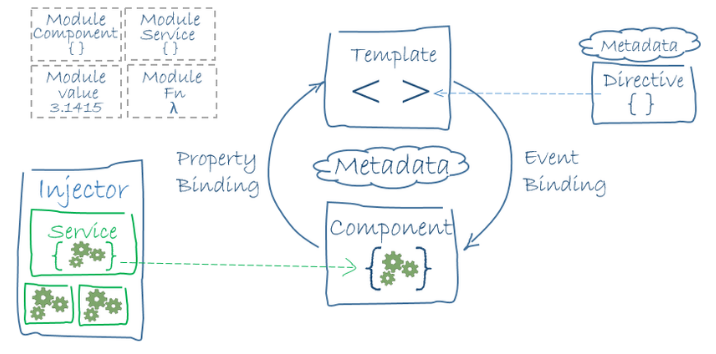
**19. Trình bày được khái niệm template trong angular**

- Component cần một view để hiển thị 🡪Template định nghĩa view.

- Template chỉ là một tập con của HTML, nó chỉ cho Angular biết làm sao để hiển thị view. Nó là một trang HTML chuẩn sử dụng các thẻ h1, h2...Nó cũng sử dụng các lệnh của Angular như {}, []...

- Template là giao diện view sử dụng ngôn ngữ Html. Sử dụng data binding và directive để tăng sức mạnh cho view.

**20. Giải thích kiến trúc của Angular**



Gồm 8 thành phần:

1. Module
2. Component
3. Template
4. Metadata
5. Data Binding
6. Service
7. Directive
8. Dependency Injection

**21. Hãy mô tả khái niệm Databinding One Way & two way phân biệt sự khác nhau?**

* **One way binding (binding 1 chiều):** dữ liệu được truyền 1 chiều. (Có thể từ view sang component hoặc ngược lại từ component sang view)
* Component 🡪 view:
* Interpolation : **{{ variable }}**
* Property Bingding : **<input [value] = ‘variable’>**
* View 🡪 Component:
* Property Binding
* Event Binding chúng ta sử dụng để bind các sự kiện như click chuột, sự kiện bàn phím …

**<input (event) = ‘tên function ($event.target.value)’>**

* **Two way binding (binding 2 chiều):** chúng ta thay đổi dữ liệu từ component qua view và ngược lại từ view chúng ta thay đổi dữ liệu, hữu dụng khi mình làm form. Chúng ta sử dụng ngModel để thực hiện việc binding 2 chiều.

**<input [(ngModule)] = ‘variable’>**

**22. ngModule là gì?**

- Angular app được chia thành các module nhỏ được gọi là NgModule

- Mỗi 1 NgModule có thể có: component, service, pipe directice…

- Mỗi 1 NgModule có thể import các tính năng của NgModule khác và có thể export tính năng của bản thân cho NgModule khác sử dụng

**23. Component là gì ? các thành phần của component**

- Component là một lớp nó thường hiểu như là controller tương tác giao diện người dùng.

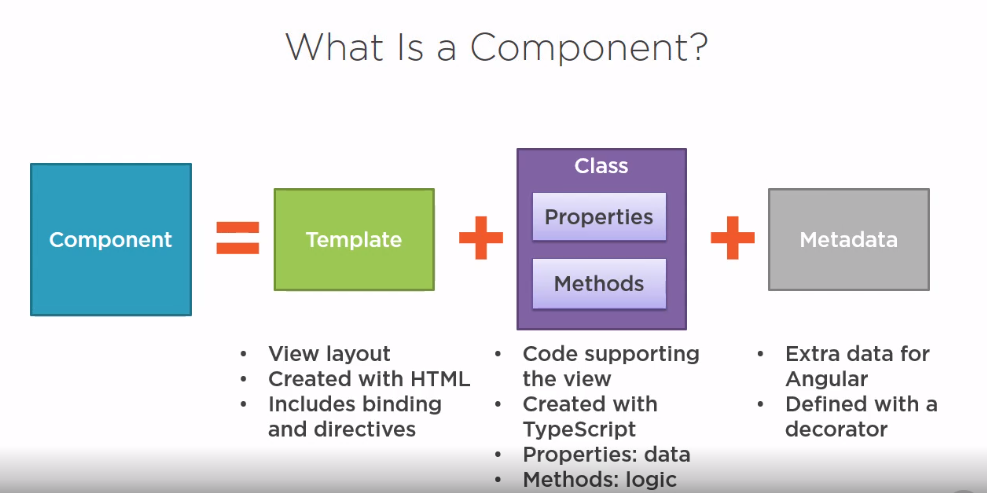
- Một component thường có 3 thành phần chính:

* Class: viết bằng typescript. Chứa các thuộc tính và phương thức được dùng để xử lí logic.
* Template HTML : tạo giao diện được viết bằng Html
* Metadata: là hàm dùng để bổ sung thông tin. (selector, file html, css…)

- Để tạo Component nhanh chóng thì dùng decorator có tên là @Component

*(Một decorator là một chức năng (hàm) để thêm các siêu dữ liệu (metadata), qua đó thêm thuộc tính, phương thức cho một lớp đã được định nghĩa trước. Các decorator của Angular có tiền tố là @ như: @Component , @NgModule, @Directive, @Injectable ...)*

- Trong 1 ứng dụng Angular thì tất cả các thành phần hiển thị trên trang web sẽ chia nhỏ thành các component riêng, mỗi 1 component sẽ có 1 file class dùng để xử lý business. 1 metadata dùng để mô tả cho component và 1 thành phần template để hiển thị giao diện



**24. Có thể chuyển mã TypeScript thành mã Javascript không, bằng cách nào?**

**-** Dùng câu lệnh tsc trỏ đến file TS

**25. Mô tả vòng đời component?**

- Vòng đời của một component hay một directive trong Angular tính từ lúc nó được tạo ra, nó bị thay đổi và bị phá huỷ.

*1. Phương thức ngOnChanges*

- Được gọi khi component phát hiện có giá trị được binding vào component = phương pháp Input properties. Để nhận biết được giá trị binding thì ngOnChange quản lý đối tượng SimpleChanege. Chúng ta sử dụng @Input trong bài truyền giá trị từ cha xuống con

*2. Phương thức NgOninit*

- Được gọi, khi component được tạo lần đầu tiên. Chúng ta được chạy sau khi hàm constructor và hàm ngOnchange được thực hiện.

*3. Phương thức ngOnDoCheck*

- Được gọi mỗi khi nó phát hiện ra có sự thay đổi dữ liệu ở component.

*4. ngAfterContentInit*

- Được gọi sau khi component được khởi tạo thành công

*5. ngAfterContentChecked*

*6. ngAfterViewInit*

- Được gọi khi component và các component con của nó được khởi tạo thành công. Chỉ được gọi 1 lần sau khi ngAfterContentChecked

*7. ngAfterViewChecked*

- Được gọi sau khi các view component cha và các view của component con được kiểm tra xong.

*8. ngOnDestroy*

- Được gọi trước khi component hoặc directive bị phá huỷ bởi Angular

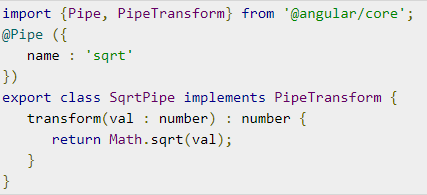
**26. Pipe là gì?**

- Được dùng để chuyển đổi DL mà bạn hiển thị lên template, nhận vào 1 đầu và cho ra kết quả ở 1 đầu (đầu vào và đầu ra có thể giống nhau)

***Cú pháp***: **{{ dữ liệu cần chuyển đổi | pipe }}**

- Một số pipe có sẵn:

* DatePipe {{ dữ liệu | date: ‘dd/mm/yyyy’ }}
* JsonPipe {{ dữ liệu | json }}
* LowerCasePipe
* UpperCasePipe
* TitleCasePipe (viết hoa chữ các đầu)
* SlicePipe (cắt chuỗi) {{ dữ liệu | slice 1:5 }} *(lấy từ vị trí 1 đến hết 4)*
* DecimalPipe {{ dữ liệu | number: ‘3.4-4’ }} *(trước dấu phẩy có 3 chữ số, sau dấu phẩy có 4 chữ số)*
* PercentPipe (%)
* CustormPipe



**27. Phân biệt sự khác nhau giữa Promise và Observable**

***- Giống:*** làm việc với các chức năng không đồng bộ trong JavaScript

***- Khác:***

|  |  |  |
| --- | --- | --- |
|  | **Promise** | **Observable** |
|  | - Gồm:  Resolve () : thành công  Reject () : thất bại | - Gồm:  Next (): (data) => {} : xử lý khi có DL  Error: (error) => {}: xử lý khi có lỗi  Complete: () => {} : xử lý khi thành công |
| **Khả năng trả về nhiều kết quả** | - Sau mỗi lần xử lý dữ liệu thì nó chỉ có thể trả về một giá trị duy nhất. | - Nó có thể trả về nhiều giá trị đồng thời. |
| **Khả năng hủy bỏ Request** | - Khi chúng ta đã tạo một request thì không có cách nào hủy bỏ request đó. | - Observable thì có khả năng hủy bỏ request đã tạo. |
| **Khả năng Retry** | - Promise không thể retry khi quá trình xử lý xảy ra lỗi, hoặc phải nhờ vào các tips, thư viện hỗ trợ của bên thứ 3. | - Observable cung cấp các operators retry, retryWhen giúp việc retry các request rất dễ dàng. |
| **Khả năng xử lý kết quả trước khi trả về** | - Do đặc tính chỉ trả về một kết quả duy nhất nên việc xử lý kết quả có thể thực hiện ở bước sau khi đã nhận dc kết quả. Ko xử lý | - Nó cung cấp rất nhiều các operators mạnh mẽ, phục vụ cho việc xử lý kết quả trước khi trả về. Xử lý kết quả trước khi hoàn thành |

**28. Biên dịch AOT là gì? Những lợi thế của AOT ? (AOT – Ahead of time: trước thời hạn)**

**- Biên dịch AOT:** Nó biên dịch tất cả các mã tại thời điểm xây dựng ứng dụng.

**- Lợi thế:**

AOT - Biên dịch TypeScript trong giai đoạn xây dựng.

* Kết xuất nhanh hơn *(trình duyệt có thể dễ dàng kết xuất ứng dụng bằng cách tải mã đã được biên dịch)*
* Được biên dịch bởi chính máy, thông qua dòng lệnh (Nhanh hơn).
* Tất cả các mã được biên dịch cùng nhau, nội tuyến HTML / CSS trong các tập lệnh.
* Không cần triển khai trình biên dịch (Một nửa kích thước góc).
* An toàn hơn, nguồn gốc không được tiết lộ.
* Thích hợp cho xây dựng sản xuất.
* Ít yêu cầu bất đồng bộ hơn.
* Phát hiện lỗi template sớm.

**29. Trình bày khái niệm Front-end & backend**

- ***front-end*** là một phần của hệ thống [phần mềm](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m), tương tác trực tiếp với người sử dụng. Cụ thể, đó là hệ thống các giao diện người dùng (GUI) và lập trình phía người dùng.

Gồm: HTML, CSS, JS ….

*-* ***Back-end*** (phần lập trình trên server) gồm có các thành phần để xử lý các thông tin từ Front-end, tương tác với [DBMS](https://vi.wikipedia.org/w/index.php?title=DBMS&action=edit&redlink=1) (hệ quản trị dữ liệu). Gồm: Java, PHP, …

**30. Làm sao để cài đặt TypeScript**

- Lệnh tạo: npm init typescript –project

- config trong file ‘tsconfig.json’ những thông tin về phiên bản ES, thư mục sau khi build :

{

“compilerOptions” : {

“outDir” : “build”,

“target” : “es6”,

“watch” : true

}

}

- tsc –watch: tự build file JS

**31. Để khởi tạo một dự án angular sử dụng câu lệnh nào**

ng new name\_project

**32. Để chạy một dự án angular sử dụng câu lệnh nào**

ng s --o

**33. Kể tên các kiểu dữ liệu trong Typescript**

- Có đầy đủ các kiểu DL của JS (null, undefined, boolean, number, string, symbol, object), ngoài ra Typescript còn có các kiểu DL như:

* Enum: là một kiểu dữ liệu đặc biệt cho phép một biến có thể là một tập hợp các hằng số định sẵn.
* Tuple: có thể chứa hai giá trị của các kiểu dữ liệu khác nhau.

*var employee: [number, string] = [1, 'Steve']*

* Any: có thể lưu trữ bất kỳ kiểu dữ liệu nào.
* Interface: cho phép bạn định nghĩa thuộc tính là gì và phương thức là gì mà đối tượng cần để được thực thi (implement). Nếu đối tượng tuân thủ đúng khuôn mẫu interface thì đối tượng đã implement interface ấy sẽ được thi hành đúng. Nếu interface không được thi hành đúng đắn thì typescript sẽ phát sinh lỗi ngay lập tức.

**34. Kể tên các vòng lặp được typescript hỗ trợ**

[1. Vòng lặp for in](https://freetuts.net/vong-lap-trong-typescript-749.html#goto-h2-0) (lấy index)

[2. Vòng lặp for](https://freetuts.net/vong-lap-trong-typescript-749.html#goto-h2-1)

[3. Vòng lặp while và do while](https://freetuts.net/vong-lap-trong-typescript-749.html#goto-h2-2)

* [Vòng lặp while](https://freetuts.net/vong-lap-trong-typescript-749.html#goto-h3-0)
* [Vòng lặp do while](https://freetuts.net/vong-lap-trong-typescript-749.html#goto-h3-1)

[4. Vòng lặp for of](https://freetuts.net/vong-lap-trong-typescript-749.html#goto-h2-3) (lấy value)

**35. ngFor là gì? Cách sử dụng**

- ngFor: để duyệt qua các tập hợp dữ liệu như array, list. Sau đó ta tạo các thành phần web tương ứng với mỗi vòng lặp.

- Cú pháp: \*ngFor = ‘let student of studentList’

- *Cách sử dụng:*

1. Duyệt từng thành phần trong mảng
2. **Sử dụng NgFor mảng trong mảng *(****Ví dụ: ta có danh sách các nhân viên, trong mỗi nhân viên lại có danh sách các skills)*
3. **Sử dụng Index :** Chúng ta có thể sử dụng biến local index trong ngFor để xem số thứ tự của mỗi phần tử trong mảng.

\*ngFor="let movie of movies; let i=index;

1. **Format dòng chẳn dòng lẻ trong table**

\*ngFor="let movie of movies; let i=index; let o= odd; let e=even;"

[ngClass]="{ odd: o, even: e }"

1. **Lấy phần tử đầu và cuối trong mảng**

\*ngFor="let movie of movies; let i=index; let first= first; let last=last;"

[ngClass]="{ first: first, last: last }"

**36. ngIf là gì? Cách sử dụng**

- Sử dụng để xoá hoặc thêm một phần tử trên web dựa vào điều kiện có thoả mãn hay không

- Cú pháp**: \*ngIf="condition"**

*- Cách sử dụng:*

1. Sử dụng phủ định trong ngIf

2. Sử dụng if else trong ngIf *(\*ngIf="condition; else elseBlock")*

3. Sử dụng then else trong ngIf *(\*ngIf="condition; then thenBlock else elseBlock")*

**37. Các cách tạo form trong Angular**

- Có 2 cách tạo form trong Angular:

1. Template – driven – form
2. Model – driven – from *(hay còn được gọi là Reactive Form)*

|  |  |
| --- | --- |
| **Model – driven – from**  ***(Reactive Form)*** | **Template – driven – form** |
| - Tạo form từ component TS và binding data tới template | - Tạo form từ HTML binding data tới file TS component |
| - Đồng bộ | - Bất đồng bộ |
| - import: Reactive Form Module | - import: Form Module |

**38. Router trong angular dùng để làm gì?**

- Router: là thành phần giúp điều hướng trong angular, quy định component hiển thị ứng dụng với url

- Config router trong file : routing.module.ts

const routes: Routes = [  
 {path: '', component: HomeComponent},  
 {path: 'employee-list', component: EmployeeListComponent},  
 {path: 'employee-add', component: EmployeeAddComponent},  
 {path: 'employee-edit/:id', component: EmployeeEditComponent},  
  
 {path: '\*\*', component: PageNotFoundComponent}  
];

- Đối tượng Router dùng để chuyển hướng trang web

**<router – outlet> :** đánh dấu vị trí view component được render

**39. Cách để lấy tham số trên URL trong angular**

constructor(private route: ActivatedRoute) {  
 this.route.params.subscribe(params => {  
 });  
}

- Sử dụng ActicatedRoute lấy giá trị của tham số trên URL

**40. Làm sao để kiểm tra người dùng có quyền truy cập vào 1 route ?**

- Route Guard: là interface có thể cho bộ định tuyến biết liệu nó có cho phép điều hướng đến 1 route được yêu cầu hay không.

* CanActivate
* CanActivateChild
* CanDeactive
* CanLoad
* Resolve

**41. Lazy-loading là gì?**

- Lazy loading (lười tải, tải chậm): tức là chỉ tải hình ảnh khi cần thiết,

VD: kKi người dùng scroll đến đâu bạn sẽ load dữ liệu đến đó. lazy loading có thể áp dụng cho bất cứ resource nào trên 1 page, thậm chí là cả file JavaScript.

**Ưu điểm**

- Kỹ thuật tải hiệu quả nhất được sử dụng phổ biến ngày nay.

- Không mất chi phí băng thông cho những gì bạn không tải xuống, không xem, không nhìn thấy.

**Nhược điểm:**

- Cách tiếp cận này có thể dẫn đến sự nhấp nháy của phần nội dung bị thiếu.

- Quá trình thực hiện lazy loading có thể gặp một vài rắc tối trong thao tác. Thực tế, phương pháp này luôn đòi hỏi nhiều JavaScript hơn các phương pháp khác. Nhiều JavaScript hơn có nghĩa là phức tạp hơn, ngốn nhiều thời gian hơn, từ đó các lỗi sai cũng có nguy cơ xảy ra hơn.

- Nếu tất cả JavaScript không được tải xuống qua một kết nối 3G chậm hoặc nếu tất cả JavaScript không được thực thi, thì hình ảnh có thể không hiển một chút nào. Nếu lazy loading không hoạt động, sẽ chẳng ai có thể xem các Images below the fold hay above the fold, hay bất cứ image nào cả.

**42. Ưu điểm của việc bind event(ngSubmit) thay vì (submit)?**

- Nhờ form không gửi request đi khi handler method có exception, tránh thay đổi trạng thái dữ liệu tại backend không mong muốn, trang không bị render lại, tránh hiện tượng "nháy"

**43. Khác nhau giữa onclick và click ?**

- onclick: là sự kiện trên HTML

- click: là sự kiện trên Angular