**CÂU HỎI AUDIT MODULE 5**

|  |  |
| --- | --- |
| **STT** | **Q&A** |
| 1 | **Q:** TypeScript là gì?  🡺 **A:**  - Là một NNLT được phát triển bởi Microsoft  - Là superset của JS  - LTV có thể transpile từ TS sang JS. Cách transpile: tsc dương\_dan/ten\_file.ts  - TS ra đời nhằm hỗ trợ thêm cho JS các phần như OOP, DataType, Generic,… |
| 2 | **Q:** Tại sao nên sử dụng Typescript, Ưu điểm của Typescript  🡺 **A:**  - Vì nó được thiết kế thêm nhiều chức năng tiện lợi, cải tiến những điểm yếu của JS, có thể hoạt động rộng rãi cho các ứng dụng của ngôn ngữ Angular2 và Nodejs  - Ưu điểm:  + Hỗ trợ quá trình tìm kiếm giúp tiết kiệm thời gian kiểm tra lại code  + Hoàn toàn miễn phí  + Thao tác khá đơn giản, tiết kiệm thời gian, khắc phục tình trạng xuất hiện lỗi, dễ đọc hơn và chỉnh sửa code dễ dàng hơn |
| 3 | **Q:** Làm sao để cài đặt TypeScript  🡺 **A:**  - B1: Cài đặt NodeJS  - B2: Mở cmd  - B3: Gõ lệnh npm -v để in ra version của npm  - B4: Cài đặt TS bằng cách gõ lệnh npm install -g typescript |
| 4 | **Q:** Có thể chuyển mã TypeScript thành mã Javascript không, bằng cách nào?  🡺 **A:**  - Được. Cách chuyển: tsc dương\_dan/ten\_file.ts |
| 5 | **Q:** Kể tên các kiểu dữ liệu trong Typescript  🡺 **A:**  - TS hỗ trợ tất cả các kiểu dữ liệu của JS và bổ sung thêm một số kiểu dữ liệu khác  như: any (một kiểu dữ liệu mà chúng ta không biết chắc chắn kiểu dữ liệu của nó),  interface, enum, tuple (khai báo mảng) |
| 6 | **Q:** Kể tên các vòng lặp được typescript hỗ trợ  🡺 **A:**  - While, for, for … of, for … in, do … while |
| 7 | **Q:** Phân biệt sự khác nhau giữa var, let, const  🡺 **A:**   |  |  |  | | --- | --- | --- | | Var | Let | Const | | - Khai báo những giá trị có thể thay đổi  - Cho phép khai báo lại biến với cùng 1 tên  - Việc khai báo được đẩy lên đầu chương trinh  - Nếu var được khai báo ở block scope, thì vòng đời nằm ở trong lẫn ngoài block scope | - Khai báo những giá trị có thể thay đổi  - Không cho phép khai báo lại biến với cùng một tên  - Không có cơ chế hoisting  - Phạm vi của let chỉ ở block scope | - Dùng để khai báo hằng số, không thay đổi giá trị tham chiếu | |
| 8 | **Q:** Trình bày khái niệm Single page application  🡺 **A:**  - Là một ứng dụng web hay một website mà ở đó tất cả các thao tác của người dùng chỉ diễn ra trên 1 trang duy nhất, tất cả các cấu trúc của trang web (HTML) được load sẵn 1 lần và sẽ không load lại ngay cả khi chuyển trang. |
| 9 | **Q:** Lazy-loading là gì?  🡺 **A:**  - Là kỹ thuật thực hiện trì hoãn tải các tài nguyên không quan trọng vào thời điểm tải trang. Thay vì tải ngay lập tức, các tài nguyên không quan trọng này chỉ tải vào thời điểm cần thiết. |
| 10 | **Q:** Để khởi tạo một dự án angular sử dụng câu lệnh nào ?  🡺 **A:**  - Lệnh: ng new ten\_du\_an |
| 11 | **Q:** Để chạy một dự án angular sử dụng câu lệnh nào ?  🡺 **A:**  - Lệnh: ng s --o |
| 12 | **Q:** Tại sao nên sử dụng Angular (điểm mạnh)?  🡺 **A:**  - Angular giúp nâng cao năng suất của các LTV  - Cấu trúc phát triển rõ ràng  - Hỗ trợ đầy đủ tính năng điều hướng (**routing**)  - Giảm tối đa kích thước và tăng tối đa hiệu suất của ứng dụng. |
| 13 | **Q:** Nhược điểm của Angular là gì?  🡺 **A:**  - Không an toàn. Angular là một trong những Framework Front-end, mà front-end không bảo mật bằng Back-end  - Với một số trinh duyệt sở hữu tính năng Disable JavaScript thì website sẽ không thể sử dụng được trên trinh duyệt đó |
| 14 | **Q:** Angular Cli là gì? Làm sao để sử dụng nó ?  🡺 **A:**  - 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.  - Sử dụng Angular Cli  + B1: Cài đặt môi trường Node.js  + B2: Mở CMD và gõ câu lệnh npm -v  + B3: Chạy câu lệnh npm install -g @angular/cli |
| 15 | **Q:** File angular.json và package.json sử dụng để làm gì?  🡺 **A:** |
| 16 | **Q:** Giải thích kiến trúc của Angular  🡺 **A:**  - Component là một class JS thuần và định nghĩa bởi @Component decorator. Decorator này cung cấp cho component thông tin về View (template) và thông tin metadata về class  - Component sử dụng data binding để lấy dữ liệu từ Component sang View (template), sử dụng HTML đặc biệt là Template  - Service cung cấp các service cho Component  - Trách nhiệm cung cấp các thể hiện của Service cho Component gọi là Injector. Nó jnject các service vào component sử dụng Dependency Injection  - Sử dụng Module để tổ chức code ứng dụng Angular. Ứng dụng sẽ gộp lại bởi nhiều Module. Mỗi Module triển khai một tinh năng cụ thể hoặc các tinh năng của ứng dụng. |
| **17** | **Q:** Angular đi theo mô hình nào? Phân biệt mô hình MVVM và MVC  🡺 **A:**  - Angular đi theo mô hình MVVM  - Phân biệt:   |  |  | | --- | --- | | MVC | MVVM | | - Một đến nhiều mối quan hệ giữa Controller & View.  - Chế độ xem không có tham chiếu đến Bộ điều khiển  - Là mô hình cũ  - Khó đọc, khó thay đổi, khó kiểm tra đơn vị và khó sử dụng lại mô hình này  - Thành phần mô hình MVC có thể được kiểm tra riêng biệt với người dùng | - Một đến nhiều mối quan hệ giữa View & View Model.  - Chế độ xem có tham chiếu đến Chế độ xem - Mô hình.  - Là một mô hình tương đối mới.  - Quá trình gỡ lỗi sẽ phức tạp khi có các ràng buộc dữ liệu phức tạp.  - Dễ dàng kiểm tra đơn vị riêng biệt và mã theo hướng sự kiện. | |
| 18 | **Q:** Component là gì ? các thành phần của component  🡺 **A:**  - Là thuật ngữ mang ý nghĩa như là bộ phận, thành phần hoặc yếu tố cấu thành của một hệ thống hoặc phần mềm nào đó, được gọi ra từ các chương trình khác, được liên kết và sử dụng như một bộ phận của chương trình.  - |
| 19 | **Q:** Mô tả vòng đời component?  🡺 **A:**  - Khi ứng dụng Angular được start lên thì đầu tiên nó sẽ tạo và render component cha (hay còn gọi là root component) sau đó nó sẽ tạo và render các component con. Khi mỗi component được load lên, component sẽ kiểm tra xem có data binding vào nó không, dữ liệu có thay đổi không và cập nhật lại chúng. Khi component bị phá huỷ thì chúng sẽ bị remove (xoá) khỏi giao diện web.  - Thứ tự từ trên xuống dưới về các method sẽ được sử dụng trong vòng đời của một component.  + ngOnChanges  + ngOnInit  + ngDoCheck  + ngAfterContentInit  + ngAfterContentChecked  + ngAfterViewInit  + ngAfterViewChecked  + ngOnDestroy |
| 20 | **Q:** Trình bày được khái niệm template trong angular  🡺 **A:**  **-** Template được xem là bộ khung xương trong website,  chúng được thiết kế và sấp xếp thành một bố cục cụ thể giúp người sử dụng có thể dễ dàng thao tác thay đổi mà không cần biết code. |
| **21** | **Q:** Biên dịch AOT là gì? Những lợi thế của AOT ?  🡺 **A:**  - Biên dịch trước thời gian (AOT) là khi mã được biên dịch như một phần của quá trình xây dựng  - Những lợi thế:  + Với AOT, mã được biên dịch trong quá trình xây dựng nên kết xuất ứng dụng Angular nhanh hơn  + Không cần tải xuống trình biên dịch Angular vì ứng dụng đã được biên dịch nên kích thước ứng dụng Angular nhỏ hơn  + Chất lượng mã tốt hơn  + Các ứng dụng an toàn và mạnh mẽ hơn |
| 22 | **Q:** Trình bày về Data binding trong component  🡺 **A:**  - Là kỹ thuật, nơi dữ liệu được đồng bộ giữa component và tầng view (template file html). Data binding trong Angular có thể chia ra làm 2 nhóm. Đó là one way binding (binding 1 chiều) và two way binding (binding 2 chiều). |
| 23 | **Q:** Hãy mô tả khái niệm Databinding One Way và Two Way phân biệt sự khác nhau?  🡺 **A:**   |  |  | | --- | --- | | Data binding One Way | Data binding Two Way | | - Dữ liệu được truyền 1 chiều. Có thể từ view sang component hoặc ngược lại từ component sang view.  - Từ component sang view chúng ta sử dụng Interpolation & Property Binding để hiển thị dữ liệu  - Sử dụng để hiển thị giá trị từ component sang view. | - Là thay đổi dữ liệu từ component qua view và ngược lại từ view chúng ta thay đổi dữ liệu.  - 2 way binding thì hữu dụng khi làm form.  - Sử dụng ngModel để thực hiện việc binding 2 chiều. | |
| 24 | **Q:** Directive trong angular là gì ?  🡺 **A:**  - Là một thanh phần của Angular Framework dùng để tăng cường sức mạnh và hỗ trợ thêm các tinh năng cho HTML. Directive giúp LTV có thể thay đổi được cấu trúc và thuộc tinh của HTML |
| 25 | **Q:** Kể tên các directive được hỗ trợ trong angular  🡺 **A:**  - Component Directive: bản thân mỗi component chính là một directive  - Structural Directive: là các directive dùng để thay đổi cấu trúc của HTML: “ngFor”, “ngIf”  - Attribute Directive: là các directive dùng để thay đổi thuộc tính của HTML: [style], [class]  - Custom Directive: là directive do LTV tự định nghĩa trong dự án |
| 26 | **Q:** Làm thế nào để lấy được dữ liệu từ component cha  🡺 **A:**  - B1: Đặt @Input cho biến x cần nhận dữ liệu từ component cha  - B2: Sử dụng directive [x] = “value” trong selector của component con |
| 27 | **Q:** Làm thế nào để lấy được dữ liệu từ component con  🡺 **A:**  - C1: Sử dụng @Output và Emitter  + B1: Khai báo @Output cho biến x, có kiểu dữ liệu là EventEmitter  + B2: Gọi hàm emit của Emitter để gửi dữ liệu lên component cha  + B3: Sử dụng (x) = “nameFunction($event)” với event chinh là dữ liệu component con gửi lên  - C2: Sử dụng biến cục bộ  - C3: Sử dụng @ViewChild |
| 28 | **Q:** Pipe là gì? Kể tên 1 số pipe thường sử dụng  🡺 **A:**  - Là thành phần của Angular dùng để thay đổi cách hiển thị của dữ liệu trên giao diện (lưu ý: pipe không làm thay đổi dữ liệu gốc)  - Các loại pipe thường dùng: titlecase, datapipe, currency pipe, decimal pipe, … |
| 29 | **Q:** ngFor là gì? Cách sử dụng  🡺 **A:**  - Sử dụng ngFor để duyệt qua các tập hợp dữ liệu như array, list. Sau đó tạo các thành phần web tương ứng với mỗi vòng lặp.  - Cách sử dụng:  <html-element ngFor="let <item> of <items>;">  <html-Template></html-Template>  </html-element>  - Trong đó:  + html-element: là thẻ web muốn sử dụng ngFor directive  + \*ngFor: dấu \* tượng trưng cho cú pháp của Angular  + let of <item>: là giá trị hiện tại của phần tử trong mảng, items chinh là mảng các phần tử |
| 30 | **Q:** ngIf là gì? Cách sử dụng  🡺 **A:**  - Sử dụng ngIf để xóa hoặc thêm một phần tử trên web dựa vào điều kiện có thoả mãn hay không.  - Cách sử dụng:  <p \*ngIf="condition">  các thẻ html sẽ được hiển thị nếu điều kiện là đúng  </p> |
| 31 | **Q:** Các cách tạo form trong Angular  🡺 **A:**  - Có 2 cách để tạo form: Template dirven form và Reactive form |
| 32 | **Q:** Phân biệt Template driven form và Reactive form  🡺 **A:**   |  |  | | --- | --- | | Template driven form | Reactive form | | - import FormModule  - Tạo form từ HTML  - Xử lý theo hướng bất đồng bộ  - Không được khuyến cáo sử dụng | - import ReactiveForm Module  - Tạo form từ TS - Xử lý theo hướng đồng bộ  - Được khuyến cáo nên sử dụng | |
| 33 | **Q:** Trình bày 1 số validate thường dùng. Cách custom validate như thế nào?  🡺 **A:**  - 1 số validate thường dùng:  + required dùng để check trống trường input  + email dùng để yêu cầu dữ liệu điền vào phải là email  + minlength: độ dài nhỏ nhất  + maxlength: độ dài lớn nhất  - |
| 34 | **Q:** Router trong angular dùng để làm gì?  🡺 **A:**  - Dùng để xác định component nào sẽ được hiển thị |
| 35 | **Q:** Cách để lấy tham số trên URL trong angular  🡺 **A:** |
| 36 | **Q:** Làm sao để kiểm tra người dùng có quyền truy cập vào 1 route ?  🡺 **A:** |
| 37 | **Q:**Trình bày sự khác nhau giữa navigateByUrl và navigate  🡺 **A:** |
| 38 | **Q:** Sự khác biệt giữa RouterModule.forChild và RouterModule.forRoot là gì?  🡺 **A:** |
| 39 | **Q:** Trình bày được ý nghĩa và khái niệm Observable  🡺 **A:**  - Là khái niệm đại diện cho một giá trị hoặc sự kiện sẽ phát sinh trong tương lai. Khi các giá trị hoặc sự kiện trong tương lai được trả về, nó sẽ được điều hướng vào observable (observer là một callback fuction có 3 tham số) |
| 40 | **Q:** Trình bày được ý nghĩa và khái niệm Observer  🡺 **A:**  - Là một behavioral design pattern (sau đây gọi tắt là DP) dùng để định nghĩa một các sự kiện xảy ra với đối tượng mà chúng đang quan sát (observe). |
| 41 | **Q:** Phân biệt sự khác nhau giữa Promise và Observable  🡺 **A:**   |  |  | | --- | --- | | Promise | Observable | | - Không có các khả năng như Observable | - Có khả năng trả về nhiều kết quả  - Có khả năng cancel ((hủy request đã gửi đi)  - Có khả năng retry request (gửi lại request)  - Có khả năng xử lý kết quả trước khi trả về (thông qua các hàm map, filler ..) | |
| 42 | **Q:** Trình bày được ý nghĩa và khái niệm subcribe  🡺 **A:**  - 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.** |
| 43 | **Q:** ngModule là gì?  🡺 **A:**  - **L**à cấu trúc cơ bản đầu tiên gặp khi làm việc với bất kỳ ứng dụng bằng Angular, nhưng nó cũng tương đối tinh vi và phức tạp, vì những scope của nó. Mục đích của NgModule là khai báo từng thứ được tạo trong Angular và nhóm chúng lại với nhau (như các package Java hoặc namespace PHP / C #). |