**Giới thiệu**

Cảm ơn đã tạo tạo cơ hội phỏng vấn.

Giới thiệu họ tên đầy đủ.

Năm sinh: tiện xưng hô

Đã tốt nghiệp trường nào, chuyên ngành là gì?

Kinh nghiệm đã làm: Js, Angular. Ngoài ra thì có kinh nghiệm về reactjs, laravel, vuejs.

Sở trường: phù hợp nhất với vị trí ứng tuyển.

Mong muốn: đc làm vị trí fullstack developer.

Cảm ơn để kết thúc phần giới thiệu.

1. **Chọn promise hay observable**

Link : <https://techtalk.vn/chon-promise-hay-observable-khi-lam-viec-voi-angular.html>

Đều để xử lí bất đồng bộ.

+ Trả về kết quả: Promise trả về 1 kết quả, Observable trả về nhiều kết quả

+ Khả năng hủy bỏ request (Promise ko có)

+ Khả năng retry khi xảy ra lỗi (Promise ko có)

+ Khả năng xử lí kết quả trước khi trả về (Promise có thể thực hiện ở bước then)

+ Observable là 1 khai báo, việc tính toán sẽ ko bắt đầu cho tới khi subcribe, Còn promise sẽ thực thi ngay khi đc tạo. -🡺 observable hữu ích là có thể chạy ở bất cứ đâu cần.

1. **Debounce và Throttle trong JS**

Link : <https://viblo.asia/p/debounce-va-throttle-trong-javascript-6J3ZgOpMZmB>

Để điều khiển 1 hàm đc gọi bao nhiêu lần trong khoảng thời gian nhất định.

Tăng performance của website và ngăn chặn việc gọi API ko cần thiết.

**Throttle**

Bất kể người dùng kích hoạt bao nhiêu lần, thì hàm cũng chỉ thực thi 1 lần kể từ khi người dùng kích hoạt sự kiện

Sử dụng trong sự kiện click để người dùng ko spam liên tục (call api)

**Debounce**

Bất kể người dùng kích hoạt bao nhiêu lần, thì hàm cũng chỉ thực thi 1 lần kể từ khi người dùng ngừng kích hoạt sự kiện.

Sử dụng trong sự kiện thay đổi kích thước window khi muốn bắt tọa độ cuối cùng mà người đã mong muốn

1. **Data binding trong Angular**

Là cơ chế lấy dữ liệu trong component và đưa ra view. Sử dụng cặp ngoặc nhọn {{…}}

1. **Binding 2 chiều trong Angular**

Tự động đồng bộ dữ liệu giữa component và view.

Trong angular có directive [(Ngmodel)] giúp đồng bộ dữ liệu Component và DOM

1. **NgTemplate**

Là 1 tính năng mới trong Angular 6.

Chỉ hoạt động khi được chỉ định với directive structural.

Khi biên dịch: Angular sẽ chuyển thẻ div thành <ng-template> và chuyển directive \*ngIf thành @Input[ngIf]



+ TH1: not render



+ TH2: render



1. **Template Reference Variable**

Chúng ta có thể truy cập tới giá trị thuộc tính của Dom element.

Được tạo bằng cách thêm dấu # vào thẻ HTML.

1. **Directive trong Angular**

<https://www.geeksforgeeks.org/angular-7-directives/>

Là 1 class được khai báo với decorator @directive

Có 3 loại:

+ Component directive (directive thành phần): chính là class khai báo @Component

* Selector: template tag
* templateUrl, template, styleUrl, style

+ Structural directive (directive cấu trúc): Quyết định Dom Element nào đc thực thi (\*ngIf, \*ngFor, \*ngSwitch)

+ Attribute directive (directive thuộc tính): Được sử dụng như 1 thuộc tính html (giống như thẻ a có thuộc tính href )

Thay đổi giao diện và hành vi DOM, chúng ta có thể custom directive

1. **Component dynamic trong angular**
2. **Local Storage, SessionStorage và Cookie**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Local Storage | SessionStorage | Cookie |
| Thời hạn lưu trữ | - Vô thời hạn  - Chỉ mất khi xóa bằng js, xóa bộ nhớ trình duyệt hoặc bằng API LocalStorage | - Không vô thời hạn  - Mất khi đóng tab | - Có thời gian sống  - Có thời gian timeout nhất định do lập trình hoặc mặc định sẽ giống session |
| Dung lượng | 5MB: Lớn nhất trong 3 loại | - Ít nhất 5MB | - Tối đa 4KB  - Vài chục cookie cho 1 domain |
| Gửi lên server | - Không | - Không | - Có  - Tải trang web => trình duyệt gửi cookie lên server |
| Bảo mật | - Tốt hơn Cookie | - Tốt hơn cookie | - Không  - Do có thể bị thay đổi, đánh cắp phía máy khách => ko lưu trữ thông tin quan trọng |
| Kiểm tra trình duyệt hỗ trợ | - Typeof (Storage) | - Typeof (Storage) |  |
| - Nơi sử dụng | - Chỉ đọc được phía máy khách | - Máy khách | - Chủ yếu đc đọc phía máy chủ( có thể cả máy khách) |

1. **Cache Storage**

- Lưu trữ dữ liệu tạm thời trong môi trường máy tính

- Giúp giảm thời gian truy cập , giảm độ trễ và cải thiện đầu vào/đầu ra (I/O) do hầu hết workload ứng dụng phụ thuộc vào hoạt động I/O

🡺 cache cải thiện hiệu suất ứng dụng.

VD: Browswer đọc dữ liệu cache nhanh hơn nhiều so với đọc lại dữ liệu từ webpage. 🡺 Khi back lại trang web thì trình duyệt có thể truy xuất hầu hết các file cần thiết từ bộ nhớ cache.

1. **State Management**

Link : <http://notcuder.com/quan-li-trang-thai-trong-angular/>

- Tại sao cần?

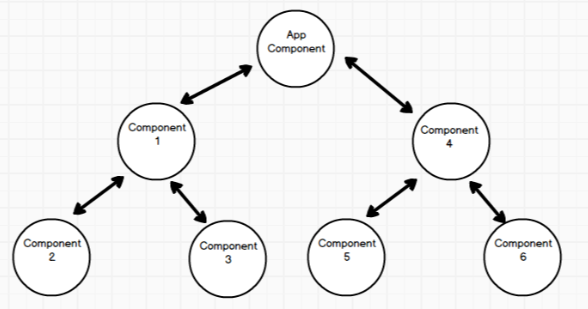
Trong Angular có cơ chế Binding 2 chiều => khi ứng dụng lớn, càng nhiều tương tác làm thay đổi giá trị biến 🡺 mất kiểm soát sự thay đổi của biến.

🡺 Cần một cơ chế quản lí trạng thái của ứng dụng ( trạng thái giao diện và trạng thái biến )

🡪 Cần State Management

🡪 Redux, Ngrx hoặc có thể là Observable

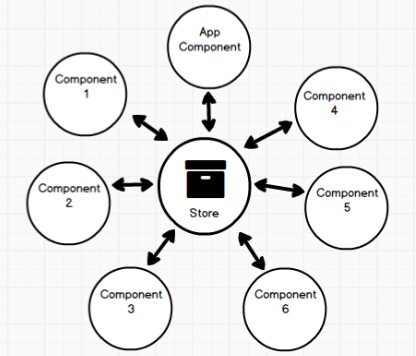
VD: Khi ứng dụng lớn, muốn truyền dữ liệu từ component 2 sang 5 thì sao? Cần truyền 3 lần.



1. **Cơ chế State Management trong Redux**

Thay đổi cách truyền dữ liệu: **Component-component** thành **Store-component**.

**+ 1 store duy nhất**



**+ State chỉ để đọc**

Để thay đổi state, một action phải đc tạo ra.

**+ Pure function reducers**

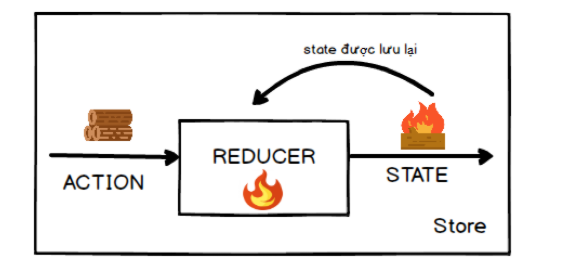
Là function ko bị thay đổi hay phụ thuộc vào các biến bên ngoài hàm.

Reducers phải đc viết bằng pure function

1. **Thành phần trong Redux**

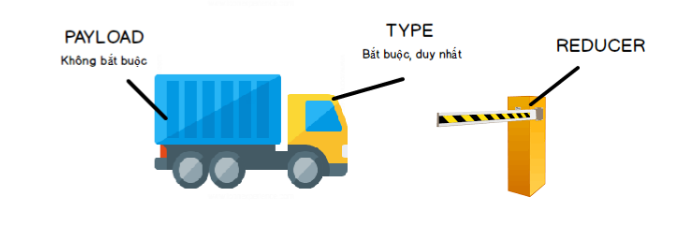
**+ Store**

Gồm Action, Reducer và State.



**+ Action**

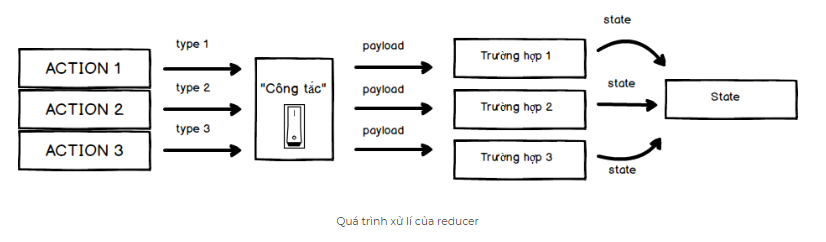
2 phần: Kiểu dữ liệu(type) và dữ liệu mang theo(payload)



**+ Reducer**

Là phần cốt lõi của store, thay đổi trạng thái ứng dụng.

Nhận action và state làm đầu vào và xử lí để tạo ra state mới.



1. NgFor

Keyword: last, first, even, odd as i

1. **Pipe trong angular**

Là cách ta chuyển hóa dữ liệu để hiển thị dữ liệu theo cách mong muốn

Chính là filters: short, date, decimal, currency, slice, json, number.

1. **Ngrx hay Redux?**

Link : <https://medium.com/@UReyesMeUp/differences-between-ngrx-and-angular-redux-d2d1d59d4cbd>

1. **Rxjs là gì?**

Link : <https://viblo.asia/p/rxjs-nhap-mon-oaKYMN1zR83E>

<https://viblo.asia/p/rxjs-va-reactive-programming-63vKj6oxK2R>

<https://luuxuantruong.info/rxjs-toan-tap-p1-gioi-thieu-rxjs-la-gi/>

Là một thư viện cho việc xử lí bất đồng bộ.

1. Các cách tạo services trong Angular, khi Component bị xóa khỏi Dom thì giá trị services đó như thế nào?
2. **Các cách share data trong Angular**

Link: <http://www.trandinhhung.com/angular/tip-tricks/sharing-data-between-angular-components>

<https://www.youtube.com/watch?v=I317BhehZKM>

+ Sử dụng @Input và @Output giữa component cha và con

+ Sử dụng ViewChild, ViewContent để truy xuất dữ liệu ban đầu của thành phần con

Chỉ đc sử dụng sau khi view tạo xong tức dùng trong hàm AfterViewInit()

+ Serivce: Sử dụng BehaviorSubject trong Rxjs

+ Ngoài ra: Ngrx, Redux

1. **Services trong Angular**

Là một class với mục đích: đóng gói một vài chức năng và chia sẻ cho phần còn lại của ứng dụng

+ Chia sẻ dữ liệu

+ Triển khai logic ứng dụng (chức năng)

+ Tương tác với bên ngoài

1. **Dependency Injection là gì?**

<https://www.tiepphan.com/thu-nghiem-voi-angular-dependency-injection-trong-angular/>

**Dependency là gì?**

Khi ta sử dụng class B trong class A thì tức là class A phụ thuộc vào class B

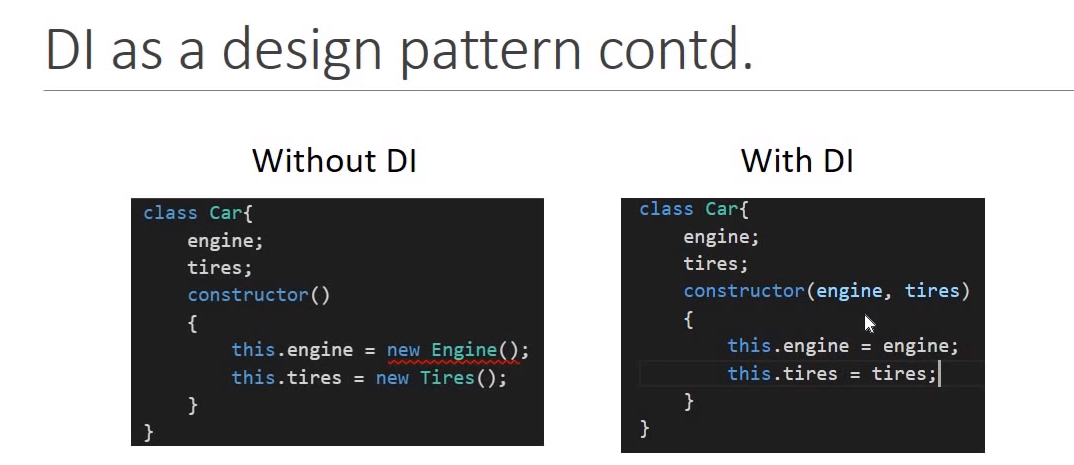
**Dependency Injection là gì?**

Là một Design Pattern.

Là một framework

Hiện tại chỉ có Angular là framework cung cấp DI.

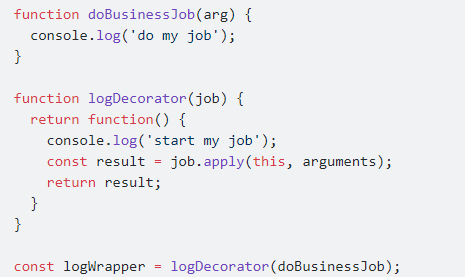
Class nhận sự phụ thuộc của nó từ bên ngoài (inject các phụ thuộc), thay vì tự tạo ra chúng.



1. **Decorator và Metadata Reflection trong TypeScript**

Link : <https://viblo.asia/p/typescript-decorator-va-metadata-reflection-m68Z0w2dKkG>

Ý tưởng: bao bọc 1 function trong 1 function khác.

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 ...

1. **Run time và Complie Time**

run time: là thời điểm chương trình đang chạy  
compile time: là thời điểm source code đang được compile

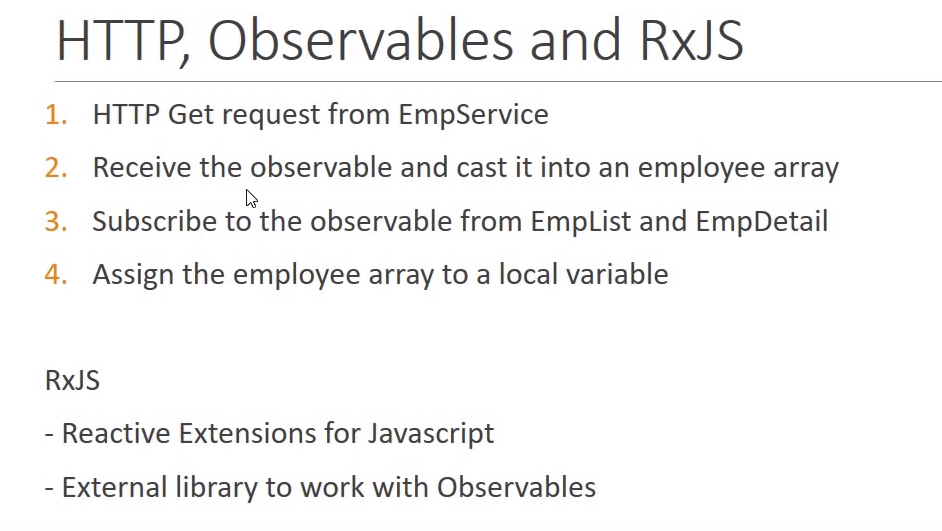
Thông thường run time và compile time được nhắc tới khi nói tới lỗi. Lỗi run time tức là lỗi khi chạy mới thấy, tức lỗi logic. Lỗi compile time là lỗi compile là thấy ngay, tức là lỗi cú pháp.

1. **Biên dịch và thông dịch?**

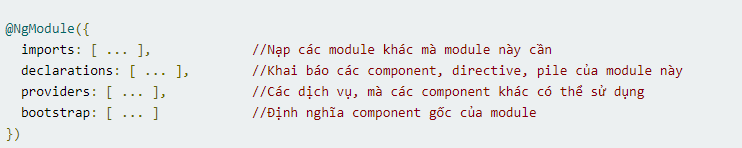
Biên dịch: chạy hết một lượt để thành mã máy(complier) rồi lưu vào ổ cứng để có thể thực thi lần sau

Thông dịch: chạy đến đâu sẽ thành mã máy tới đó.

1. **Luồng hoạt động với Http, Observables, Rxjs?**



1. Router và ActivatedRoute trong Angular
2. @NgModule, @Component, @Injectable, @Animation
3. redirectTo, pathMatch (full và prefix) trong router
4. navigate trong Angular
5. paramMap Observable trong Angular
6. ParamMap trong Angular
7. **@NgModule**



1. TabContent trong Angular
2. **Babel và polyfill là gì ?**

<https://viblo.asia/p/babel-dung-thi-sao-ma-khong-dung-thi-sao-Do754p1W5M6>

Check Compatition: <http://kangax.github.io/compat-table/es6/>

Nguyên nhân: JS có nhiều cú pháp mới nhưng browser(chủ yếu IE ko chịu cập nhật…) ko hiểu nó.

🡪 Cần 1 công cụ chuyển đổi mã code ES6,7.. thành mã code mà mọi trình duyệt hiểu đc.



+ @babel/cli: thực hiện việc chuyển đổi với vài dòng code

+ @babel/polyfill: cho Babel khả năng convert code tới cả browser cổ điển cũng hiểu

+ @babel/core: nền tảng để thêm các tính năng khác như @babel/present-env để convert ES5++ về ES5

@babel/preset-react để convert React code về ES5

@babel/preset-typescript để convert typescript về ES5

1. **Bundle là gì?**

Link: <https://viblo.asia/p/su-dung-webpack-trong-angular-2-DzVkpLdgknW>

Bundle là quá trình gom(nén) các tài nguyên khác nhau (source code) vào 1 file duy nhất và sau đó trả về client.

Bundle: css, html, js,… hầu hết các loại file.

1. **Ivy, Bazel trong Angular 8+?**

<https://tedu.com.vn/lap-trinh-angular-2-can-ban/angular-80-%E2%80%93-co-gi-moi-cach-nang-cap-ra-sao-202.html>

Là một cải tiến trong Angular 8.

Hoạt động theo cơ chế render mới.

Ứng dụng giảm kích thước bundle.

Bazel: Công cụ build của Google được phát hành như mã nguồn mở mới được tích hợp trong angular 8(có thể tới 9 mới dung)

1. **Stateless và statefull**

<https://topdev.vn/blog/stateless-la-gi-stateful-la-gi/>

Trạng thái tương tác dữ liệu giữa client và server

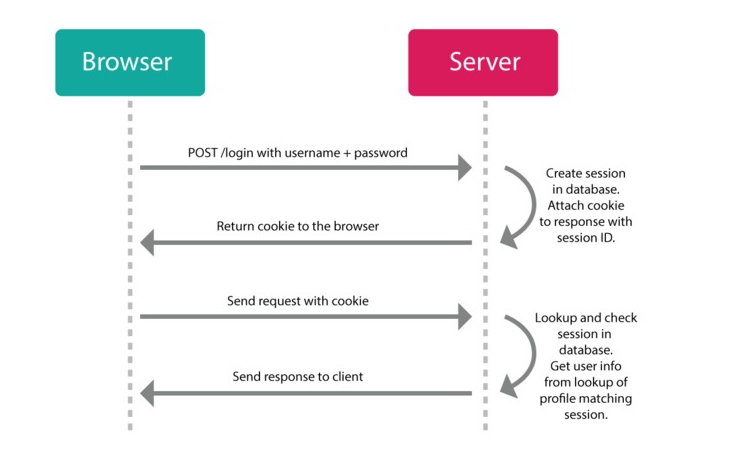
Stateless: ko lưu trữ dữ liệu client trên server. Tức khi client gửi dữ liệu lên server và server trả về kết quả thì quan hệ ‘client-server’ bị cắt đứt.

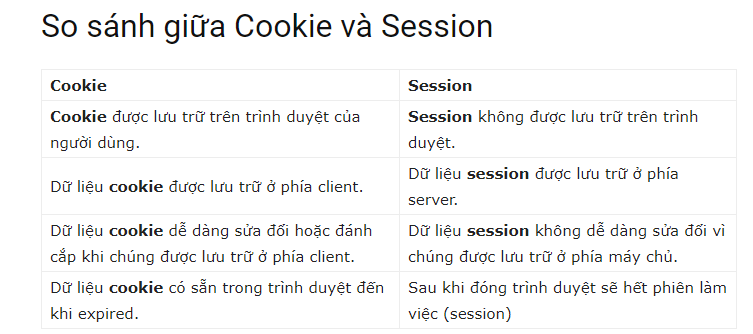
Stateful: Cần server lưu dữ liệu client, tức ràng buộc đc giữ sau mỗi request. Data đc lưu phía client có thể đc làm param cho request tiếp theo.

1. **Duy trì trạng thái đăng nhập? Session and Cookie**

Nơi lưu trữ cookie: C:\Users\Your User Name\AppData\Local\Google\Chrome\User Data\Default."

Phân biệt trình duyệt khác nhau đã login chưa? Session ra đời

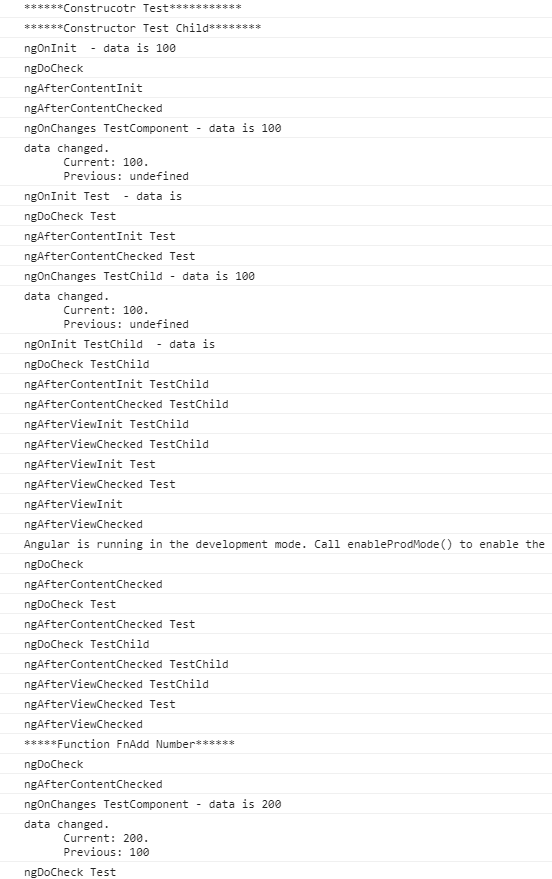
<https://topdev.vn/blog/session-la-gi-cookie-la-gi/> 



1. **LifeCycle trong Angular**

<https://www.youtube.com/watch?v=DZGt8qw2b5g>

<https://www.youtube.com/watch?v=RCLwwWQ8-V0>



1. **Change Detection trong Angular**

<https://viblo.asia/p/tim-hieu-ve-change-detection-trong-angular-djeZ18EjKWz>

Cơ chế theo dõi sự thay đổi, cho phép view luôn đồng bộ với sự thay đổi của model

1. **Phân biệt attribute và property**

<https://javascript.info/dom-attributes-and-properties>

Khi browser load page, Html chuyển thành object DOM.

|  |  |
| --- | --- |
| Attribute | Property |
| - Xác định bởi HTML | - Xác định bởi DOM |
| - Khởi tạo DOM object  - Sau đó done, hết nhiệm vụ |  |
| - Ko thay đổi đc value | - Thay đổi đc value |

1. **Cách code thế nào là đẹp?**

Comment: giải thích tại sao mọi thứ đc làm như vậy, chứ ko phải chỗ này làm gì…

Indention: viết code thụt dòng chuẩn

Chuẩn hóa file readme: giới thiệu dự án, cài đặt môi trường

Naming convention: đặt tên hàm, lớp, biên theo chuẩn.

Tránh magic number: tức số đó là gì? Ko ai hiểu ý nghĩa? 🡺 biến const

Code đơn giản, đọc dễ hiểu

Mỗi hàm chỉ làm 1 việc(tái sử dụng), và mỗi hàm ko quá dài.

Quá nhiều param 🡪 chuyển thành object.

Don’t return null: thay vào đó trả về empty list

Tên biến có ý nghĩa.

1. **Refactor code?**

Viết lại source code một cách khoa học mà vẫn giữ đc tính đúng đắn.

Để giúp dễ đọc, dễ hiểu -> dễ bảo trì, nâng cấp.

Khi nào refactor code: bất cứ khi nào, review code, thêm chức năng vào source cũ, bàn giao code.

1. **SSR và CSR là gì?**

<https://techblog.vn/tim-hieu-ve-render-o-server-side-va-client-side-ssr-cs-csr>

Server side render và client side render (Tùy dự án, quốc gia)

+ SSR: Request -> server xử lí -> html -> return cho browser hiển thị (SSR: thời kì đầu và có nhiều nhược điểm)

Nhược điểm: reload lại từ đầu kể cả thay đổi rất nhỏ, server quá tải do làm quá nhiều việc, tốn tài nguyên server

+ CSR: máy tính cá nhân ngày càng mạnh -> tận dụng nó thay vì server xử lí hết, nhưng vẫn có nhược điểm

Nhược điểm: Lượng dữ liệu lần đầu load về khá chậm, SEO bị ảnh hưởng do nội dung web đc hiển thị trên client -> search engine ko tìm thấy 🡪 Google đã giải quyết

1. **Template Driven From and Reactive Form**

<https://www.pluralsight.com/guides/difference-between-template-driven-and-reactive-forms-angular>

|  |  |
| --- | --- |
| Template Driven Form | Reactive Form |
| - Sử dụng FormModule | - Dựa vào ReactiveFormModule |
| - Không đồng bộ | - Bản chất đồng bộ |
| - Logic hầu hết trong template | - Logic chủ yếu trong Component/TS code |
| - Binding 2 chiều [(NgModel)] |  |

1. Rxjs
2. **Progressive Web Apps**

<https://viblo.asia/p/gioi-thieu-ve-progressive-web-apps-maGK7WPBKj2>

Xu hướng phát triển ứng dụng di động sử dụng công nghệ web.

Hỗ trợ offline, tải ứng dụng nhanh, bảo mật tốt hơn, đẩy notification, full màn hình(ko có thanh url)

Ứng dụng khá nhẹ, Seo tốt, ko tải qua app store…

1. **HTTP and HTTPS**

<https://viettelidc.com.vn/tin-tuc/su-khac-nhau-giua-http-va-https>

HTTP là giao thức mạng trong giao tiếp giữa client-server

HTTPS: giao thức HTTP có sử dụng thêm chứng chỉ SSL giúp mã hóa dữ liệu truyền tải

1. **HTTP Interceptor trong Angular**

<https://techblog.vn/xu-ly-http-interceptor-trong-angular-2>

Giao tiếp với backend sử dụng HTTP Service 🡪 thiết lập thông số header, body, ..hoặc điều khiển các request trước khi gửi đến server, hoặc xử lí các phản hồi từ máy chủ trước khi thông báo thực hiện xong(xử lí lỗi,..).

1. Angular Từ A-> Z
2. **Agile Scrum**

Release theo tuần, đầu tuần nhận task, giao task. Hàng ngày họp, các vấn đề phát sinh? Hướng giải quyết?.

Cuối tuần bàn giao cho khách hàng.

1. Form Group và Form Builder Serivce
2. Get, set function in angular
3. Docker
4. **SQL and NoSql**

|  |  |
| --- | --- |
| SQL | NOSQL |
| - Hệ thống quản lí dữ liệu quan hệ | - Dữ liệu phân tán |
| - Mở rộng chiều dọc (Thêm CPU, RAM,..) | - Mở rộng chiều ngang (Thêm máy chủ) |
| - Query phức tạp | - Không tốt cho query phức tạp |
| - Ko phù hợp lưu trữ dữ liệu phân cấp | - Phù hợp cho lưu trữ dữ liệu phân cấp |
| - Schema cố định/dự đoán đc | - Schema động |
|  |  |

1. Schema và Database

Mô tả quan hệ các đối tượng và thông tin trong cơ sở dữ liệu.

1. **SOAP và REST**

Cách thức truy cập Webserivce

|  |  |
| --- | --- |
| SOAP | REST |
| - NGôn ngữ, nền tảng độc lập(nhiều giao thức HTTP, SMTP) | - Bắt buộc HTTP |
| - Hoạt động tốt trong môi trường doanh nghiệp | - Ko cần dùng XML phức tạp (Json, csv,…) cho request, response. |
| - Tiêu chuẩn hóa | - Nhanh hơn. |
| - Khả năng mở rộng đáng kể |  |
| - Tích hợp xử lí lỗi |  |
| - Tự động hóa với 1 số ngôn ngữ |  |

1. **Design Pattern**

<https://viblo.asia/p/design-pattern-trong-javascript-Az45bYYolxY>

Module Pattern: taoj ra code độc lập(chỉnh sửa và ko ảnh hưởng chỗ khác), tạo ra scpe cho các biến, tái sử dụng.

🡪 giúp code sạch, tách biệt,

🡪 tạo các public function để gọi private function-variable

Revealing Module Pattern: phiên bản cải tiến (map tới object kết quả trả về)

ES6 Module: keyword export, import và luôn là ‘strict mode’

Singleton pattern: object chỉ khởi tạo 1 lần, nếu đã tồn tại t hì trả lại instance đó. (tránh lãng phí bộ nhớ)

🡪 object literal trong js chính là 1 kiểu singleton: khi gọi tới object sẽ nhận đc 1 tham chiếu.

Factory Pattern: tạo object mà ko cần chỉ định rõ class hay constructor nào.

Decorator Pattern: Thêm tính năng cho một đối tượng object và ko làm thay đổi class hay hàm constructor tạo nó.