**Giới thiệu về Angular**

1. Angular là gì?

* Angular là một framework phát triển ứng dụng web front-end mã nguồn mở, được phát triển và duy trì bởi Google. Phiên bản đầu tiên là AngularJS (2010), nhưng kể từ Angular 2+, kiến trúc của framework đã thay đổi đáng kể, trở thành một framework hoàn toàn mới dựa trên TypeScript thay vì JavaScript.

2. Lịch sử phát triển của Angular

* AngularJS ra mắt vào năm 2010 với mô hình MVC (Model-View-Controller), giúp tách biệt phần hiển thị và logic nghiệp vụ.
* Năm 2016, Angular 2 ra đời, đánh dấu sự thay đổi lớn từ AngularJS với TypeScript và cấu trúc module mới.
* Các phiên bản Angular 2+ tiếp tục phát triển, với các bản cập nhật lớn hàng năm, hiện tại đang ở phiên bản Angular 16 (2023).

3. Sự khác biệt giữa AngularJS và Angular (Angular 2+)

* AngularJS sử dụng JavaScript, trong khi Angular 2+ chuyển sang TypeScript với nhiều tính năng mạnh mẽ hơn.
* Angular 2+ cung cấp kiến trúc component-based, giúp tối ưu hóa khả năng tái sử dụng mã và hiệu suất cao hơn.

4. Tại sao nên sử dụng Angular?

* Khả năng tái sử dụng mã: Component-based architecture giúp tách biệt các phần của ứng dụng và dễ dàng tái sử dụng.
* Hiệu suất cao: Angular hỗ trợ Ahead of Time Compilation (AOT) và Lazy Loading, giúp cải thiện tốc độ tải ứng dụng.
* Dễ mở rộng: Cộng đồng lớn, hệ sinh thái phong phú.
* Phát triển nhanh chóng: Angular CLI giúp giảm thời gian tạo và quản lý dự án.

5. So sánh với các framework khác

* Angular vs. React: Angular là một framework toàn diện, trong khi React chỉ là một thư viện UI.
* Angular vs. Vue.js: Vue.js nhẹ hơn, dễ học hơn, nhưng không mạnh mẽ bằng Angular trong việc phát triển ứng dụng lớn.

6. Các công ty đang sử dụng Angular

* Một số công ty lớn sử dụng Angular trong sản phẩm của họ:
  + Google (ứng dụng nội bộ).
  + Microsoft (Office 365).
  + IBM (IBM Cloud).

II. Kiến trúc của Angular

1. Kiến trúc tổng quát

* Angular sử dụng kiến trúc Component-based và Module-based, chia nhỏ ứng dụng thành các component và module dễ quản lý.
* Model-View-Controller (MVC) vẫn được duy trì trong cách Angular tổ chức ứng dụng.

2. Components (Thành phần)

* Định nghĩa: Component là thành phần cơ bản nhất trong Angular, bao gồm phần hiển thị (template) và logic (class).
* Tạo Component: ng generate component <component-name>.
* Mối quan hệ Parent-Child: Component có thể được lồng vào nhau, parent component có thể truyền dữ liệu cho child component qua input binding.

3. Modules (Mô-đun)

* Modules: Mỗi ứng dụng Angular được chia thành nhiều module (app module, feature module).
* Vai trò: Tổ chức và quản lý các thành phần của ứng dụng.
* Cấu trúc và sử dụng: Mỗi module có một file app.module.ts để khai báo các component, directive, và service.

4. Services và Dependency Injection (DI)

* Service: Chứa logic nghiệp vụ, được dùng chung bởi nhiều component.
* Dependency Injection: Angular hỗ trợ DI để giảm thiểu sự phụ thuộc giữa các phần của ứng dụng.

5. Data Binding

* One-way Data Binding: Dữ liệu đi từ model đến view hoặc từ view đến model.
* Two-way Data Binding: Cập nhật cả hai chiều (sử dụng [(ngModel)]).
* Event Binding và Property Binding: Kết nối giữa các sự kiện và thuộc tính DOM.

6. Routing

* Routing trong Angular cho phép điều hướng giữa các view.
* Cấu hình routing: Khai báo các route trong file app-routing.module.ts.

III. Các tính năng quan trọng của Angular

1. Template

* Angular sử dụng HTML template để tạo giao diện.
* Tích hợp các biểu thức Angular để hiển thị và cập nhật dữ liệu động.

2. Directives

* Structural Directives: Thay đổi cấu trúc DOM (\*ngIf, \*ngFor).
* Attribute Directives: Thay đổi hành vi hoặc thuộc tính của các phần tử DOM (ngClass, ngStyle).

3. Pipes

* Pipes: Chuyển đổi dữ liệu trực tiếp trong template (uppercase, date, currency).
* Custom Pipes: Tạo pipes tùy chỉnh.

4. Forms trong Angular

* Template-driven forms: Sử dụng template HTML để quản lý form.
* Reactive forms: Kiểm soát form qua logic trong code.
* Validation: Angular hỗ trợ xác thực dữ liệu form theo thời gian thực.

5. HTTP Client

* HttpClient giúp gọi API RESTful.
* Các thao tác CRUD: GET, POST, PUT, DELETE.
* Xử lý lỗi: Angular cung cấp cơ chế xử lý lỗi API với catchError.

IV. Angular CLI và Best Practices

1. Angular CLI

* CLI giúp tạo và quản lý dự án dễ dàng với các lệnh như ng new, ng serve, ng generate component.
* Lệnh phổ biến:
  + ng new <project-name>: Tạo dự án mới.
  + ng serve: Chạy dự án ở chế độ phát triển.

2. Best Practices

* Tổ chức thư mục: Phân chia hợp lý giữa components, services, và modules.
* Performance Optimization: Sử dụng Lazy Loading, AOT để tối ưu hóa hiệu suất.
* Tái sử dụng mã: Sử dụng các module và service để giảm thiểu mã lặp lại.

**Hướng dẫn cài đặt và demo**

1. Cài đặt môi trường phát triển:
2. Vs code
3. Node js : <https://nodejs.org/en>
4. Angular CLI: <https://angular.dev/tools/cli/setup-local>

* Chạy lệnh npm install -g @angular/cli để cài đặt Angular CLI
* Chạy lệnh Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned ở Powershell với quyền quản trị để đảm bảo chạy được các lệnh run.

1. Tạo dự án mới: ng new my\_app --no-standalone

A screenshot of a computer

Description automatically generated

Chọn thư mục muốn lưu dự án

Gõ như hình ở đường dẫn thư mục để mở PowerShell

Sau khi mở PowerShell chạy lệnh ng new my\_app --no-standalone (tạo mới 1 dự án).

1. Vi du de tao demo goi api voi Angular:

Lệnh tạo component: ng generate component users

Điều này tạo ra bốn file cho component Users:

* users.component.ts: Chứa logic của component.
* users.component.html: Chứa template HTML cho giao diện.
* users.component.css: Chứa các kiểu CSS (nếu cần).
* users.component.spec.ts: Dùng để viết test cho component (không cần quan tâm cho mục đích này).

Lệnh tạo service: ng generate service api

Điều này tạo ra file api.service.ts

File api.service.ts sẽ như sau:

A computer screen with many colorful text

Description automatically generated

Giải thích từng phần

* @Injectable({ providedIn: 'root' }): Đánh dấu service là injectable, cho phép nó được sử dụng ở bất kỳ đâu trong ứng dụng mà không cần phải thêm vào providers của module.
* HttpClient: Được tiêm vào constructor để cho phép thực hiện các cuộc gọi HTTP.
* apiUrl: Địa chỉ API mà bạn sẽ gọi để lấy dữ liệu người dùng.
* getUsers(): Phương thức này sử dụng HttpClient để thực hiện yêu cầu GET đến apiUrl và trả về một Observable, cho phép bạn xử lý dữ liệu bất đồng bộ.

File users.component.ts:

A screen shot of a computer program

Description automatically generated

**Giải thích từng phần**

* **OnInit**: Một lifecycle hook mà Angular gọi sau khi component đã được khởi tạo. Đây là nơi tốt để gọi API.
* **users: any[] = []**: Khai báo biến users để lưu trữ dữ liệu người dùng.
* **constructor(private apiService: ApiService)**: Tiêm ApiService vào constructor để sử dụng trong component.
* **ngOnInit()**: Gọi phương thức loadUsers() khi component được khởi tạo.
* **loadUsers()**: Gọi API để lấy dữ liệu người dùng. Nếu thành công, gán dữ liệu cho biến users. Nếu có lỗi, in ra console.

File users.component.html:

A screen shot of a computer program

Description automatically generated

Trong file app.component.html, bạn sẽ có:

<h1>Demo Angular Gọi API</h1>

<app-users></app-users>

Chạy dự án bằng lệnh “ng serve”