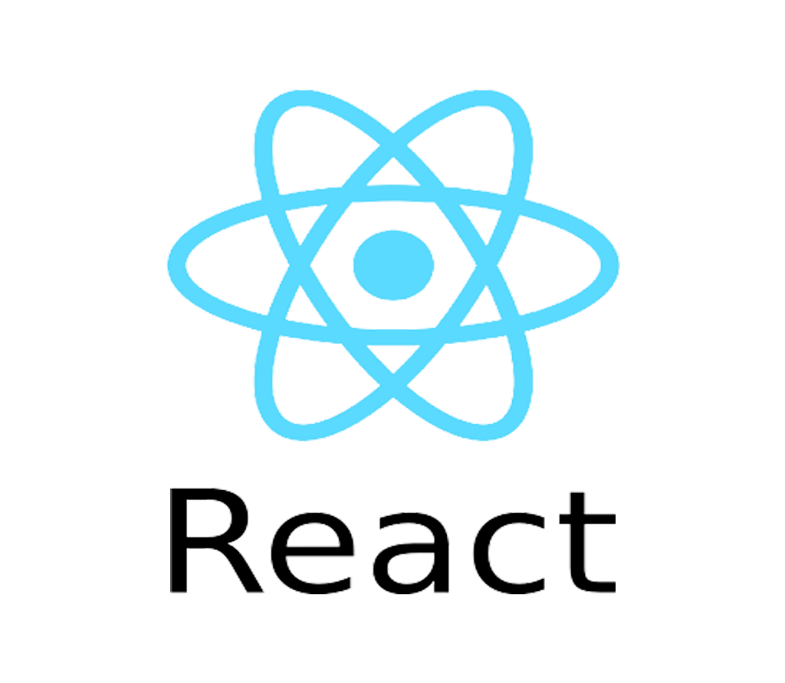
**Tìm hiểu về React**

**React**



React (ReactJS) là một thư viện [JavaScript](https://200lab.io/blog/javascript-la-gi/) mã nguồn mở, được dùng để xây dựng giao diện người dùng (frontend) cho web. React chỉ tập trung vào phần hiển thị giao diện (view), chứ không can thiệp vào cách sắp xếp logic nghiệp vụ hoặc cấu trúc ứng dụng.

**Những đặc điểm nổi bật của React**

1. **Linh hoạt trong thiết kế kiến trúc**

React tập trung vào việc hiển thị giao diện người dùng và cho phép lập trình viên tự do quyết định cách sắp xếp logic nghiệp vụ. Chính sự linh hoạt này làm cho React trở nên phổ biến, vì nó phù hợp với nhiều loại dự án và phong cách phát triển khác nhau.

Khác với các framework có kiến trúc cố định như Angular, React không ép buộc người dùng vào một mô hình cụ thể, khiến nó linh hoạt cho nhiều dự án khác nhau. Mặc dù điều này trong React đem lại lợi ích cho lập trình viên có kinh nghiệm, nhưng lại gây khó khăn cho người mới bắt đầu vì thiếu sự hướng dẫn cụ thể.

1. **Kiến trúc Component đơn giản và nhẹ**

React được xây dựng dựa trên kiến trúc component, nơi mỗi thành phần có thể được tái sử dụng, giúp ứng dụng dễ mở rộng và duy trì. Các component trong React rất nhẹ và có thể chỉ là các hàm đơn giản trả về JSX.

Điều này giúp các lập trình viên dễ dàng phát triển các ứng dụng lớn hơn bằng cách chia nhỏ thành các thành phần độc lập, có thể tái sử dụng mà không cần phải lo về việc tích hợp phức tạp.

1. **Cộng đồng hỗ trợ lớn**

Một trong những lý do chính khiến React trở nên phổ biến là nhờ vào số lượng người dùng lớn và sự hỗ trợ từ cộng đồng cũng như các công cụ học tập. Đây là yếu tố quan trọng giúp các lập trình viên, đặc biệt là người mới bắt đầu, dễ dàng học và sử dụng React một cách hiệu quả.

Các diễn đàn lớn như Stack Overflow, Reddit, Dev.to, và Github chứa đầy các câu hỏi, câu trả lời và bài viết liên quan đến React, giúp người học dễ dàng tìm thấy sự trợ giúp khi gặp khó khăn.

React được duy trì và phát triển bởi Facebook (Meta), một trong những công ty công nghệ lớn nhất thế giới. Điều này đảm bảo rằng React sẽ nhận được sự hỗ trợ liên tục và các bản cập nhật mới, giúp lập trình viên yên tâm rằng React sẽ tiếp tục phát triển và phù hợp với các dự án dài hạn.

**Các thành phần quan trọng trong ReactJS**

JSX (JavaScript XML) là một cú pháp mở rộng cho phép lập trình viên viết mã giống như [HTML](https://200lab.io/blog/html-la-gi/) trong JavaScript. Trong các ngôn ngữ khác, lập trình viên thường phải viết code HTML và JavaScript riêng rẽ. Tuy nhiên, với JSX, React cho phép lập trình viên kết hợp cả hai trong cùng một mã nguồn, giúp quản lý dễ dàng hơn, đặc biệt là trong các ứng dụng phức tạp.

Virtual DOM (Document Object Model ảo) là một bản sao nhẹ hơn của DOM thật. DOM thật là cấu trúc cây chứa tất cả các thành phần HTML trong trang web. Khi người dùng tương tác với ứng dụng (ví dụ: nhập văn bản, nhấn nút), ứng dụng sẽ thay đổi nội dung và DOM thật phải được cập nhật.

Tuy nhiên, việc cập nhật trực tiếp DOM thật có thể gây ra tình trạng chậm chạp và kém hiệu quả, đặc biệt trong các ứng dụng lớn với nhiều phần tử. Để giải quyết vấn đề này, React sử dụng Virtual DOM, với mục đích chỉ những phần có sự thay đổi mới được cập nhật lên DOM thật, giúp tiết kiệm thời gian và tài nguyên.

**Những lợi ích tuyệt vời mà ReactJS mang lại cho lập trình viên**

* Hiệu suất cao: ReactJS sử dụng Virtual DOM để tối ưu hóa hiệu suất của ứng dụng. Virtual DOM cho phép ReactJS cập nhật các thay đổi trên trang web một cách nhanh chóng và hiệu quả hơn so với cách truyền thống, giúp tăng tốc độ và hiệu suất của ứng dụng.
* Tái sử dụng: ReactJS cho phép tái sử dụng các thành phần UI, giúp giảm thiểu thời gian và chi phí phát triển. Các thành phần UI có thể được sử dụng lại trong nhiều phần khác nhau của ứng dụng, giúp tăng tính linh hoạt và khả năng mở rộng của ứng dụng.
* Dễ dàng quản lý trạng thái: ReactJS giúp quản lý trạng thái của ứng dụng một cách dễ dàng. Sử dụng State và Props, ReactJS cho phép các nhà phát triển quản lý trạng thái của các thành phần UI một cách chính xác và dễ dàng.
* Hỗ trợ tốt cho SEO: ReactJS cho phép các nhà phát triển xây dựng ứng dụng web với khả năng tương thích tốt với SEO. Với sự hỗ trợ của các thư viện như React Helmet, ReactJS cho phép các nhà phát triển tùy chỉnh và quản lý các phần tử meta và title cho từng trang web.
* Hỗ trợ đa nền tảng: ReactJS không chỉ được sử dụng để phát triển các ứng dụng web, mà còn được sử dụng để phát triển các ứng dụng di động với React Native. Sử dụng React Native, các nhà phát triển có thể xây dựng ứng dụng di động cho cả iOS và Android sử dụng cùng một mã nguồn.

**Các tính năng nổi bật của ReactJS**

* Components: ReactJS cho phép phát triển ứng dụng web theo mô hình component. Các component là các phần tử UI độc lập có thể được tái sử dụng trong nhiều phần khác nhau của ứng dụng.
* Virtual DOM: ReactJS sử dụng Virtual DOM để tối ưu hóa hiệu suất của ứng dụng. Virtual DOM là một bản sao của DOM được lưu trữ trong bộ nhớ và được cập nhật một cách nhanh chóng khi có thay đổi, giúp tăng tốc độ và hiệu suất của ứng dụng.
* JSX: JSX là một ngôn ngữ lập trình phân biệt được sử dụng trong ReactJS để mô tả các thành phần UI. JSX kết hợp HTML và JavaScript, giúp cho việc viết mã dễ hiểu và dễ bảo trì hơn.
* State và Props: ReactJS cho phép quản lý trạng thái của các thành phần UI thông qua State và Props. State là trạng thái của một thành phần được quản lý bởi nó chính, trong khi Props là các giá trị được truyền vào từ bên ngoài để tùy chỉnh hoặc điều khiển hành vi của một thành phần.
* Hỗ trợ tốt cho SEO: ReactJS hỗ trợ tốt cho việc tối ưu hóa SEO. Với các thư viện như React Helmet, các nhà phát triển có thể quản lý các phần tử meta và title cho từng trang web, giúp tăng khả năng tìm kiếm và tăng cường trải nghiệm người dùng.
* Hỗ trợ đa nền tảng: ReactJS không chỉ được sử dụng để phát triển ứng dụng web, mà còn được sử dụng để phát triển ứng dụng di động với React Native. Sử dụng React Native, các nhà phát triển có thể xây dựng ứng dụng di động cho cả iOS và Android sử dụng cùng một mã nguồn.
* Redux: Redux là một thư viện quản lý trạng thái cho các ứng dụng ReactJS. Nó giúp quản lý trạng thái của ứng dụng một cách chính xác và dễ dàng, đồng thời giúp tăng tính linh hoạt và khả năng mở rộng của ứng dụng.

**Cách sử dụng ReactJS trong phát triển web**

**Bước 1 Cài đặt Node.js và npm**

ReactJS được xây dựng trên nền tảng Node.js, do đó lập trình viên cần cài đặt Node.js và npm để phát triển ứng dụng ReactJS.

**Bước 2 Tạo một ứng dụng React**

lập trình viên có thể tạo một ứng dụng React bằng cách sử dụng lệnh "create-react-app" trong Command Prompt hoặc Terminal.

**Bước 3 Tạo các component**

Tạo các component để xây dựng giao diện người dùng cho ứng dụng. Lập trình viên có thể tạo component bằng cách sử dụng class hoặc hàm.

**Bước 4 Xây dựng giao diện người dùng**

Sử dụng JSX để xây dựng giao diện người dùng cho ứng dụng. JSX là một ngôn ngữ phân biệt được sử dụng trong ReactJS để mô tả các thành phần UI.

**Bước 5 Quản lý trạng thái**

Sử dụng State và Props để quản lý trạng thái của các thành phần UI. State là trạng thái của một thành phần được quản lý bởi nó chính, trong khi Props là các giá trị được truyền vào từ bên ngoài để tùy chỉnh hoặc điều khiển hành vi của một thành phần.

**Bước 6 Kết nối với API**

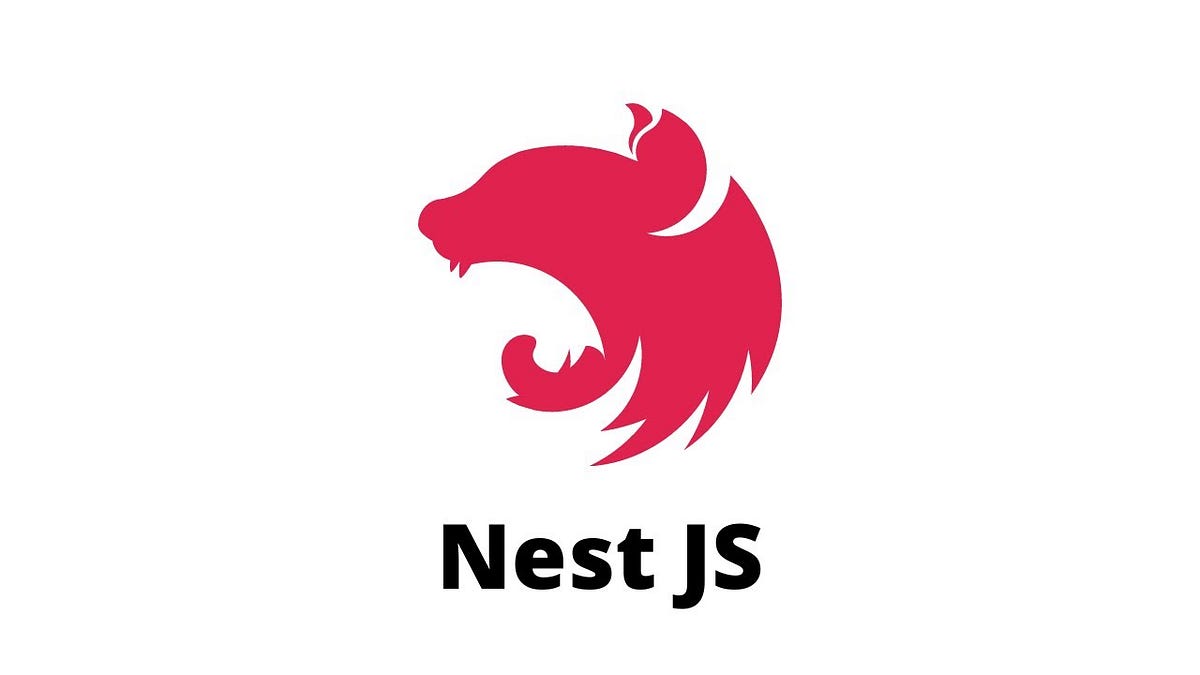
Sử dụng thư viện như Axios để kết nối với API và lấy dữ liệu từ server.

**Bước 7 Build và triển khai ứng dụng**

Sử dụng lệnh "npm run build" để build ứng dụng và triển khai nó trên môi trường sản phẩm.

**Tìm hiểu về NestJS**

**NestJS**



NestJS là một framework mã nguồn mở để phát triển ứng dụng server-side (backend applications) bằng ngôn ngữ TypeScript hoặc JavaScript. Nó được xây dựng trên cơ sở của Node.js và sử dụng các khái niệm từ TypeScript để tạo ra một môi trường phát triển hiện đại và mạnh mẽ cho việc xây dựng các ứng dụng web và API.

Mục tiêu chính của NestJS là cung cấp một cấu trúc ứng dụng rõ ràng và dễ quản lý, giúp tăng tính bảo trì và sự tổ chức trong mã nguồn. Để đạt được điều này, NestJS triển khai mô hình kiến trúc lõi (core architecture) dựa trên các nguyên tắc của Angular, đặc biệt là sử dụng Dependency Injection (DI) và Modules (Các module).

**Cấu trúc của NestJS**

Cấu trúc của NestJS được xây dựng dựa trên mô hình kiến trúc lõi (core architecture) giúp tạo ra một ứng dụng server-side (backend application) rõ ràng, dễ quản lý và dễ mở rộng. Cấu trúc NestJS thường được tổ chức thành các phần chính sau:

Module (Các module): Module là một phần cơ bản trong cấu trúc NestJS. Mỗi ứng dụng NestJS bao gồm ít nhất một module gốc (root module) và có thể có nhiều module con. Module là nơi tổ chức các thành phần của ứng dụng như Controllers, Providers và các thành phần khác. Mỗi module đại diện cho một phần chức năng cụ thể của ứng dụng.

Controller (Bộ điều khiển): Controllers là thành phần chịu trách nhiệm xử lý các yêu cầu HTTP từ phía client và trả về kết quả tương ứng. Controllers là nơi xử lý các request và trả về các response. Các phương thức của controller được chú thích (decorated) bằng các decorator như `@Get()`, `@Post()`, `@Put()`, v.v., để chỉ định các route và phương thức HTTP tương ứng.

Provider (Các nhà cung cấp): Providers là thành phần chịu trách nhiệm cung cấp các dịch vụ cho ứng dụng. Đây có thể là các service, repository, logger, v.v. Providers sử dụng dependency injection để chèn vào các thành phần khác và có thể được sử dụng bởi các controllers hoặc các providers khác.

Middleware (Trung gian): Middleware là các hàm xử lý mà NestJS sử dụng để xử lý các yêu cầu HTTP trước khi chúng đến các route xử lý chính. Middleware có thể được sử dụng để thực hiện các thao tác chung như xác thực, ghi log, xử lý lỗi, v.v.

Filter (Bộ lọc): Filters được sử dụng để xử lý các exception (ngoại lệ) xảy ra trong ứng dụng. Filters cho phép lập trình viên xử lý và thay đổi response trước khi gửi về client khi có exception xảy ra.

Guard (Bảo vệ): Guards được sử dụng để kiểm tra xem một yêu cầu có thể được xử lý hoặc không. Guards cho phép lập trình viên thực hiện các kiểm tra xác thực hoặc kiểm tra quyền trước khi xử lý một yêu cầu.

Interceptor (Bộ chặn): Interceptors là các hàm xử lý mà NestJS sử dụng để chặn và thay đổi response trước khi nó được gửi về client. Interceptors có thể được sử dụng để thực hiện các thao tác chung trên response trước khi nó đi ra ngoài.

Exception (Ngoại lệ): Exception handling (xử lý ngoại lệ) là một phần quan trọng của cấu trúc NestJS. Exception handling cho phép xử lý các exception xảy ra trong ứng dụng và trả về các thông báo lỗi thích hợp cho client.

**Hướng dẫn cài đặt NestJS**

Để cài đặt NestJS, lập trình viên cần đảm bảo đã cài đặt Node.js và npm (Node Package Manager) trước tiên. Sau đó, có thể sử dụng npm để cài đặt NestJS CLI (Command Line Interface) và tạo một dự án NestJS mới. Dưới đây là hướng dẫn cài đặt NestJS trên hệ điều hành Windows, macOS và Linux:

**Bước 1: Cài đặt Node.js và npm**

Trước tiên, hãy truy cập trang chủ của Node.js (https://nodejs.org/) để tải về phiên bản mới nhất và cài đặt Node.js và npm theo hướng dẫn trên trang web.

**Bước 2: Cài đặt NestJS CLI**

Sau khi cài đặt Node.js và npm thành công, mở cửa sổ dòng lệnh hoặc terminal trên máy tính và chạy lệnh sau để cài đặt NestJS CLI một cách toàn cầu (global):

npm install -g @nestjs/cli

**Bước 3: Tạo một dự án NestJS mới**

Sau khi cài đặt NestJS CLI, có thể sử dụng nó để tạo một dự án NestJS mới. Để tạo một dự án mới, hãy chạy lệnh sau trong cửa sổ dòng lệnh hoặc terminal:

nest new project-name

Trong đó, `project-name` là tên của dự án muốn tạo. NestJS CLI sẽ tạo ra một cấu trúc dự án NestJS cơ bản với các tập tin và thư mục cần thiết.

**Bước 4: Chạy ứng dụng NestJS**

Sau khi dự án NestJS đã được tạo thành công, có thể di chuyển vào thư mục dự án bằng cách chạy lệnh:

cd project-name

Sau đó, có thể chạy ứng dụng NestJS bằng lệnh sau:

npm run start

Hoặc nếu muốn tự động khởi động lại ứng dụng khi có thay đổi trong mã nguồn, có thể sử dụng lệnh:

npm run start:dev

Sau khi ứng dụng đã chạy thành công người dùng có thể truy cập vào <http://localhost:3000> (hoặc cổng khác nếu đã thay đổi cài đặt) để xem ứng dụng NestJS