## Cơ bản về JavaScript

## ****JavaScript là gì?****

**JavaScript là một ngôn ngữ lập trình website, được tích hợp và nhúng trong HTML giúp website sống động hơn**. **JavaScript** cho phép kiểm soát các hành vi của trang web tốt hơn so với khi chỉ sử dụng mỗi HTML. Vậy ứng dụng thực tiễn của **JavaScript là gì**? Các slideshow, pop-up quảng cáo và tính năng autocomplete của Google là những ví dụ dễ thấy nhất cho bạn, chúng đều được viết bằng **JavaScript.**

**JavaScript** là ngôn ngữ lập trình được hỗ trợ hầu như trên tất cả các trình duyệt như Firefox, Chrome, Safari, … thậm chí các trình duyệt trên thiết bị di động.

## Lợi ích khi dùng JavaScript

Là một ngôn ngữ lập trình phổ biến sử dụng trên 92% nền tảng website hiện nay, **JavaScript** đã thể hiện vai trò quan trọng với lĩnh vực này. Tầm quan trọng của nó thể hiện qua các thao tác, công dụng như:

* ***Triển khai tập lệnh phía máy khách:*** Nhờ **Javascript**, các lập trình viên có thể dễ dàng viết tập lệnh phía máy khách, tích hợp các tập lệnh một cách liền mạch vào HTML, cho phép website tương tác, trả lời người dùng ngay lập tức và tạo ra giao diện hiển thị phong phú hơn.
* ***Viết mã phía máy chủ:*** Các lập trình viên có thể viết mã phía máy chủ bằng **JavaScript**.
* ***Đơn giản hóa phát triển ứng dụng web phức tạp:*** Javascript cho phép các nhà phát triển đơn giản hóa thành phần của ứng dụng, qua đó đơn giản hóa việc phát triển các ứng dụng web phức tạp.
* ***Thiết kế web responsive:*** **JavaScript** cho phép **thiết kế web responsive** – tối ưu trên cả máy tính và thiết bị di động chỉ với một bộ mã.
* ***Nhiều bộ chuyển đổi:*** Mặc dù thiếu một số tính năng phức tạp được cung cấp bởi các ngôn ngữ lập trình hiện đại như Java và C#, **JavaScript**vẫn có thể dễ dàng mở rộng bằng cách sử dụng các bộ chuyển đổi như CoffeeScript, TypeScript, DukeScript và Vaadin.

## ****Các thư viện Javascript phổ biến****

Hiện nay có rất nhiều thư viện và framework được viết từ Javascript như:

* **jQuery**: Chuyên về hiệu ứng.
* **NodeJS**: dùng để lập trình cho server.
* **AngularJS**: Chuyên xây dựng ứng dụng trang cá nhân.
* **ReactJS**: Chuyên viết ứng dụng mobile.
* Một số thư viện khác như **ExtJS**, **Sencha Touch**,….

## ****Ưu điểm của JavaScript****

JavaScript có rất nhiều ưu điểm vượt trội hơn so với các đối thủ khác, chẳng hạn như:

* **JavaScript** là ngôn ngữ lập trình **dễ học**.
* **Lỗi của JavaScript dễ phát hiện** hơn và vì vậy dễ sửa hơn.
* **JavaScript hoạt động trên nhiều trình duyệt**, nền tảng.
* **JavaScript giúp website tương tác tốt hơn** với khách truy cập.
* **JavaScript nhanh hơn và nhẹ hơn** các ngôn ngữ lập trình khác.

Không ngôn ngữ lập trình nào là hoàn hảo và không có khuyết điểm. Bởi **JavaScript** là ngôn ngữ lập trình phổ biến, được sử dụng rộng rãi trên nền tảng website, vì vậy nó cũng là đối tượng tiềm năng của một lượng lớn hacker, scammer – những kẻ chuyên tìm kiếm lỗ hổng và các lỗi bảo mật và thâm nhập, đánh cắp dữ liệu nội bộ.

## Tìm hiểu về NodeJS

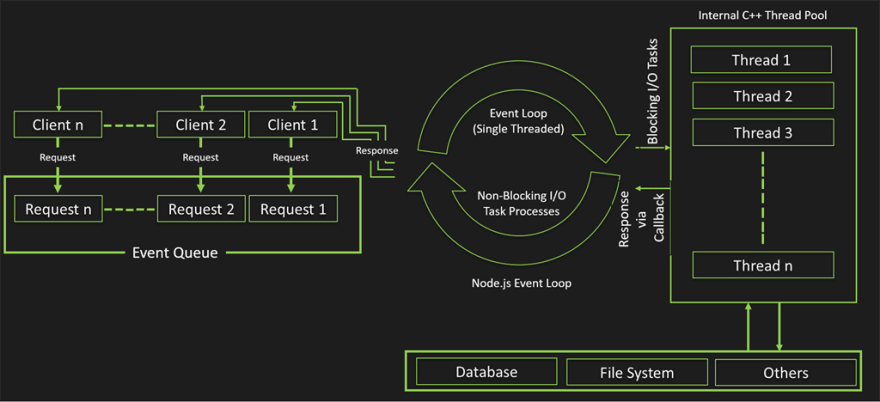
## Nodejs là gì?

* NodeJS là một nền tảng (Platform) được xây dựng trên nền tảng Javascript V8 Engine. Được xây dựng để phát triển những ứng dụng server side.
* Phần core sử dụng Javascript và C++ cho phép xử lý với hiệu năng cao.
* Phù hợp với các ứng dụng xử lý nhanh, real time hoặc những ứng dụng cần thay đổi công nghệ nhanh.
* Chạy sigle thread nhưng có cơ chế non blocking giúp chạy bất đồng bộ, tăng khả năng xử lý.
* Thường phát triển theo hướng module hóa thành các phần nhỏ của ứng dụng và có thể thêm module có sẵn bên ngoài thông qua NPM, yarn....

## Đặc điểm cơ bản của Nodejs:

* **Không đồng bộ:** Đặc điểm đầu tiên của Nodejs là tính bất đồng bộ. Node.js không cần đợi API trả dữ liệu về, vậy nên mọi APIs nằm trong thư viện Node.js đều không được đồng bộ, hiểu đơn giản là chúng không hề blocking (khóa). Server có cơ chế riêng để gửi thông báo và nhận phản hồi về các hoạt động của Node.js và API đã gọi.
* **Tốc độ nhanh:** Với phần core phía dưới lập trình gần như toàn bộ bằng ngôn ngữ C++, kết hợp với V8 Javascript Engine mà Google Chrome cung cấp, tốc độ vận hành, thực hiện code của thư viện Node.js diễn ra rất nhanh.
* **Đơn giản - Hiệu quả:** Tiến trình vận hành của Node.js đơn giản song lại mang đến hiệu năng cao nhờ ứng dụng mô hình single thread và các sự kiện lặp. Một loạt cơ chế sự kiện cho phép server trả về phản hồi bằng cách không block, đồng thời tăng hiệu quả sử dụng. Các luồng đơn cung cấp dịch vụ cho nhiều request hơn hẳn Server truyền thống.
* **Không đệm**: Nền tảng Node.js không có vùng đệm, tức không cung cấp khả năng lưu trữ dữ liệu buffer.
* **Chạy trên nhiều thiết bị:** Nodejs vận hành trên đa nền tảng của Server, bởi vậy nó hoàn toàn có thể chạy trên nhiều thiết bị.
* **Có giấy phép:**Đây là nền tảng đã được cấp giấy phép, phát hành dựa trên MIT License.

## Kiến trúc



# **Những khái niệm cần biết trong NodeJS**

## Npm và package

NPM viết tắt của **N**ode **P**ackage **M**anager là một công cụ tạo và quản lý các thư viện lập trình Javascript cho NodeJS. Trong cộng đồng Javascript, các lập trình viên chia sẻ hàng trăm nghìn các thư viện với các đoạn code đã thực hiện sẵn một chức năng nào đó. Nó giúp cho các dự án mới tránh phải viết lại các thành phần cơ bản, các thư viện lập trình hay thậm chí cả các framework.

## ****Cài đặt và sử dụng Npm****

Npm được tích hợp sẵn có trong NodeJS, để kiểm tra xem trên hệ thống của bạn đã được cài npm chưa chúng ta sử dụng lệnh **npm -v**, nếu một phiên bản hiện ra thì hệ thống của bạn đã được cài đặt npm.

Nếu bạn tạo mới 1 project và muốn sử dụng npm, tốt hơn hết bạn nên bắt đầu với câu lệnh: **npm init**

Câu lệnh trên đơn giản là sẽ tạo ra 1 file có tên là **package.json** – thành phần này được gọi là **Local Package Database**, lưu trữ thông tin (tên package, phiên bản, các dependencies) mà project của bạn sử dụng.

Sau khi chạy câu lệnh này, Npm sẽ hỏi chúng ta một vài câu hỏi về project của bạn:

* Đầu tiên là về package name: Đây là tên dự án của bạn
* Version: Đây là phiên bản dự án của bạn, mặc định sẽ là 1.0.0
* Description: Mô tả về dự án của bạn.
* Entry point: Entry point sẽ quy định root của Node, file này sẽ chứa các câu lệnh về server, mặc định sẽ là index.js. Chú ý là khi bạn chạy server, Node sẽ đọc dữ liệu từ file này để khởi tạo. Tên này có thể thay đổi được, nhưng hãy chắc rằng file đó tồn tại ở root của dự án.
* Test command: Đây là dòng lệnh mà sẽ chạy khi bạn gọi **npm test** Mặc định để trống
* Git repository: Như tên gọi, nó là git repository ứng với dự án của bạn. Mặc định để trống.
* Keywords: Các từ khóa tương ứng với dự án của bạn. Mặc định để trống
* Author: Tên tác giả của dự án. Mặc định để trống
* License: Giấy phép. Mặc định là ISC.

### **Các loại package**

* **Dependecies:** là những packagebắt buộc phải có trong quá trình chạy sản phẩm, kiểu như một thư viện cung cấp các hàm mà code của bạn cần. Khi cài đặt dependencies, Npm sẽ tự động cài đặt tất cả các dependencies cần thiết.



* **Development dependencies:**là những package bắt buộc khi phát triển cũng như phát hành sản phẩm. Kiểu như các trình biên dịch giúp biên dịch đoạn code của bạn về javascript, rồi các framework phục vụ cho việc kiểm thử,…



### **Cài đặt các package như thế nào nhé**

* **Dependencies package:** Cú pháp như sau:

npm install [tên package]

* **Development dependencies package:** Cú pháp như sau**:**

npm install [tên package] --save-dev

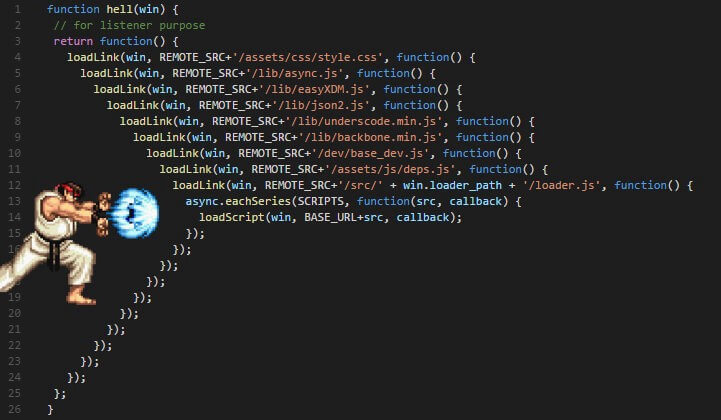
## Những khái niệm cơ bản khi viết ứng dụng NodeJS backend

## Async callback

Bất đồng bộ callback xuất hiện rất nhiều ở Javascript



* Ưu điểm: Dễ hiểu (vì chỉ là một callback function đơn giản)
* Nhược điểm: Khó khăn trong việc xử lý lỗi, code bị lồng vào nhau quá nhiều (callback hell)



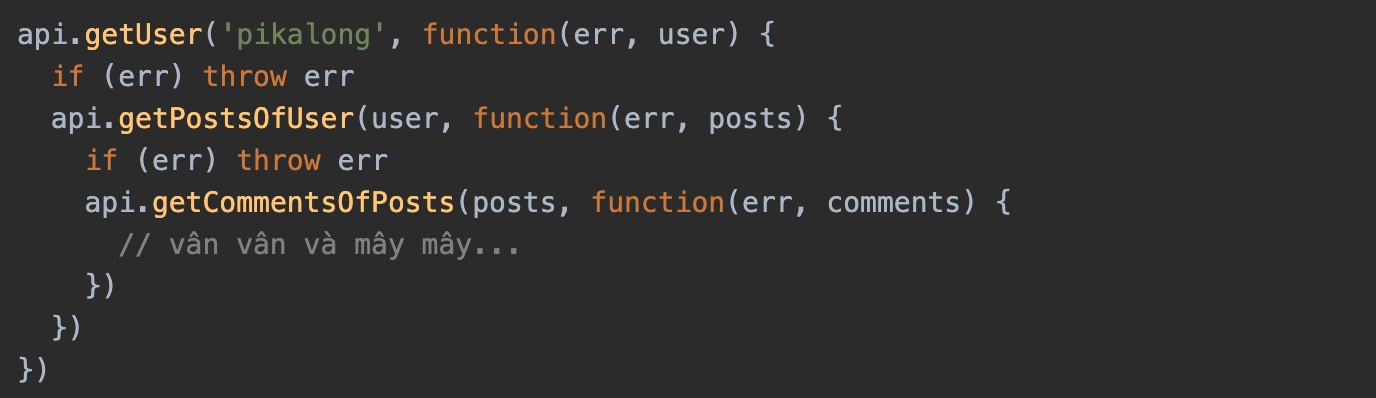
*Hiện tượng Callback Hell (tức là có quá nhiều callback lồng nhau)*

## Promise ES6

Promise là một cơ chế trong JavaScript giúp bạn thực thi các tác vụ bất đồng bộ mà không rơi vào callback hell, là tình trạng các hàm callback lồng vào nhau ở quá nhiều tầng.

Một Promise có 3 trạng thái sau:

* **pending**: đang chờ kết quả
* **fulfilled**: đã có kết quả
* **rejected**: lỗi xảy ra



Một ví dụ về callback hell



*Ví dụ trên khi được viết lại bằng Promise*

## Async/Await ES7

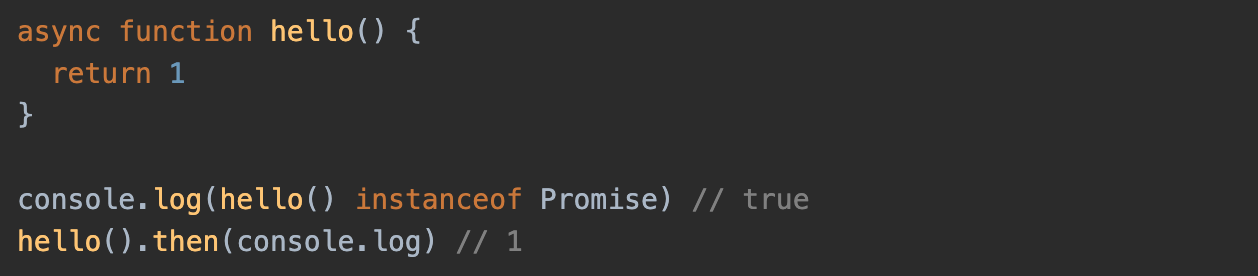
Nếu ở phiên bản ES6 thì promise có thể giải quyết được phần nào callback hell, nhưng vẫn chưa được triệt để. Đến ES7 chúng ta có thể xử lý bất đồng bộ một cách dễ nhìn và dễ đọc hơn với async/await.

Async/Await không thay thế promise mà nó kết hợp với promise để cho ra cú pháp dễ nhìn hơn. Vì thế để học async/await yêu cầu bạn phải hiểu được promise.

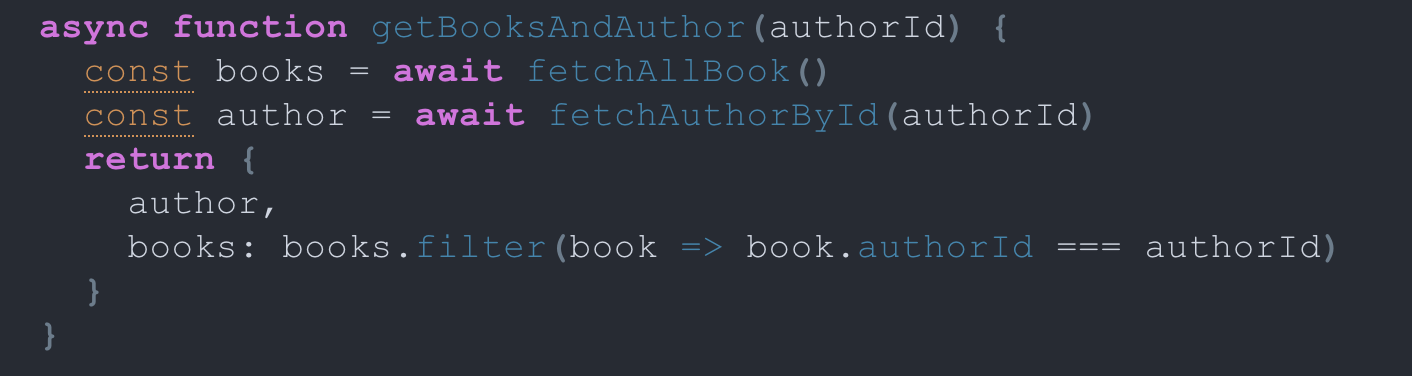
Để sử dụng, bạn phải khai báo hàm với từ khóa **async**. Khi đó bên trong hàm bạn có thể dùng **await**.



Cần lưu ý là kết quả trả về của async function luôn là một Promise

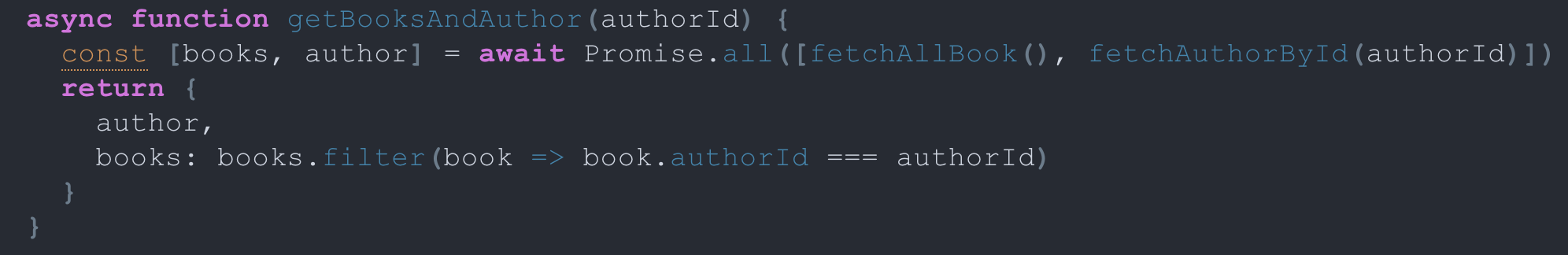


**Gọi tuần tự với async/await**



**Tối ưu performance với Promise.all()**

Nhìn vào đoạn code bên trên, thay vì đợi lấy fetchAllBook() xong rồi đếnfetchAuthorById() thì ta có thể cho 2 function này chạy ‘cùng lúc’



## Giao thức HTTP

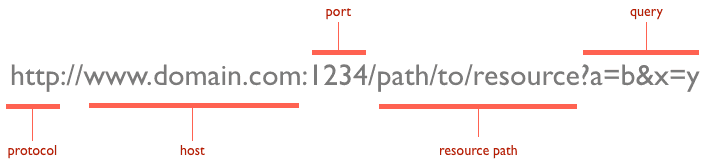
## HTTP là viết tắt của Hypertext Transfer Protocol. Đó là tập hợp các quy tắc chuẩn dành cho việc truyền và nhận dữ liệu.

## Được dùng để liên hệ thông tin giữa Máy cung cấp dịch vụ (Web server) và Máy sử dụng dịch vụ (Web client) là giao thức Client/Server dùng cho World Wide Web-WWW, HTTP là một giao thức ứng dụng của bộ giao thức TCP/IP (các giao thức nền tảng cho Internet).

## ****URLs****

Trọng tâm của truyền thông web là thông báo yêu cầu, được gửi thông qua Uniform Resource Locators (URL), dùng để tham chiếu tới tài nguyên trên Internet.

Một URL có cấu trúc như dưới đây.



**Protocol** : chính là Giao thức sử dụng, thường là HTTP hoặc HTTPs. Trong trường hợp truyền nhận file thì là FTP.

**Host** : cái này chính là một tên miền hoặc một địa chỉ IP của một máy tính nào đó nơi sẽ xử lý của yêu cầu từ Client.

**Port** : Số cổng mà server đang chạy để lắng nghe việc xử lý các Request HTTP.

Bạn có thể hình dung một khu phố có rất nhiều ngôi nhà, mỗi ngôi nhà đều có địa chỉ riêng của mình, tương tự máy tính cũng vậy, tại một thời điểm máy tính có thể có rất nhiều phần mềm cùng chạy. Mỗi phần mềm sẽ chạy trên một Cổng (Port) khác nhau. Việc định rõ số cổng chính là việc xác định ứng dụng nào đang chạy trên máy tính đó sẽ xử lí yêu cầu của bạn.

Thông thường dịch vụ web sẽ chạy ở cổng 80 của máy tính theo mặc định.

**Resource Path** : Đây chính là đường đến tài nguyên, đây là phần cực kỳ quan trọng, xác định tài nguyên nào sẽ được yêu cầu.

**Query**: Đây là tập các đối số (Param) gửi kèm theo để cung cấp thêm thông tin.

## ****HTTP Request methods****

Giống như tương tác giữa người với người, khi bạn nói là bạn yêu cầu ai đó thì bạn sẽ phải định rõ yêu cầu là gì, hay cụ thể hơn tùy vào yêu cầu của bạn mà người kia sẽ đáp lại như thế nào.

Giao thức HTTP định nghĩa ra một tập phương thức yêu cầu ( Request methods ) để bạn sử dụng trong việc tương tác với máy chủ . cụ thể là :

**GET :**GET được sử dụng để lấy thông tin từ Server.

Khi bạn truy câp một trang web từ trình duyệt thông qua URL thì bạn đang thực hiện một GET request trên URL đó để lấy dữ liệu.

**POST :**Một yêu cầu POST được sử dụng để gửi dữ liệu tới Server.

Khi bạn nhập mật khẩu và password trong một trang web nào đó và nhấp Login, thì lúc này bạn đang gửi dữ liệu của bạn đến máy chủ của trang web đó thông qua phương thức POST.

**PUT :**Phương thức này được dùng với ý nghĩa là bạn đang thay đổi, cập nhật lại nội dung của một nguồn dữ liệu nào đó.

**DELETE :**Phương thức này sử dụng với ý nghĩa bạn muốn xóa một tài nguyên nào đó.

## ****Status Code****

Tiếp theo kế sau các Request Methods chính là các Status Code ( Mã trạng thái ).

Mã trạng thái là một con số để xác định trạng thái sau khi thực thi một yêu cầu HTTP.

Cấu trúc mã Status Code chia thành các loại dưới đây:

* **2xx**: Successful (Thông báo thành công)
* **3xx**: Redirection (Yêu cầu chuyển hướng)
* **4xx**: Client Error (Người dùng gặp lỗi)
* **5xx**: Server Error (Server gặp lỗi)

## ****Cấu trúc một gói tin HTTP****

Cấu trúc một gói tin HTTP bao gồm các thành phần sau:

* Request Line : Chứa Request method và đường dẫn URL của tài nguyên đích.
* Header: Chứa các thông tin của thiết bị thực hiện Request này.
* Body : Phần thân chứa dữ liệu của Request.

## Express Framework

## Khái niệm cơ bản

Express là một module hay nói theo cách thường được nhắc đến thì nó là một framework trong node.js. Express framework cung cấp cho chúng ta đầy đủ tiện ích để có thể xây dựng một ứng dụng web hay một API.

Các ưu điểm các bạn sẽ nhìn thấy được khi sử dụng express framework:

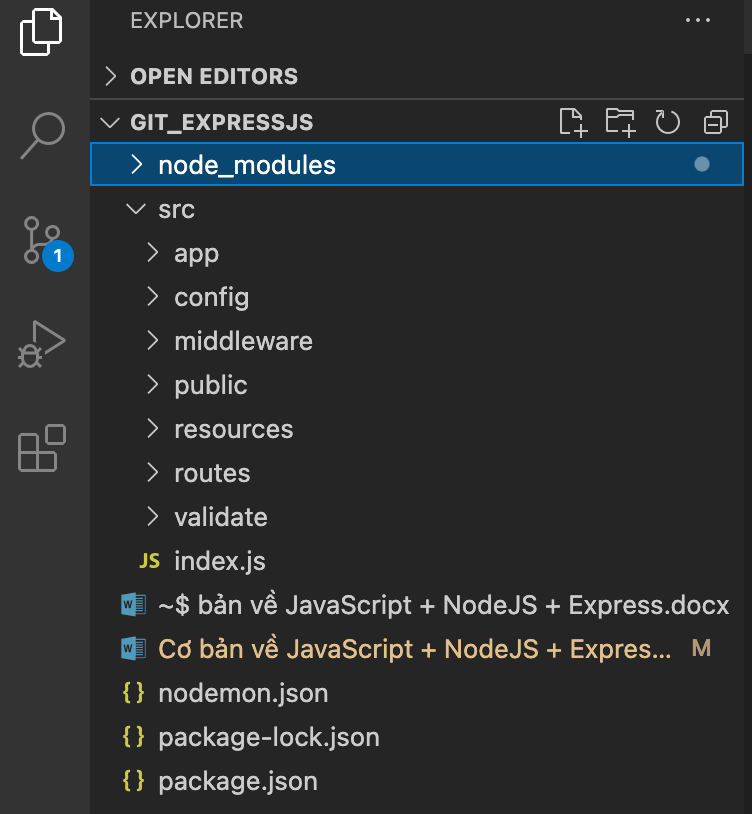
* Express hỗ trợ chúng ta phát triển ứng dụng theo mô hình MVC.
* Cho phép định nghĩa các middleware.
* Định nghĩa rõ ràng các request methods trong route.
* Hỗ trợ mạnh về REST API.

## Cài đặt Express

Do Express framework không được tích hợp sẵn vào trong node.js nên khi muốn sử dụng nó chúng ta phải tải nó về máy với sử dụng được. Và để cải đặt nó chúng ta sử dụng cú pháp của npm.

npm install express --save

Sau khi cài đặt xong thì trong thư mục dự án của bạn sẽ có thêm một thư mục *node\_modules* (đây là thư mục chứa các package download từ npm).



## Các tính năng của ExpressJS

ExpressJS có những tính năng cụ thể như sau:

* **Phát triển máy chủ nhanh chóng**

Express.js cung cấp nhiều tính năng thường được sử dụng của Node.js dưới dạng các hàm có thể dễ dàng sử dụng ở bất kỳ đâu trong chương trình. Điều này loại bỏ nhu cầu viết mã trong vài giờ và do đó tiết kiệm thời gian.

* **Phần mềm trung gian – Middleware**

Middleware – là một phần của chương trình có quyền truy cập vào cơ sở dữ liệu, yêu cầu của khách hàng và các phần mềm trung gian khác. Nó chịu trách nhiệm chính cho việc tổ chức có hệ thống các chức năng khác nhau của Express.js.

* **Định tuyến – Routing**

Express.js cung cấp cơ chế định tuyến nâng cao giúp duy trì trạng thái của trang web với sự trợ giúp của URL của chúng.

* **Tạo mẫu – Templating**

Expressjs cung cấp các công cụ tạo khuôn mẫu cho phép các nhà phát triển xây dựng nội dung động trên các trang web bằng cách xây dựng các mẫu HTML ở phía máy chủ.

* **Gỡ lỗi – Debugging**

Gỡ lỗi là rất quan trọng để phát triển thành công các ứng dụng web. ExpressJS giúp việc gỡ lỗi trở nên dễ dàng hơn bằng cách cung cấp cơ chế gỡ lỗi có khả năng xác định chính xác phần ứng dụng web có lỗi.

## Middleware trong ExpressJS

ExpressJS khi hoạt động, về cơ bản sẽ là một loạt các hàm **Middleware** được thực hiện liên tiếp nhau. Sau khi đã thiết lập, các request từ phía người dùng khi gửi lên ExpressJS sẽ thực hiện lần lượt qua các hàm Middleware cho đến khi trả về response cho người dùng. Các hàm này sẽ được quyền truy cập đến các đối tượng đại diện cho Request - **req**, Response - **res**, hàm Middleware tiếp theo - **next**, và đối tượng lỗi - **err** nếu cần thiết.

Một hàm Middleware sau khi hoạt động xong, nếu chưa phải là cuối cùng trong chuỗi các hàm cần thực hiện, sẽ cần gọi lệnh next() để chuyển sang hàm tiếp theo, bằng không xử lý sẽ bị treo tại hàm đó.

**Các chức năng mà middleware có thể thực hiện trong ExpressJS sẽ bao gồm :**

* Thực hiện bất cứ đoạn code nào
* Thay đổi các đối tượng request và response
* Kết thúc một quá trình request-response
* Gọi hàm middleware tiếp theo trong stack

**Trong Express, có 5 kiểu middleware có thể sử dụng :**

* Application-level middleware (middleware cấp ứng dụng)
* Router-level middleware (middlware cấp điều hướng - router)
* Error-handling middleware (middleware xử lý lỗi)
* Built-in middleware (middleware sẵn có)
* Third-party middleware (middleware của bên thứ ba)