Đề Cương

Các Công Nghệ Lập Trình Hiện Đại

Nhóm 10

Thành viên:

Hoàng Phương Nam – 3116410075

Hồ Thanh Huy – 3116410041

Phạm Minh Nhật – 3116410179

Trần Quang Danh - 3116410010

Mục lục

[1 Giới thiệu Node.js 1](#_Toc43333738)

[1.1 Node.js là gì? 1](#_Toc43333739)

[1.2 Lịch sử Node.js 1](#_Toc43333740)

[1.3 Các đặc trưng của Node.js 1](#_Toc43333741)

[1.4 Những hiểu lầm khi học lập trình node.js 2](#_Toc43333742)

[1.5 Những lí do nên học lập trình node.js 2](#_Toc43333743)

[1.6 Cách hoạt động của Node.js 2](#_Toc43333744)

[1.7 Khi nào nên sử dụng Node.js? 3](#_Toc43333745)

[1.7.1 Nên sử dụng 3](#_Toc43333746)

[1.7.2 Không nên sử dụng 3](#_Toc43333747)

[1.8 Những công ty đang sử dụng Node.js 3](#_Toc43333748)

[2 Cài đặt Node.js 3](#_Toc43333749)

[3 Tạo các ứng dụng đơn giản với Node.js 12](#_Toc43333750)

[3.1 Chạy file JavaScript 12](#_Toc43333751)

[3.2 Tạo server sử dụng HTTP 14](#_Toc43333752)

[4 Module (Package) 15](#_Toc43333753)

[4.1 Khái niệm module trong Node.js 15](#_Toc43333754)

[4.2 Sử dụng module 15](#_Toc43333755)

[4.3 Tạo module 16](#_Toc43333756)

[4.4 Kế thừa (mở rộng hoặc sử dụng lại) module 17](#_Toc43333757)

[4.5 Xuất bản module 18](#_Toc43333758)

[4.6 Sử dụng npm để quản lý các module 21](#_Toc43333759)

[4.6.1 Cài đặt module 22](#_Toc43333760)

[4.6.2 Xem danh sách các module đã cài đặt 22](#_Toc43333761)

[4.6.3 Nâng cấp phiên bản 22](#_Toc43333762)

[4.6.4 Tìm kiếm 22](#_Toc43333763)

[4.6.5 Dở cài đặt 22](#_Toc43333764)

[4.7 File package.json 22](#_Toc43333765)

[4.8 File package-lock.json 24](#_Toc43333766)

[4.9 Thư mục node\_modules 25](#_Toc43333767)

[5 Tạo các ứng dụng cơ bản với Node.js 26](#_Toc43333768)

[6 Giới thiệu về đồ án của nhóm 30](#_Toc43333769)

[7 Tài liệu kham khảo 46](#_Toc43333770)

# Giới thiệu Node.js

## Node.js là gì?

* Node.js là 1 nền tảng giúp chạy JavaScript phía server.
* Node.js là chạy trên môi trường V8 JavaScript runtime - một trình thông dịch JavaScript cực nhanh chạy trên trình duyệt Chrome.
* Node.js là 1 nền tảng mã nguồn mở, đa nền tảng dùng để phát triển ứng dụng web phía server.
* Nodejs không phải là một ngôn ngữ lập trình. Bản chất Nodejs là một JavaScript runtime.
* Để học được Nodejs thì bạn cần phải biết về Javascript, kỹ thuật lập trình, một số giao thức…

## Lịch sử Node.js

* Node.js là một hệ thống phần mềm được thiết kế để viết các ứng dụng internet có khả năng mở rộng, đặc biệt là máy chủ web.
* Chương trình được viết bằng JavaScript, sử dụng kỹ thật điều khển theo sự kiện, nhập/xuất không đồng bộ để tối tiểu tổng chi phí và tối đại khả năng mở rộng. Node.js bao gồm có V8 JavaScript engine của Google, libUV, và vài thư viện khác.
* Node.js được tạo bởi Ryan Dahl từ năm 2009, và phát triển dưới sự bảo trợ của Joyent.
* Mục tiêu ban đầu của Dahl là làm cho trang web có khả năng push như trong một số ứng dụng web như Gmail. Sau khi thử với vài ngôn ngữ Dahl chọn Javascript vì một API Nhập/Xuất không đầy đủ. Điều này cho phép anh có thể định nghĩa một quy ước Nhập/Xuất điểu khiển theo sự kiện, non-blocking.
* Vài môi trường tương tự được viết trong các ngôn ngữ khác bao gồm Twisted cho Python, Perl Object Environment cho Perl, libevent cho C và EventMachine cho Ruby.
* Khác với hầu hết các chương trình Javascript, Nodejs không chạy trên một trình duyệt mà chạy trên Server.
* Node.js sử dụng nhiều chi tiết kỹ thuật của CommonJS. Nó cung cấp một môi trường REPL (read-eval-print-loop) cho kiểm thử tương tác.
* Node.js được InfoWorld bình chọn là “Công nghệ của năm” năm 2012.
* Để bắt đầu dùng Node.js, bạn phải hiểu sự khác nhau giữa Node.js với các môi trường truyền thống chạy trên server (server side) phổ biến như PHP, Python, Ruby, etc

## Các đặc trưng của Node.js

* Node.js sử dụng kiến trúc hướng sự kiện không đồng bộ và cơ chế Non-blocking I/O giúp xử lý các yêu cầu nhập xuất đồng thời. Có nghĩa là Node khi nhận được 1 yêu cầu nhập xuất (vd: nhập xuất file), nó sẽ xử lý ở dưới nền và tiếp tục xử lý các yêu cầu khác.

var fs = require('fs');

fs.readFile("Sample.txt",function(error,data) {

console.log("Reading Data completed");

});

//code1

Figure 1: Code demo

* Vd: ở các ngôn ngữ lập trình khác, code1 chỉ được xử lý khi toàn bộ file đã được đọc. Node.js có thể xử lý tiếp code1 trong khi file đang được đọc dưới nền.
* fs.readFile là phương thức không đồng bộ có tham số thứ 2 là 1 phương thức khác, phương thức này được gọi là **callback function** và sẽ được gọi sau khi file đã được đọc xong.
* Nodejs không hỗ trợ đa luồng, nó là một máy chủ đơn luồng.
* Node sử dụng môi trường V8 JavaScript runtime, là trình thông dịch JavaScript chạy trên Google Chrome. Node bao bọc môi trường JavaScript là cho môi trường runtime này nhanh hơn.
* Node có thể xử lý các kết nối đồng thời trên một tiến trình với chi phí tối thiểu.
* Sử dụng JavaScript, là 1 ngôn ngữ phổ biến phía client, giúp việc học Node trở nên dễ dàng hơn.
* Có một cộng đồng lớn sử dụng, hỗ trợ lẫn nhau và lượng thư viện đa dạng giúp Node luôn luôn cập nhật được các xu hướng phát triển web mới nhất.

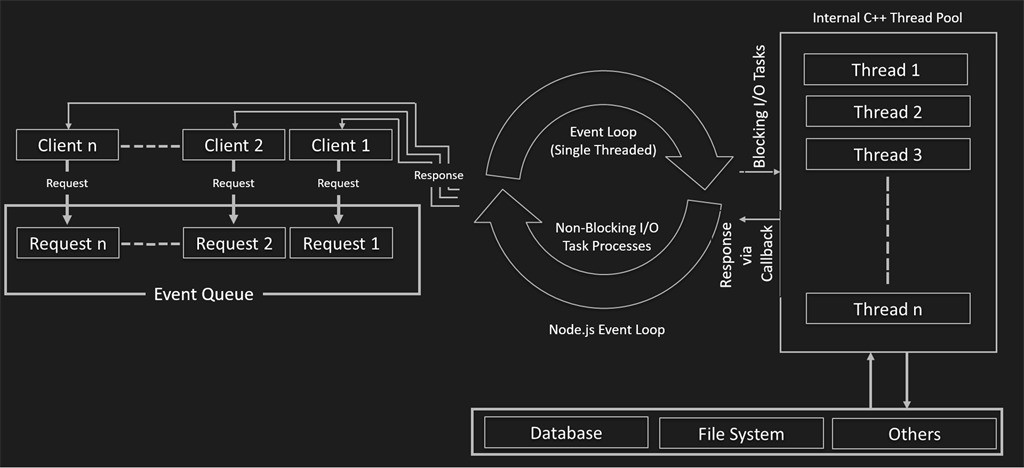
## Những hiểu lầm khi học lập trình node.js

* Thứ nhất, Nodejs không phải là một ngôn ngữ lập trình. Bản chất Nodejs là một JavaScript runtime
* Thứ hai, Nodejs không hỗ trợ đa luồng, nó là một máy chủ đơn luồng.
* Cuối cùng, chính vì Nodejs không phải là ngôn ngữ lập trình, nên để học được Nodejs thì bạn cần phải biết về Javascript, kỹ thuật lập trình, một số giao thức…

## Những lí do nên học lập trình node.js

* Thứ nhất: Nodejs sử dụng Javascript là ngôn ngữ để xây dựng ứng dụng. Đây là một ngôn ngữ rất phổ biến cho lập trình front-end, ứng dụng client.
* Thứ hai: Nodejs chạy đa nền tảng phía Server(Mac – Window – Linux đều được), sử dụng kiến trúc hướng sự kiện Event-driven, cơ chế non-blocking I/O làm cho nó nhẹ và hiệu quả.
* Thứ ba: Cộng đồng Nodejs đang rất phát triển, thư viện miễn phí cực nhiều.

## Cách hoạt động của Node.js



* Non-Blocking I/O: các công việc nhập xuất mà không dừng việc xử lý các dòng code tiếp theo.
* Blocking I/O: các công việc nhập xuất mà dừng việc xử lý các dòng code tiếp theo. Vd: nhập xuất Database, File, …
* Event Queue: hàng đợi các Request từ Client.
* Event Loop: việc xử lý đơn luồng của Node.js, lặp đi lặp lại 2 công việc:
  + Lấy yêu cầu từ Event Queue.
  + Xử lý yêu cầu vừa lấy từ Event Queue.
* Thread Pool: nơi chưa các luồng đợi Server cấp việc.
* Bước 1: client gửi tới server.
* Bước 2: các Request được xếp vào hàng đợi Event Queue.
* Bước 3: server lấy 1 Request từ Event Queue.
  + Nếu Request là Non-Blocking I/O Task thì Server xử lý và phản hồi lại Client.
  + Nếu Request là Blocking I/O Task thì Server chỉ định 1 luồng từ Theard Pool để xử lý.
  + Quay lại bước 3.
  + Khi đã xử lý xong Blocking I/O Task thì luồng này phản hồi lại cho Event Loop xử lý bằng **callback function**.

## Khi nào nên sử dụng Node.js?

### Nên sử dụng

* Node.js sử dụng tốt nhất cho các ứng dụng trực tuyến như:
  + Ứng dụng chat.
  + Máy chủ game. Máy chủ cần phải nhanh và hiệu xuất cao để xử lý nhiều yêu cầu cùng 1 lúc.
  + Các ứng dụng quản lý tập tin, project, có nhiều người truy cập và chỉnh sửa.
  + Các ứng dụng streaming.

### Không nên sử dụng

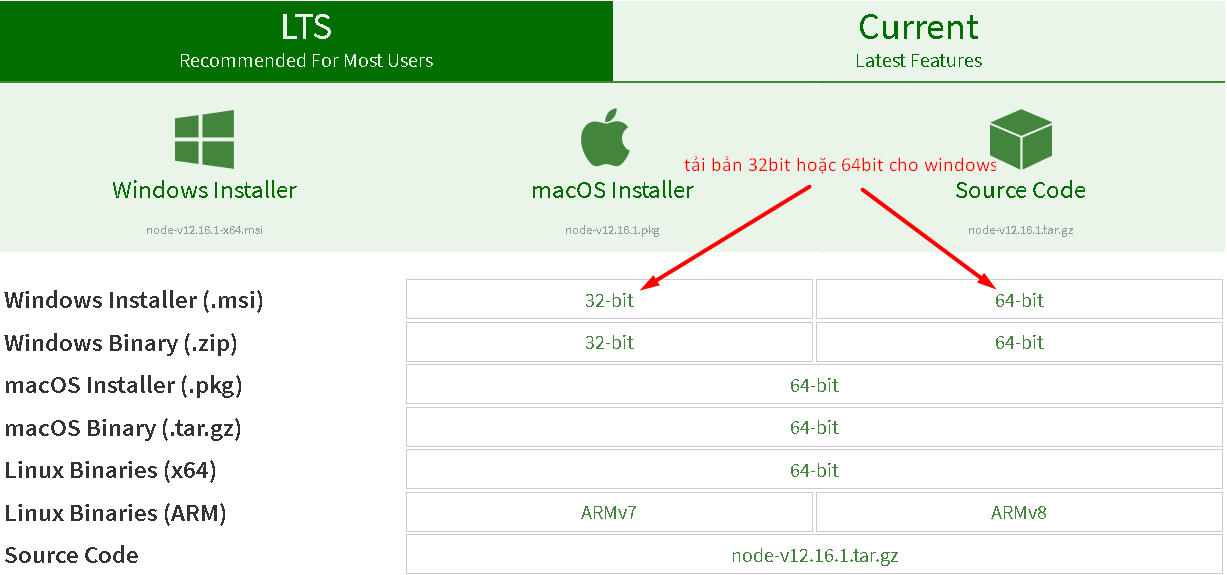
* Các ứng dụng cần tính toán, xử lý lâu bởi CPU. Khi đó máy chủ không thể xử lý thêm bất cứ yêu cầu nào cho tới khi công việc tính toán hoàn tất.
* Vì giống như JavaScript, Node.js không hỗ trợ đa luồng. Có nghĩa là 1 lúc nó chỉ có thể xử lý 1 yêu cầu.

## Những công ty đang sử dụng Node.js

|  |  |  |  |
| --- | --- | --- | --- |
| Paypal | LinkedIn | Mozilla | Ebay |
|  |  |  |  |

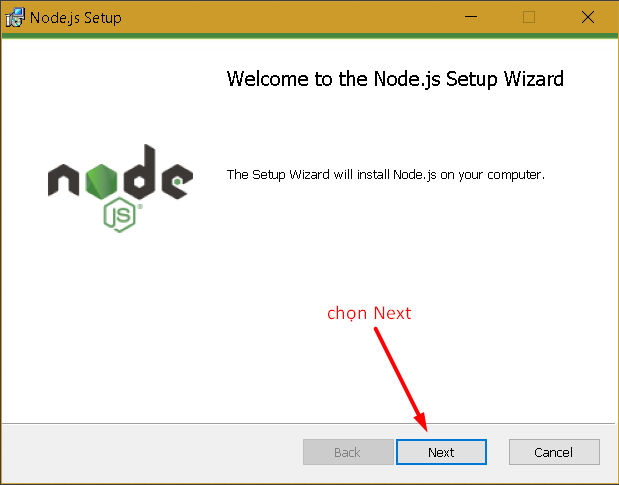
# Cài đặt Node.js

* Tải Node.js từ link [**https://nodejs.org/en/download/**](https://nodejs.org/en/download/).

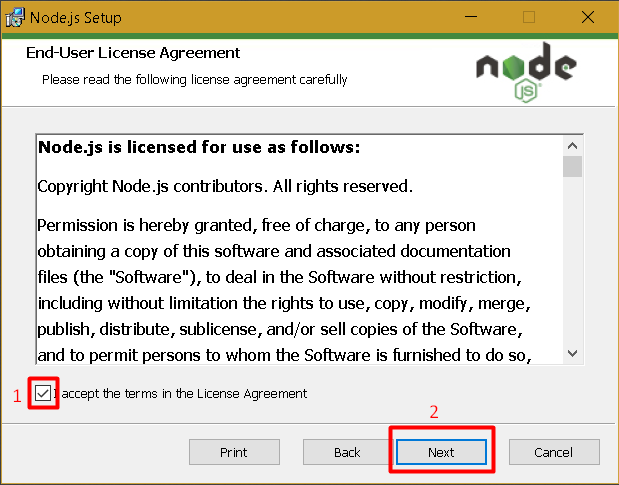


Hình 1: Giao diện tải Node.js.

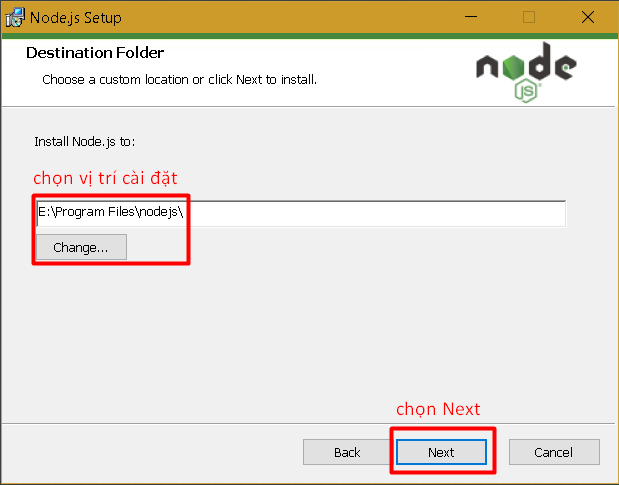
* Chạy installer.



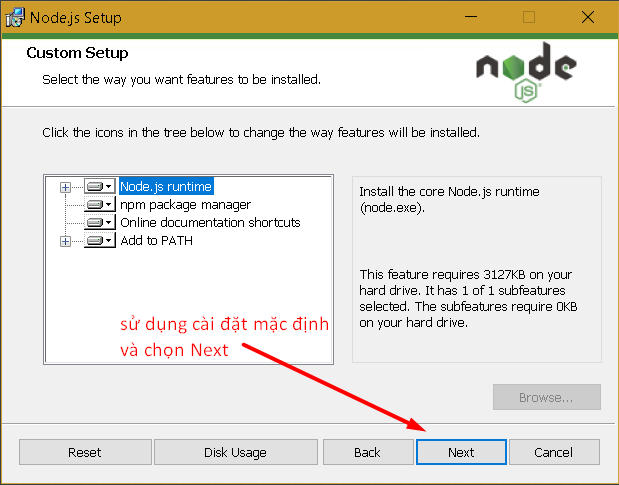
Hình 2: Cài đặt Node.js



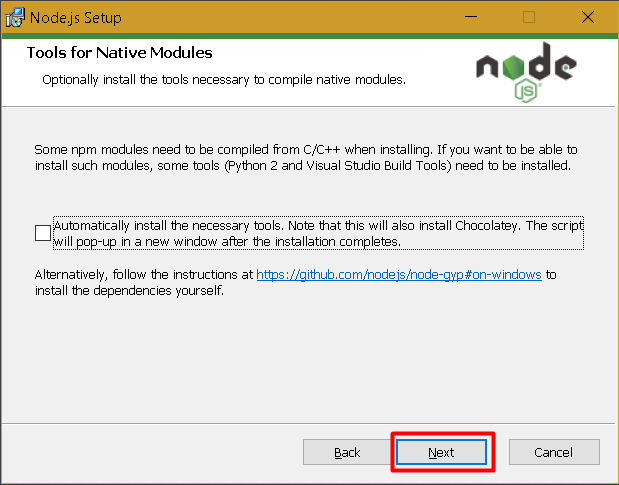
Hình 3: Cài đặt Node.js



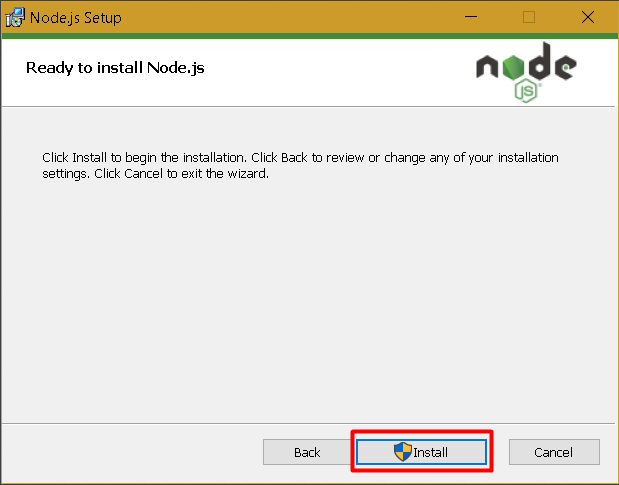
Hình 4: Cài đặt Node.js



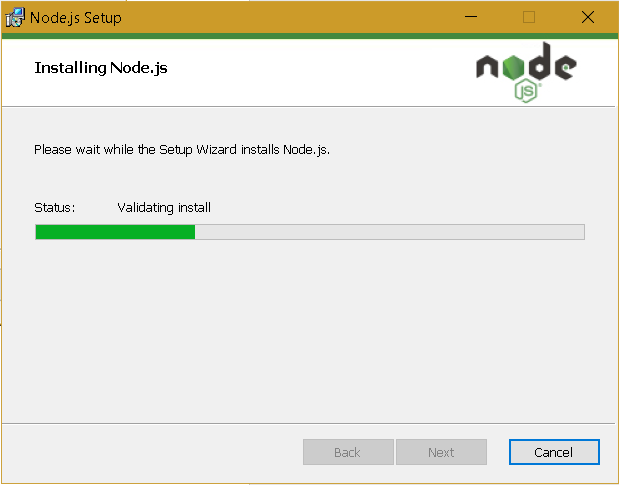
Hình 5: Cài đặt Node.js



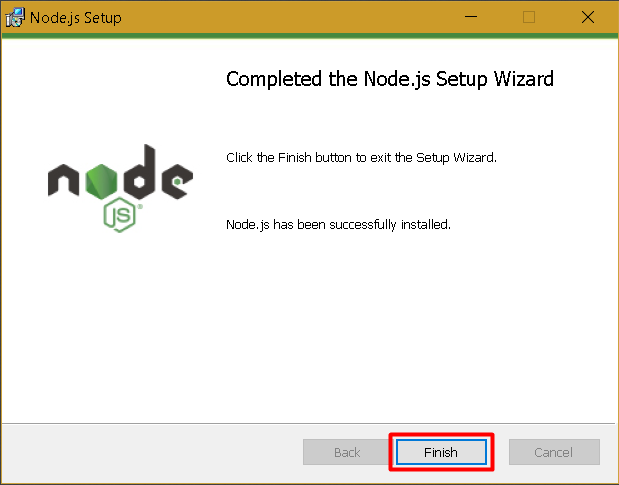
Hình 6: Cài đặt Node.js



Hình 7: Cài đặt Node.js



Hình 8: Chờ Node.js cài đặt xong



Hình 9: Hoàn tất cài đặt Node.js

# Tạo các ứng dụng đơn giản với Node.js

## Chạy file JavaScript

* Node.js là cách để chạy JavaScript phía Server.
* Tạo file JavaScript **addition.js** chứa nội dung:

let a = 5;

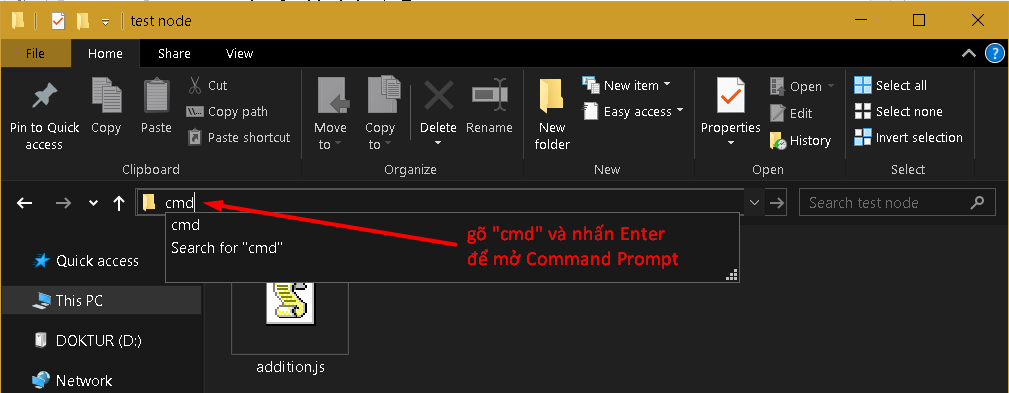
let b = 3;

let c = a + b;

console.log("a + b = " + c);

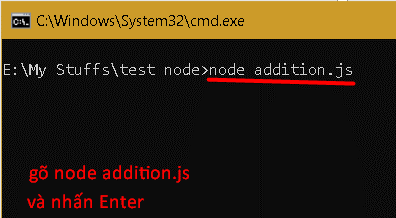
Figure 2: Nội dung file addition.js

* Mở **Command Prompt** ở thư mục chứa file **addition.js.**

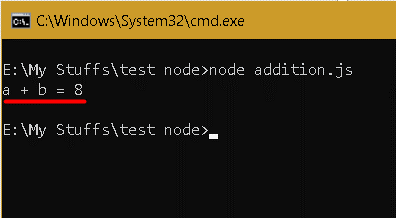


Hình 10: Mở Command Prompt

* Gõ node addition.js và nhấn nút **Enter**.



Hình 11: Chạy file JavaScript phía Server



Hình 12: Kết quả chạy file addition.js

## Tạo server sử dụng HTTP

* Tạo file JavaScript **web.js** chứa nội dung:

var http = require('http');

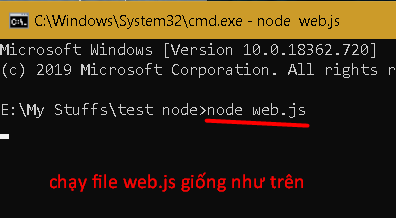
http.createServer(function (req, res) {

res.writeHead(200, {'Content-Type': 'text/html'});

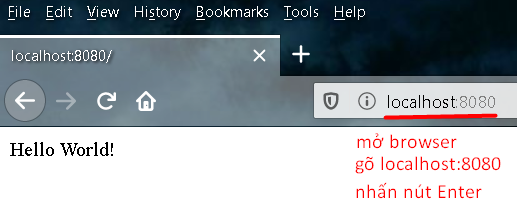
res.end('Hello World!');

}).listen(8080);

Figure 3: Nội dung file web.js



Hình 13: Chạy file web.js



Hình 14: Thử kết quả sau khi chạy file web.js

* Nhấn **Ctrl+C** ở Command Prompt hoặc tắt Command Prompt để dừng lại.
* Giải thích:
  + Phương thức require có chức năng đọc file JavaScript, thực thi file và trả về đối tượng. Ta sử dụng đối tượng này để gọi các phương thức, biến của module trong file JavaScript đó.
  + http là module của Node.js, cho phép vận chuyển dữ liệu qua giao thức HTTP.
  + http.createServer để tạo ra 1 Server dựa trên phương thức truyền vào nó. Phương thức này được gọi mỗi khi có yêu cầu tới Server.
  + Phương thức writeHead dùng để gửi dữ liệu header tới cho client, phương thức end để đóng kết nối với client.
  + Phương thức listen(8080) làm cho server mở ở cổng 8080.

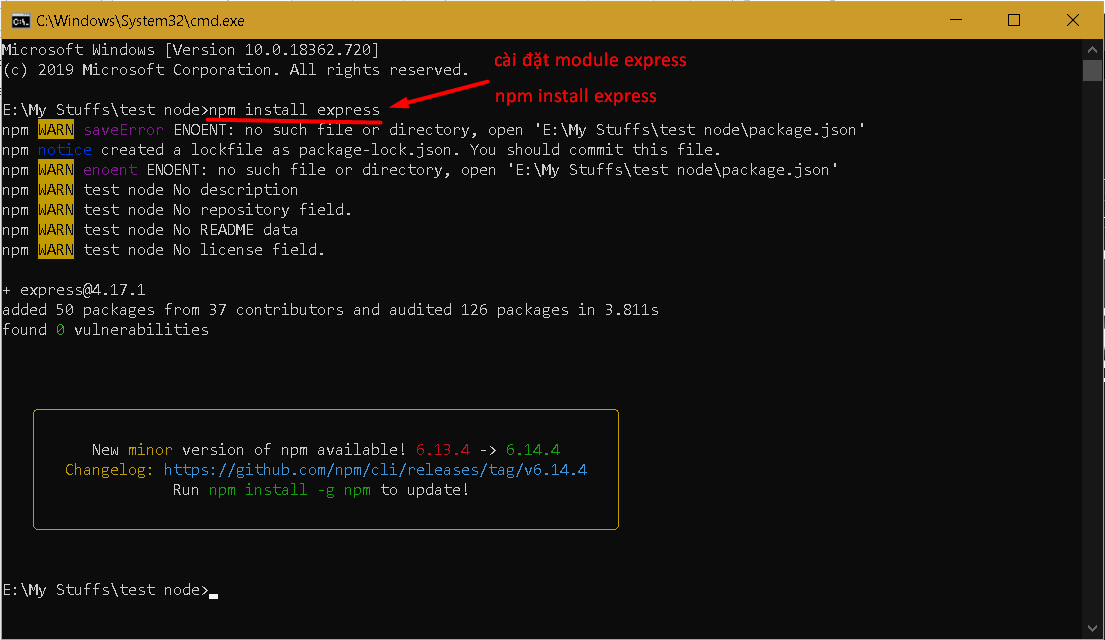
# Module (Package)

## Khái niệm module trong Node.js

* Module trong Node.js là cách đóng gói code thành 1 đơn vị duy nhất để dễ dàng quản lý và mở rộng cho tương lai. Một số module phổ biến: Express, Socket.io, Jade, …

## Sử dụng module

* Để sử dụng được module, trước hết là phải tải về.
* Chạy lệnh giống như hình dưới trong Command Prompt để tải module express về thư mục mà Command Prompt đang chỉ tới.



Hình 15: Cài đặt module express

* Tiếp theo để sử dụng module này, ta dùng phương thức require như demo trước đó.

var express = require('express');

Figure 4: Sử dụng module vừa mới tải về

## Tạo module

* Tạo file first-test.js có nội dung như sau:

var exports = {};

exports.write = function() {

console.log("first test");

};

module.exports = exports;

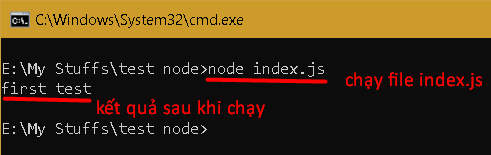
Figure 5: Nội dung first-test.js

* Giải thích code:
  + Tạo dynamic object tên là exports.
  + Gán phương thức vào thuộc tính write của object exports.
  + Làm cho object exports này có thể được sử dụng bởi các file khác bằng cách gán object này cho module.exports.
* Tiếp theo để sử dụng module vừa mới tạo, ta tạo file index.js có nội dung như sau:

var firstTest = require("./first-test");

firstTest.write();

Figure 6: Nội dung file index.js



Hình 16: Chạy file index.js và kết quả

## Kế thừa (mở rộng hoặc sử dụng lại) module

* Tạo file second-test.js có nội dung như sau:

var firstTest = require("./first-test");

module.exports = function() {

firstTest.write();

console.log("second test");

};

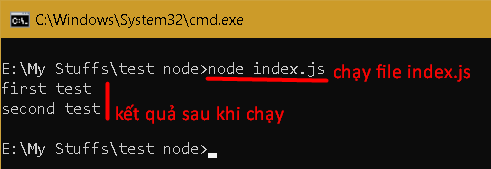
Figure 7: Nội dung file second-test.js

* Để kế thừa module first-test, ta sử dụng require("./first-test").
* Trong phương thức của module second-test ta gọi lại phương thức write của module first-test.
* Thay đổi nội dung file index.js như sau:

var secondTest = require("./second-test");

secondTest();

Figure 8: Nội dung file index.js



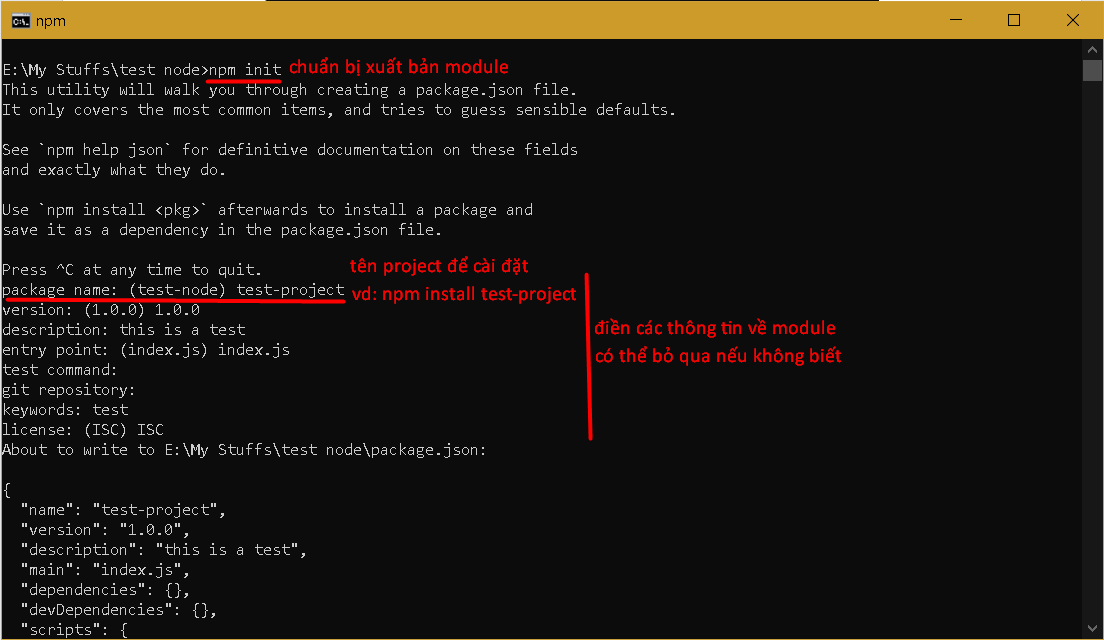
Hình 17: Chạy file index.js và kết quả

## Xuất bản module

* Xuất bản module vào Github repository của mình để ta có thể sử dụng lại module của mình ở bất kỳ đâu mà không cần lưu trữ cục bộ.
* Đầu tiên ta cần tạo tài khoản npm. Link đăng ký: <https://www.npmjs.com/signup>.

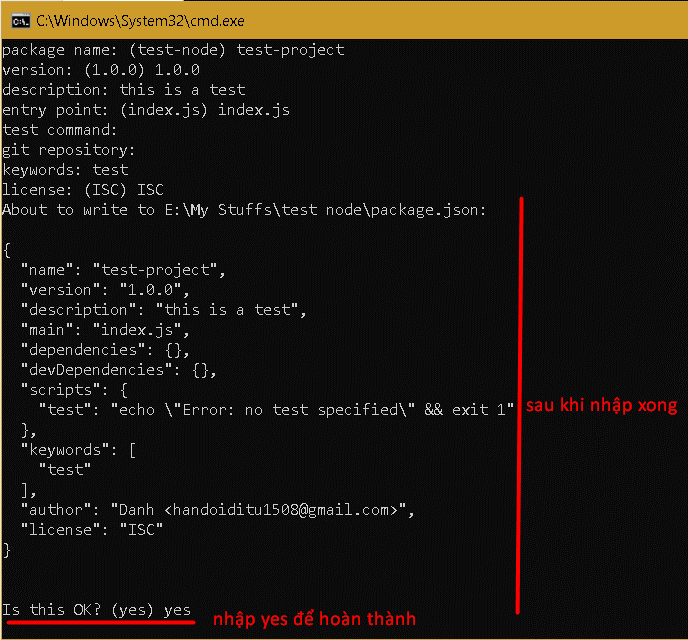


Hình 18: Đăng nhập npm

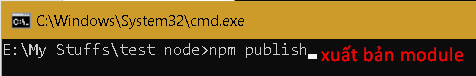


Hình 19: Nhập thông tin về module

* Sau khi chạy npm init, Node.js sẽ tạo ra file package.json.



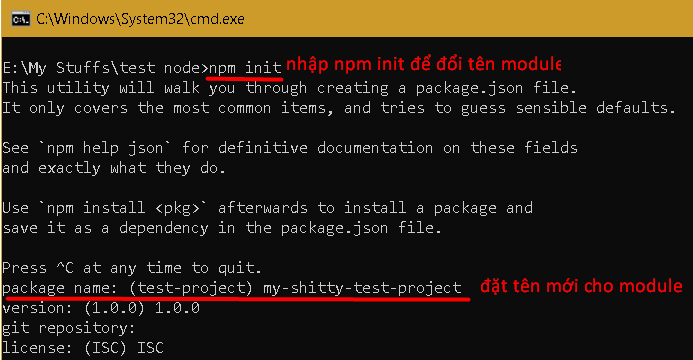
Hình 20: Hoàn thành nhập thông tin



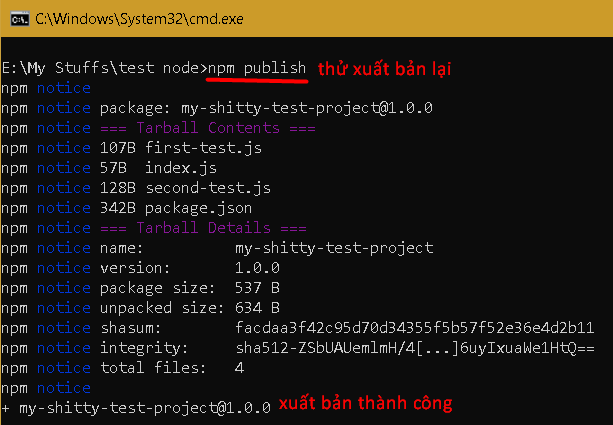
Hình 21: Xuất bản module



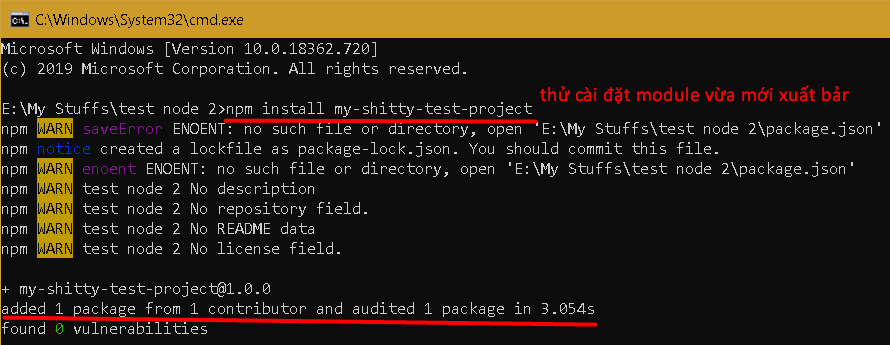
Hình 22: Lỗi khi xuất bản



Hình 23: Thay đổi tên module



Hình 24: Xuất bản thành công



Hình 25: Cài đặt module vừa xuất bản

* Sử dụng module: require(‘my-shitty-test-project’).

## Sử dụng npm để quản lý các module

* Npm(Node Package Manager) là chương trình quản lý thư viện (package manager) ngầm định trong môi trường Javascript Node.js. Bao gồm một trình gọi dòng lệnh (Command Line) từ máy khách Client với tên gọi là npm, và cơ sở dữ liệu trực tuyến chứa các gói public và private còn đuợc gọi là npm registry. Npm registry đuợc gọi từ npm client và trên web browser. Tất cả được quản lý trực tiếp từ NPM, inc.
* Lúc này dự án của chúng ta sẽ không cần phải copy thủ công. Thay vào đó, npm sẽ tự động download từ máy chủ về dự án của bạn.

### Cài đặt module

* Cài đặt trong thư mục mà Command Prompt đang trỏ tới: npm install [package-name]. Module chỉ tồn tại trong thư mục đó.
* Cài đặt module trên toàn bộ máy: npm install [package-name] --global hoặc npm install [package-name] -g.
* Cài đặt phiên bản cụ thể. Vd 4.11.5: npm install [package-name]@4.11.5.

### Xem danh sách các module đã cài đặt

* npm list --global

### Nâng cấp phiên bản

* npm update [package-name]

### Tìm kiếm

* npm search [package-name]

### Dở cài đặt

* npm uninstall [package-name]

## File package.json

Về cơ bản, Nodejs sẽ không là gì cả nếu không có các modules.

Có hai loại modules: Module built-in và module 3rd-party.

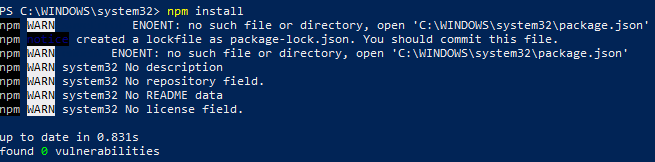
Có những modules cơ bản như: http, fs, os, path… sẽ được tích hợp sẵn trong Nodejs khi bạn cài đặt. Còn những modules kiểu 3rd-party thì khi nào bạn cần thì sẽ gọi.

Tất cả những module 3rd-party sẽ được quản lý thông qua file package.json( File này có vai trò giống như build.gradle trong Android vậy).

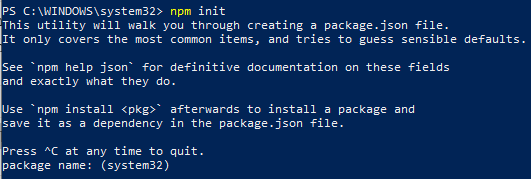
Đây là một ví dụ về package.json :



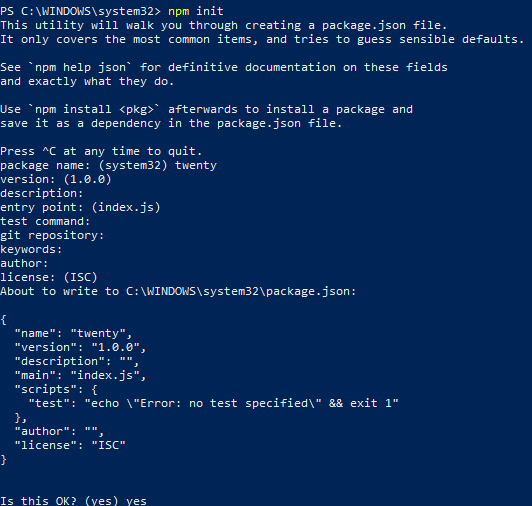
Để cài đặt tất cả các module được khai báo trong package.json :



Nếu bạn tạo mới dự án từ ban đầu, bạn có thể tạo package.json bằng lệnh :



Sau đó nhập thông tin cho command line :

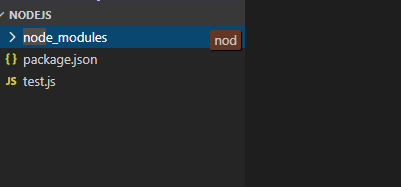


## File package-lock.json

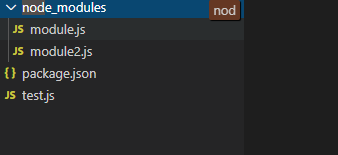
* Package-lock.json là file chứa các thông tin chi tiết của các modules được liệt kê trong file Package.json. Nó bao gồm các thông tin như phiên bản, nơi lưu trữ, những module cần thiết để chạy được module mình cần chạy. File Package-lock.json sẽ được tự động khởi tạo khi người dùng cài các module vào project của mình. Hỗ trợ cho quá trình nâng cấp ứng dụng hay sử dụng các công nghệ mới.

## Thư mục node\_modules

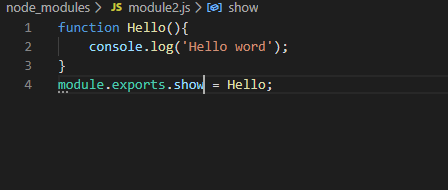
* Như phần giới thiệu về module trong nodejs ở phía trên, khi khởi tạo 1 module và muốn sử dụng lại module đó trong một file khác thì ta chỉ cần require đến đường dẫn tương đối file nơi khởi tạo module đó.Và ngoài ra chúng ta vẫn còn một cách khác mà không cần sử dụng đến các đường dẫn tương đối bằng cách đặt chúng vào một thư mục đặc biệt gọi là node\_modules.
* Trong thư mục chính ta tạo thêm 1 folder là node\_modules.



* Sau đó ta tạo một file module2.js trong thư mục này.



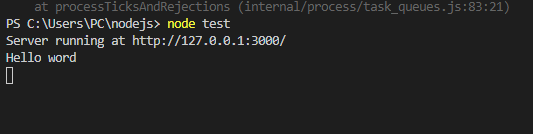
* File module2.js như sau:



* Rồi sau đó chúng ta qua test.js và sử dụng lại function trong file module2.js

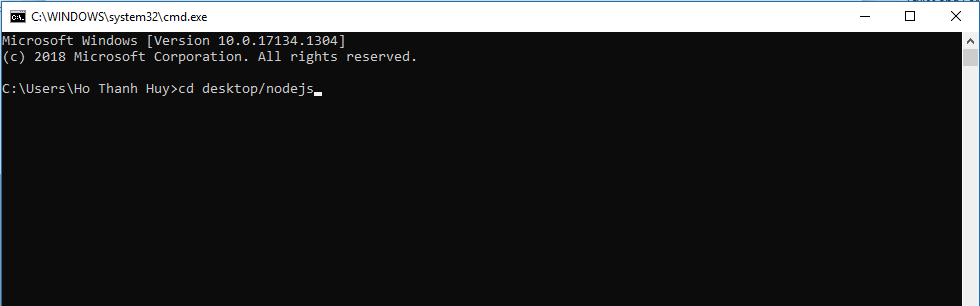


* Như chúng ta đã thấy việc require lại module không cần sử dụng đến dường dẫn tương đối.
* Run file:

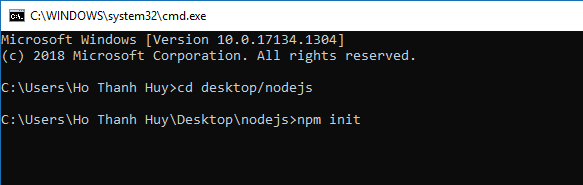


# Tạo các ứng dụng cơ bản với Node.js

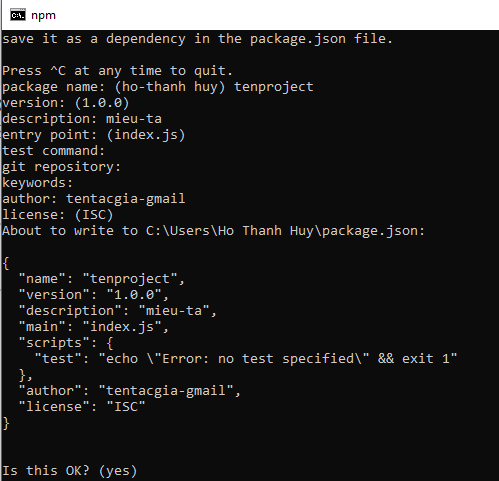
* Tạo một project mới:



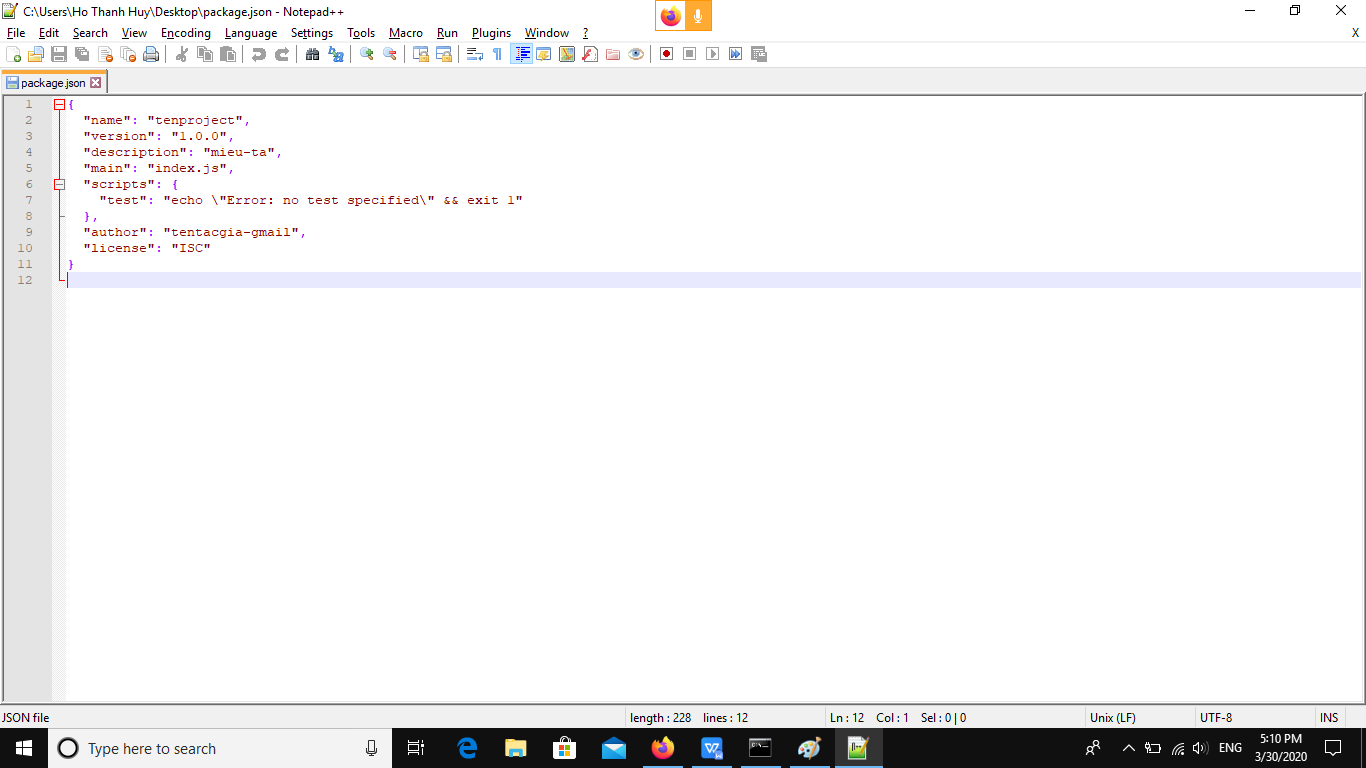
Hình 26: Mở CMD chuyển tởi thư mục tạo Project



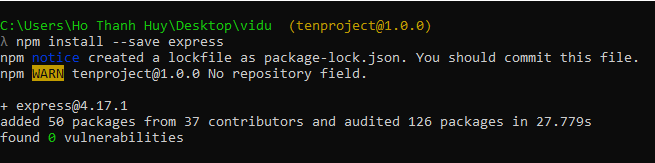
Hình 27: Nhập npm init để tạo một project mới



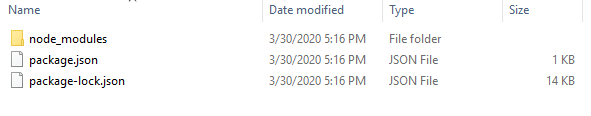
Hình 28: Nhập các thông tin của project



Hình 29: 1 file package.json được tạo ra

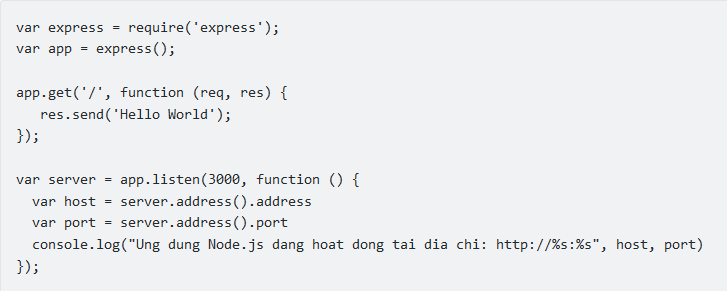


Hình 30: Gõ npm install --save express để tải module express và cài vào project



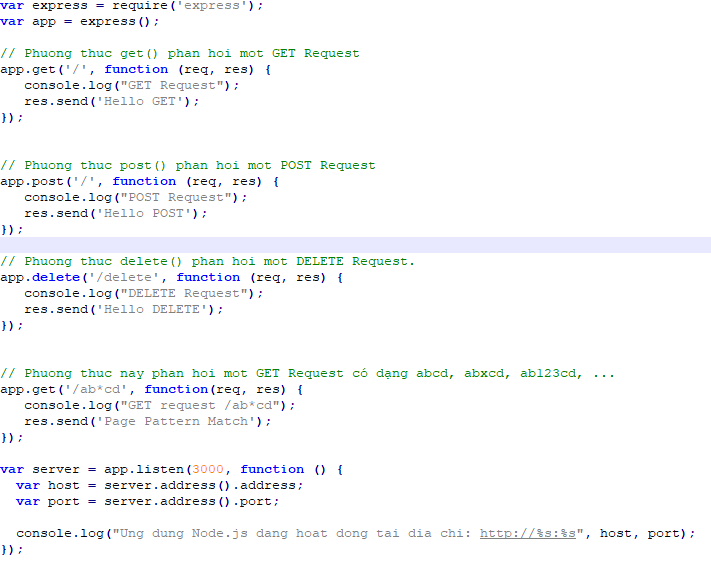
Hình 31: trong thư mục chứa file package.json xuất hiện thêm thư mục modules và package-lock.json

* Trong file package-lock.json chứa thông tin các module đã được cài đặt.
* Ngoài ra cũng có thể cài thêm một số module quan trọng đi kèm express như:
  + **body-parser** - Đây là một lớp trung gian node.js để xử lí JSON, dự liệu thô, text và mã hóa URL.
  + **cookie-parser**- Chuyển đổi header của Cookie và phân bố đến các req.cookies
  + **multer** - Đây là một thành phần trung gian trong node.js để xử lí phần multipart/form-data.
* Ứng dụng express cơ bản:



Express sử dụng một hàm callback có các tham số là các đối tượng request và response.

* Request - Biểu diễn một HTTP request và có các thuộc tính cho các request như các chuỗi truy vấn, tham số, body, HTTP header và những phần khác.
* Response - Biểu diễn một HTTP response được ứng dụng Express gửi đi khi nó nhận về một HTTP request.

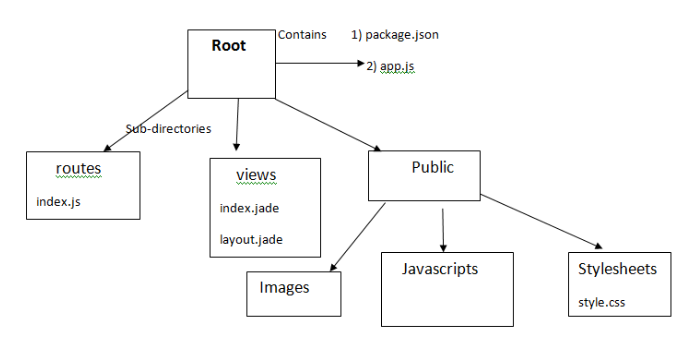


Hình 32: Router cơ bản trong express

# Giới thiệu về đồ án của nhóm

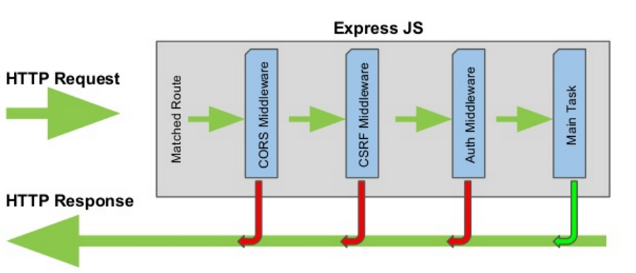
Trước khi đi vào demo của nhóm,thì em xin giới thiệu về framework em đã dùng để tạo một ứng dụng web trên server với nodejs,đó chính là Express.Nó cung cấp cho chúng ta rất nhiều tính năng mạnh mẽ trên nền tảng web cũng như trên các ứng dụng di động.Express hỗ trợ các phương thức HTTP và midleware tạo ra một API vô cùng mạnh mẽ và dễ sử dụng.Có thể tổng hợp một số chức năng chính:

* Thiết lập các lớp trung gian để trả về các HTTP request.
* Định nghĩa router cho phép sử dụng với các hành động khác nhau dựa trên phương thức HTTP và URL.
* Cho phép trả về các trang HTML dựa vào các tham số



***Đây là sơ đồ cây thể hiện cấu trúc một Express project***

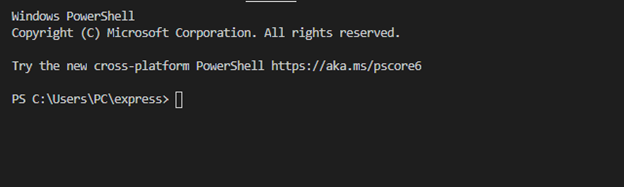
Cách mà Express hoạt động như sau:



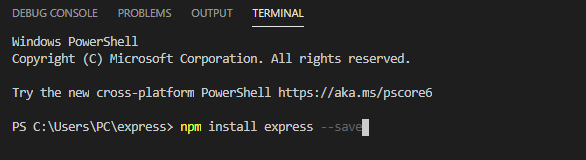
Tiếp đến là khởi tạo một project với framework ExpressJS.Để có thể hỗ trợ trong việc tìm hiểu và làm việc thì trang chính của Express là **http://expressjs.com** có hỗ trợ cho chúng ta document để có thể hiễu rõ hơn về từng phần.

Ở đây nhóm của chúng em dùng Visual Studio Code để viết app vì đây thích hợp chạy đa ngôn ngữ,đầu tiên mở visual code lên,rồi chuyển qua terminal với tổ hợp phím CRTL+ ~ hoặc có thể sử dụng trực tiếp bằng việc click vào terminal trên thanh công cụ và chọn new terminal.Sau đó chúng ta thực hiện các bước như sau:

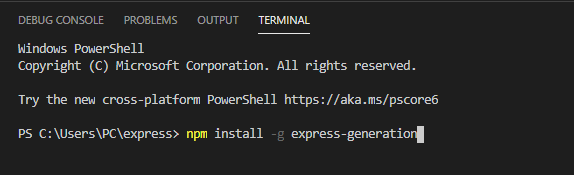
Bước 1: Gõ lệnh mkdir [tên folder project] để khởi ta thư mục chứa project.Hoặc chúng ta có thể tạo bên ngoài và mở,ở đây mình đã tạo sẵn 1 folder tên là express.



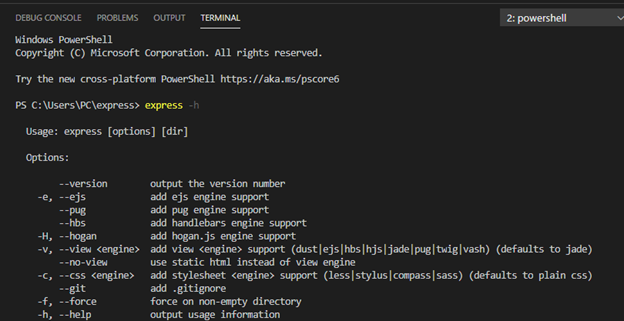
Bước 2:Chúng ta gõ lệnh npm install express –save để cài đặt framework Express về cho project.



Và đợi một khoảng thời gian ngắn để có thể tại về file cần thiết cho project.Sau đó tiếp tục ta thêm lệnh như sau:

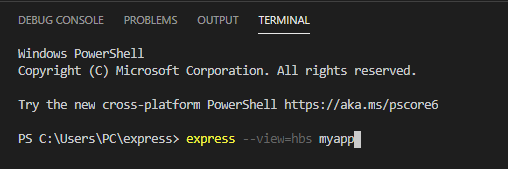


Dòng lệnh này cho phép chúng ta tạo ra 1 application nhanh hơn với cấu trúc đã được viết sẵn.Đợi 1 thời gian để hoàn thành việc cài đặt,và chúng ta gõ tiếp lệnh express –h để xem các option

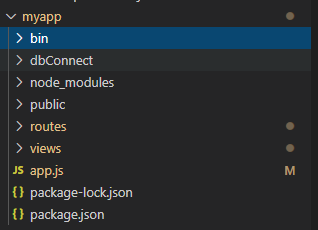


Ở đây sẽ show cho chúng ta những view engine mà express hỗ trợ trong việc thiết kế template và cách add view vào project.

Bước 3:Sau đó tiếp theo ta sẽ khởi tạo 1 Express app tên là myapp trong folder express với dòng lệnh express –view=hbs myapp.Tự đông sẽ khởi tạo thêm một folder myapp nơi chứa project và set view engine là hbs.



Và chúng ta sẽ thấy express-generation đã tạo ra 1 folder tên là myapp với cấu trúc thư mục như hình bên dưới.



Nói sơ qua về cấu trúc và ý nghĩa của các thư mục dưới đây:

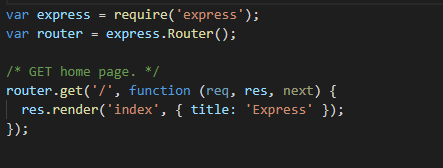
* Bin:thư mục này sẽ chứa một file js với chức năng là khởi tạo một server web.
* Node\_modules: Chứa các module của express-framework,hay nói đơn giản là thư viện hỗ trợ
* Public:Chứa các file như css,javascript,img,…
* Routes:Đường dẫn,có thể hiểu đơn giản nó giống với controller trong mô hình MVC với chức năng điều hướng xử lý các url.
* Views:Chứa template của project.
* App.js: nơi chúng ta cấu hình,khởi tạo các chức năng cần thiết hoặc sử dụng các module.

Sơ đồ tổng quát một project Express:



Nhìn vào sơ đồ cây ta sẽ hiểu rõ hơn về cấu trúc của Express app.Khá là giống với mô hình MVC của những ngôn ngữ khác,với routes đảm nhân vai trò xử lý như Controller,views thì hiện thị giá trị ,tương tác với người dùng,pulbic thì chứa các file image,bootstrap,.. để views sử dung.

Trước hết ta vào file index.js trong folder routes xem nội dung bên trong nó như thế nào:



*Hình ảnh về file index.js*

Routing đề cập đến việc xác định làm thế nào mà một ứng dụng có thể đáp ứng với yêu cầu mà phía client gửi lên cho server thông qua URL và các phương thức HTTP cụ thể như (GET,POST,…).Mỗi route có thể có một hoặc nhiều chức năng xử lý,được thực hiện khi routing.Được định nghĩa với cấu trúc như sau:



Chú thích:

* app là một ví dụ của express,nó tượng trưng cho appliaction mà chúng ta đã khởi tao.
* METHOD là một phương thức yêu cầu HTTP(get,post,..)
* PATH là đường dẫn trên máy chủ (URL)
* HANDLER là chức năng được thực thi khi route matched

Chẳng hạn lấy ví dụ về trang index.js ở trên hình,khi router nhân được đường dẫn ‘ / ‘ với phương thức GET,nó lập tực tạo ra một function với chức năng hiện thị trang chủ về cho phía client.Sau khi đã hiểu cách hoạt động thì ta sẽ thiết kế trang index.js để xem coi nó có hoạt động không.

Bây giờ chúng ta sẽ xử lý phần template cho project.Thì như tìm hiểu có rất nhiều view engine hỗ trợ cho việc thiết kế template cho Express project và hbs lẫn ejs đang là 2 view engine được sử dụng phổ biến trong cộng đồng nodejs.Thì như ban đầu khi khởi tạo project thì mình đã set sẵn view engine cho project là hbs.Về phía hbs thì cũng khá giống với html mà chúng ta đã biết tới chỉ khác một vài chỗ về cú pháp.Hoặc nếu bạn không sử dụng view engine là hbs thì bạn có thể vào file app,js và chỉnh sửa đoạn này theo ý của mình:

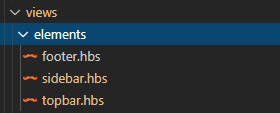


Ở đây thì mình thiết kế giao diện admin nên sẽ lấy một vài template đã có sẵn trên mạng,có thể truy cập trang dưới đây để tham khảo nhé.

Đây là đường link: **https://startbootstrap.com**

Sau khi đã setup xong templateSau khi đã setup xong template thì hãy nhớ copy các thư mục như css,stylesheet,img,bootstrap của template đã download vào folder public của project để khi chạy lên project có thể nhận được các file đó.

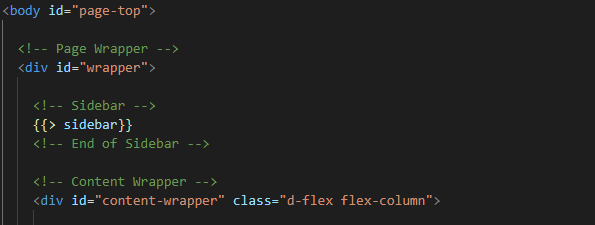
Và một điều quan trọng khi thiết kế giao diện đó chính là việc chúng ta sử dụng những layout có thể dùng chung như header,footer,sidebar,...Và hbs cũng tương tự như vậy trong việc tái sử dụng lại các layout chung.Đầu tiên ở folder views hãy tạo thêm một folder con tên là elements để chứa các layout chung.



Nhưng làm sau để có thể include được những file này thì ta phải bắt buộc cấu hình ở app.js

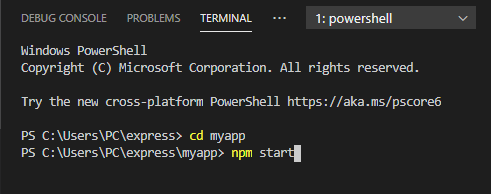


Dòng lệnh đây có nghĩa là chúng ta sẽ chỉ ra đường dẫn của các partials layout vừa tạo ở phía trên.Sau đó muôn sử dụng lại nó thì ta chỉ cần dùng cú pháp {{> tên file}}

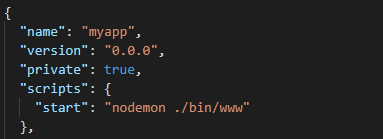


*Ví dụ một đoạn trong project để minh họa*

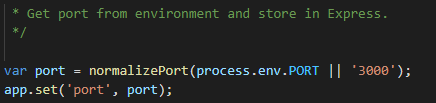
Làm tương tự như vậy với các layout khác,sau đó ta chạy thử project để xem giao diện trang index.hbs như thế nào.

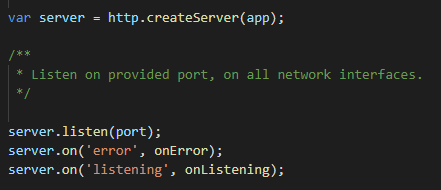


Giải thích về lệnh npm start,thì chúng ta hãy vào trang package.json



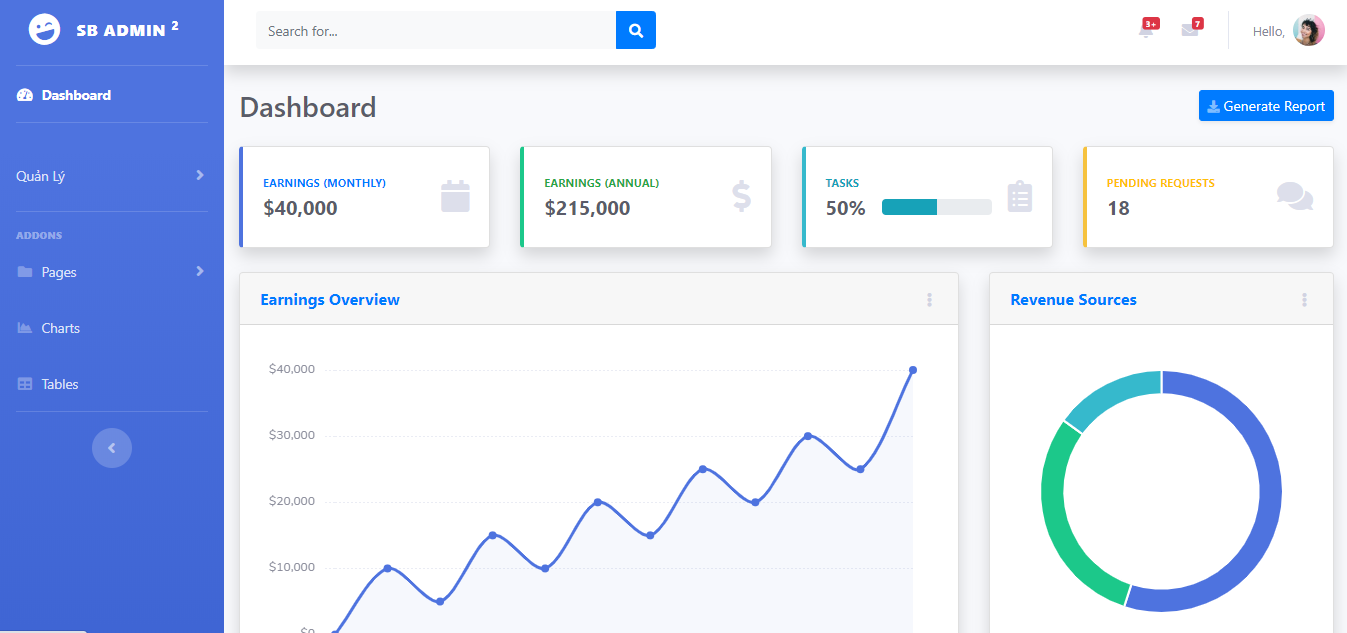
Khi dòng lệnh npm start được thực hiện thì project sẽ gọi vào file package.json,chú ý đến dòng được đánh mũi tên.Đoạn thông tin đó có nghĩa là start chương trình sẽ chạy file [www.js](http://www.js) trong folder bin mà chúng ta đã đề cập ở phía trên.Và trong file [www.js](http://www.js) chứa server,server lắng nghe và khởi tạo cho chúng ta.





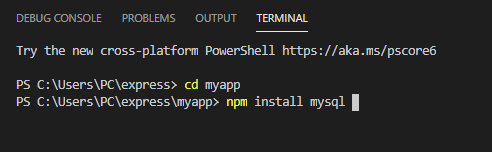


Sao khi server đã lắng nghe yêu cầu thì nó khởi tạo,tham số app có nghĩa là nó khởi tạo dựa vào các cài đặt trong file app,js, và dựa vào router setting trong file app.js mà nó chạy lên trang chủ admin như hình bên dưới:

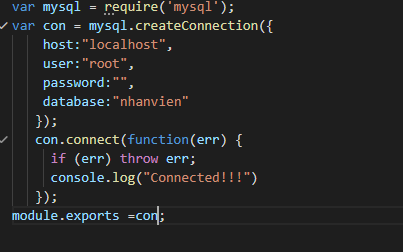


Như vậy là chúng ta đã giải quyết xong phần template cho project,tiếp đến chúng ta sẽ xử lý phần back-end cho project.

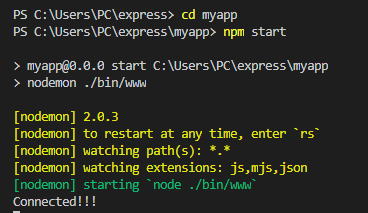
Điều đầu tiên là hãy tạo ra một csdl với mysql và ở đây thì mình dùng xampp để tạo csdl,sau khi đã hoàn tất csdl thì chúng ta cần một driver để kết nối project với CSDL.Vào lại folder myapp gõ lệnh như sau



Khi đã cài đặt xong thì chúng ta bắt đầu tạo một connection đến CSDL,tạo một thư mục dbConnect trong myapp.Trong thư mục dbConnect tạo ra 1 file db.js với nội dung như sau



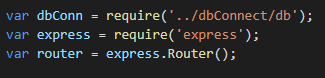
Với mysql thì các giá trị mặc đinh cho user và password giống với hình trên.Sau khi khởi tạo connect xong thì chúng ta viết một function để kiểm tra coi kết nối có thành công hay ko.Vì đây là một module nên muốn sử dụng lại thì chúng ta phải export nó ra.Chạy lại project để kiểm tra.

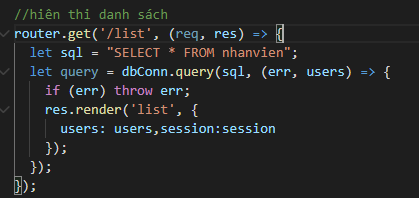


Như vậy là chúng ta đã thành công rồi!!!

Sau khi kết nối với CSDL thành công thì chúng ta bắt đầu viết CRUD áp dụng lên project này.Như đã đề cập ở phần trên mọi thao tác đều được route xử lý.Vào folder routes thì chúng ta thấy đã có sẵn 2 file index.js và users.js,ở đây chúng ta sẽ thực hiên trong file index.js.Nếu sau này càng thêm nhiều chức năng thì chúng ta chỉ cần khởi tạo ra các route tương ứng để cho việc quản lí code được dễ dàng.

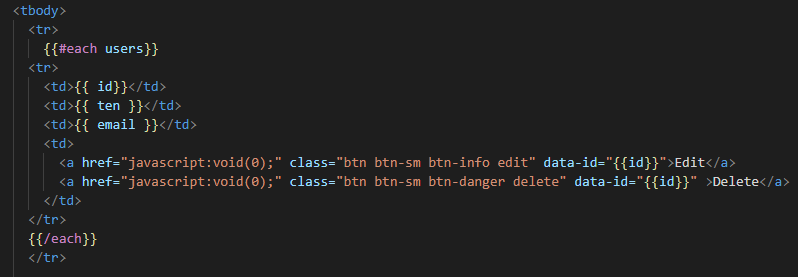
Đầu tiên là hiện thị trang danh sách các nhân viên trong dữ liêu,tương ứng với Read trong CRUD,về phần giao diện thì mình sẽ tạo 1 file list.hbs trong folder views để xổ ra dữ liệu.Tiếp đến ta viết phần xử lý để load dữ liệu lên trong file index.js với nội dung như sau:

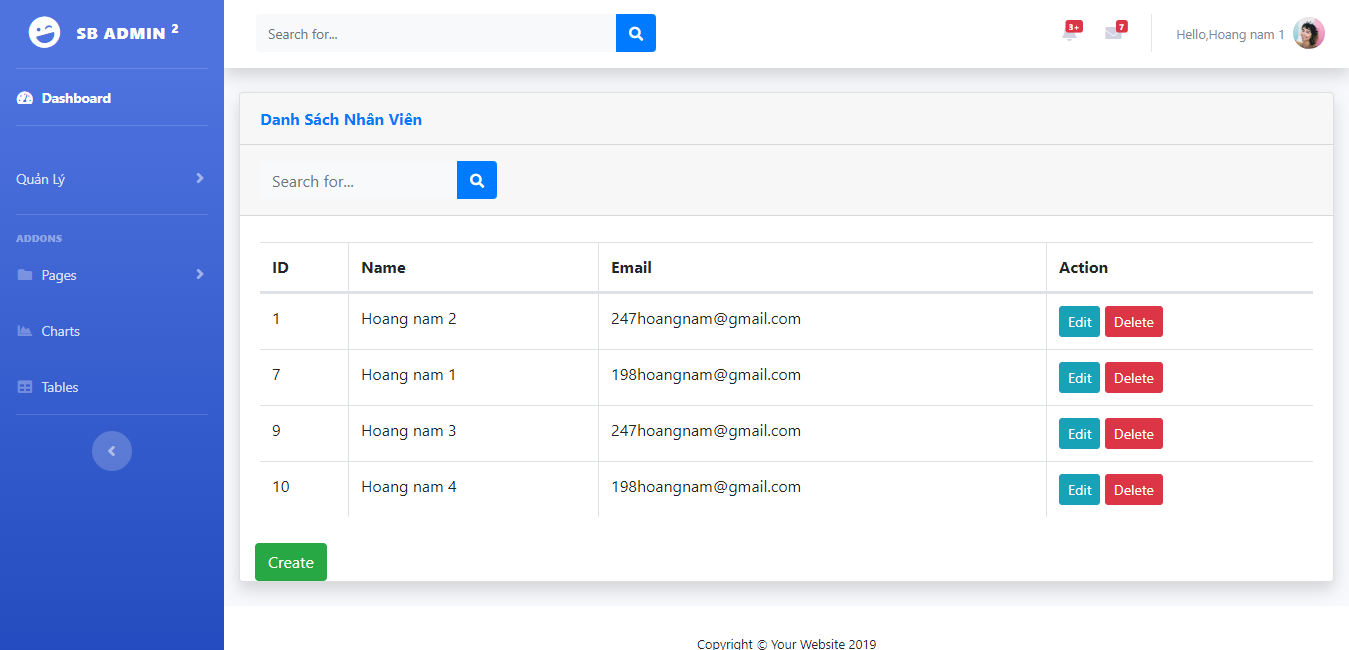




Cũng giống như các ngôn ngữ khác,khi thực thi với database thì chúng ta sử dụng lại connection trong file dbConnect lúc nãy.Tiếp tục,khi chúng ta nhấn vào đường dẫn “/list” router sẽ get giá trị đường dẫn,sau đó function có chức năng load dữ liệu từ database và cuối cùng response.render dữ liệu về trang list.hbs với tham số là một object users là kết quả của truy vấn.

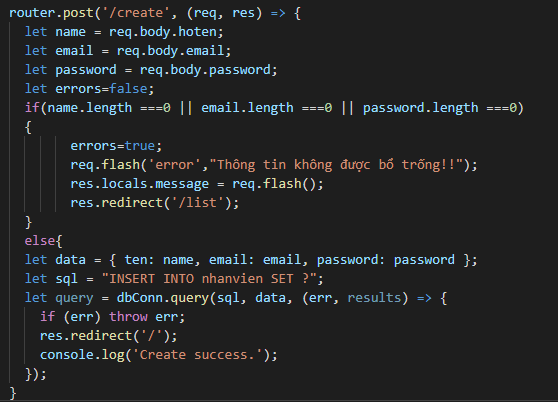
Sau khi được trả về dữ liệu từ server,trang list.hbs sẽ hiển thị thông tin cho phía client



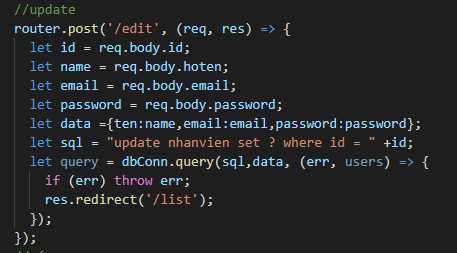


*Giao diện trang list.hbs*

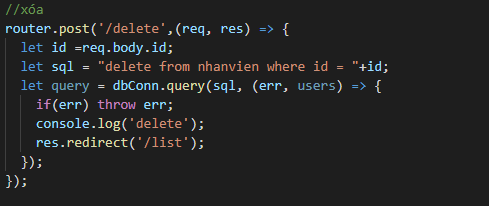
Tiếp tục là đến các chức năng như thêm sưa xóa,vẫn thực hiện như phía trên viết thêm các function cho từng yêu cầu.



*Chức năng thêm mới*

**

*Chức năng edit*

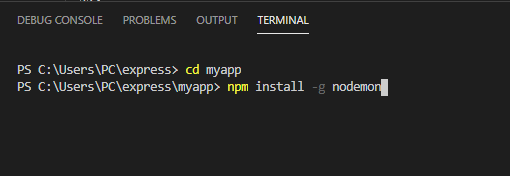


*Chức năng xóa*

Chúng ta sẽ tìm hiểu ý nghĩa,và cách hoạt động của các function trên ,khi các dữ được điền từ các form từ phía client theo user submit ,router sẽ nhân chỉ thị theo đường dẫn URL mà nó bắt được sau đó dữ liêu từ form client sẽ được gửi lên server và thực hiên truy vấn thay đổi trên database,và cuối cùng respone lại về phía client kết quả hiện thị.

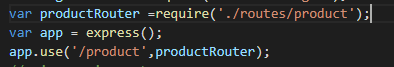
Về phía dữ liệu được lấy từ form như thế nào thì hãy chú ý đến cú pháp req.body,quay lại tìm hiểu về thuật ngữ body-parser,bodyParser trả về 1 function hoạt động như một middleware.Chức năng lắng nghe req,on(‘dâta’) và xây dựng req.body từ các đoạn giữ liệu mà nó nhận được.Nói tóm lại bodyParser phân tích dữ liệu một cách khác nhau tùy thuộc vào loại của nó.Vói việc sử dụng bodyParser bạn có thể lấy được data form từ req.body

Đến đây chúng ta đã hoàn thành một nửa rồi,nhưng trong quá trình làm thì mình lại rút ra được một điều đó là chính là một lần update một bản ghi hay thay đổi code thì chúng ta thường phải restart lại server mới có thể thay đổi,điều này làm cho ta mất nhiều thời gian.Và rồi mình có tìm hiểu được Nodemon,một thư viện hỗ trợ cho chúng ta có thể thay đổi những bản ghi mà không phải ngắt server.

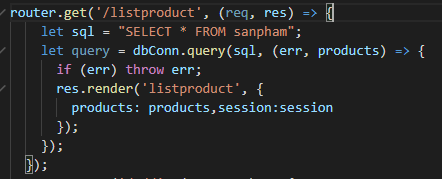


Vậy là đã xong một phần,bây giờ khi project càng ngày lớn hơn chúng ta làm nhiều chức năng hơn thì không thể làm tất cả trong một router được,vừa khó quản lí,sửa chữa lại không thẩm mỹ,vì thế ta phát sinh thêm nhiều router khác phục vụ cho những mục đích riêng biệt.Chẳng hạn trong project này mình muốn trang admin quản lý thêm sản phẩm thì phải tạo thêm 1 productRouter để thực hiên.

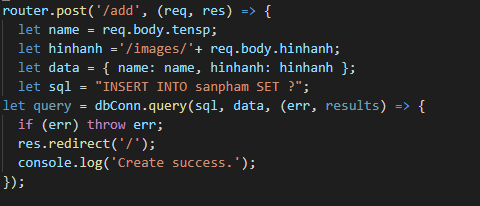
Đầu tiên vào folder routes tạo thêm 1 file là product.js,tiếp đến để có thể khai báo nó cho server có thể nhận biết thì ta lại vào file app.js và khởi tạo với nội dung như sau:



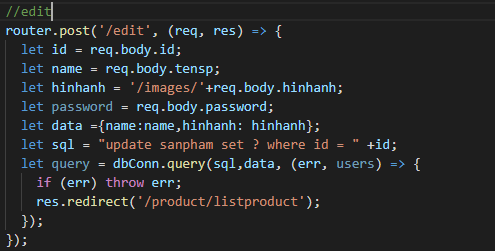
Sau khi xong thì chúng ta làm các chức năng như trong router index.js



*Hiển thị danh sách*

**

*Thêm mới sản phẩm*



*Chỉnh sửa thông tin*

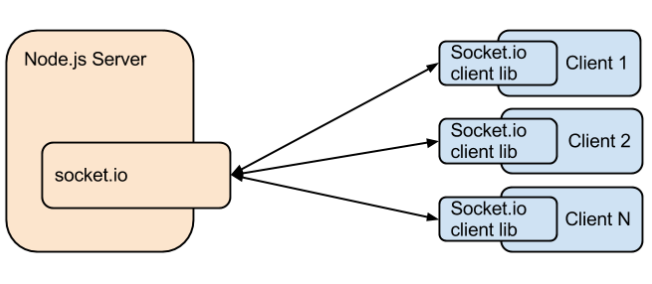
Ngoài ra trong quá trình làm,mình cũng có tìm hiểu một số module chức năng hỗ trợ cho việc thiết kế Express application chẳng hạn như Express-session cho phép tạo ra một session tồn tại khi đăng nhập vào hệ thống,như Express-flash với chức năng hiện thị thông báo từ server đến client,module Multer cho phép UploadFile từ máy tính của mình lên server hoặc học được truy vấn MySQL trong JavaScript như thế nào.

Sau khi hoàn thành một Web server với framework Express,bây giờ mình sẽ kết họp một framwork khác là Socket.io để tạo một chat room với thời gian thực để các user có thể chat với nhau mỗi khi login vào.Trước hết phải tìm hiểu cơ chế hoạt động của Socket.io,cùng nhìn lại về websocket.

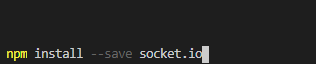
* Trong trình duyệt mà được hỗ trợ WebSockets protocol, một kết nối giữa server và browser được tạo ra quanh HTTP và được gọi là một “HTTP hanshake” (cái bắt tay). Một kết nối được tạo ra trình duyệt và server mở ra một cổng giao tiếp liên tục thông qua một TCP socket. Nó sẽ hỗ trợ cho cả việc gửi và truy vấn message trên một cổng kết nối. Điều này giúp server load ít hơn, giảm số message bị trễ, “and unify PUSH communication using standalone connection”.
* Tuy nhiên WebSocket mắc phải vấn đề đó là HTTP proxies, firewall và hosting provider. Khi Websocket sử dụng một phương thức giao tiếp ngoài HTTP, một phần nhiều trong số đó chưa được hỗ trợ và block bất cứ kết nối socket nào. Vấn đề này chỉ được giải quyết khi sử dụng thư viện trừu tượng mà có thể dễ dàng thay đổi giữa các giao thức dựa trên resoures có sẵn.
* [Socket.io](http://socket.io/) được xây dựng để giải quyết vấn đề này và nó luôn sẵn sàng được sử dụng cho NodeJS developer.

**Socket.io**

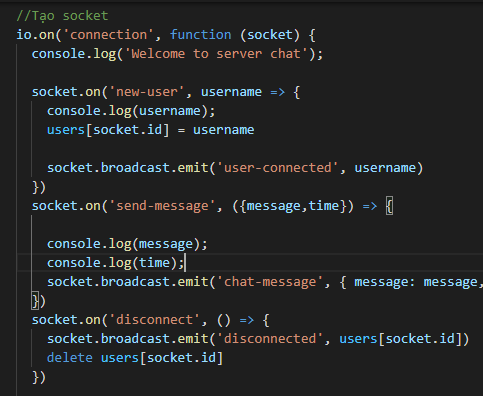
* Là một module của NodeJs
* Được xây dựng nhằm mục đích tạo ra real time NodeJS application. [Socket.io](http://socket.io/) cung cấp cho lập trình viên các đặc trưng như event, room và tự động phục hồi lại kết nối.
* Khi chúng ta include [Socket.io](http://socket.io/) module vào trong ứng dụng của mình nó sẽ cung cấp cho chúng ta hai object đó là: socket server quản lý functionality phía server và socket client điều khiển funtionality phía client.
* Khi client muốn kết nối tới [Socket.io](http://socket.io/) server, nó sẽ gửi cho server một “handshake HTTP request”. Server sẽ phân tích request đó với những thông tin cần thiết trong suốt quá trình kết nối. Nó sẽ tìm cấu hình của middleware mà đã được đăng ký với server và thực thi chúng trước khi đưa ra sự kiện kết nối. Khi kết nối thành công thì connection event listener được thực thi, tạo ra một instance mới của socket có thể coi như định danh của client mà mỗi một client kết nối tới sẽ có 1 định danh. Các bạn có thể thấy rõ khi xem hình dưới đây



Sau khi đã hiểu về cơ chế hoạt động của socket.io,bây giờ mình sẽ kết hợp vào project và viết một room chat cho các user.Đầu tiên install framework Socket.io



Vì Socket.io chạy trên cả 2 phia client lẫn server nên mình phải cấu hình cho cả 2 file này.Vào file [www.js](http://www.js) trong thư mục bin là nơi khởi tạo server thêm nội dung như hình sau:

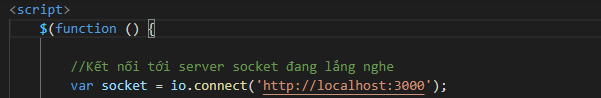


*Khởi tạo socket cho phía server*

Một số thao tác chính bên server như trên hình:

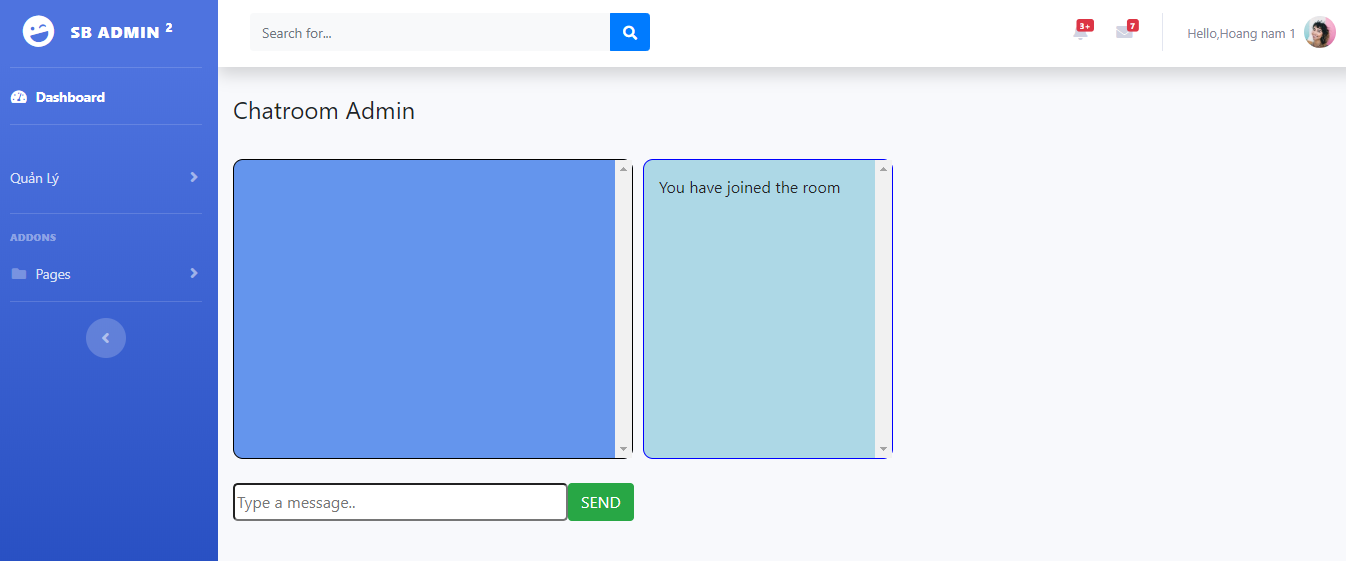
* “new user" : Khi phía client phát lên sự kiện "new user" kèm theo dữ liệu mà nó gửi lên thì phía server mình sẽ quản lý và lưu thông tin tin của người dùng này vào object **users** đã được khai báo phía server.
* "send message": Khi một message được submit, phía client sẽ emit (phát) ra sự kiện "send message" và kèm theo nội dung của tin nhắn và người nhận tin nhắn. Server mình sẽ nhận lấy gửi về cho người gửi request và người được gửi message sự kiện "new message" gồm nội dung của message. Client chỉ việc "on" bắt lấy sự kiện này, nhận lấy thông tin message và hiển thị.
* "disconnect": Khi client ngắt kết nối với server là lúc mà bạn tắt trình duyệt kết nối với server đi. Server sẽ nhận thông tin user ngắt kết nối và gửi lại cho toàn bộ client đang được kết nối

Tiếp đến là khởi tạo cho phía client một socket tương tự để có thể xử lý giao thức giũa 2 bên.Ở đây mình sẽ khởi tạo trong user UI với đoạn script như sau:



Như vậy là cả 2 bên đều đã có socket.Tiếp đến làm sao để có thể hoạt động được thì bạn phải nắm rõ 2 phương thức sau một là Socket.emit và Socket.on

* Socket.emit:Lấy dữ liệu được gửi từ server hoặc có thể từ client,dữ liệu được gửi sẽ được truyền đi đến mọi client khi kết nối đến server.
* Socket.on:Khởi tạo mội socket với chức năng xử lý chẳng hạn như khi có một user kết nối,send-message,hoặc là disconnect.

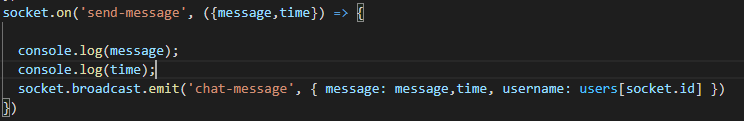


Đây là giao diện trong chatroom giữa các user,như bạn đã thấy khi một user vạo hệ thống chat thì lập tức có thông báo là You have join the room,chính là giờ socket xử lý ở client và báo lại cho server biết.



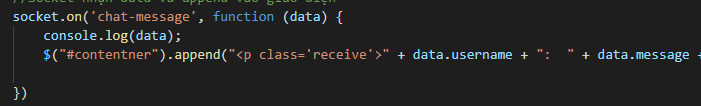
*Ví dụ khi client gửi dữ liệu*

Nhìn vào đoạn code phía trên ta sẽ thấy socket hoạt động như thế nào,khi message và username được gửi đi socket phía client sẽ tạo một emit với nội dung là ‘send-message’ với dữ liệu lên phía server.Sau khi được gửi làm thế nào server nhận được thì như sau:



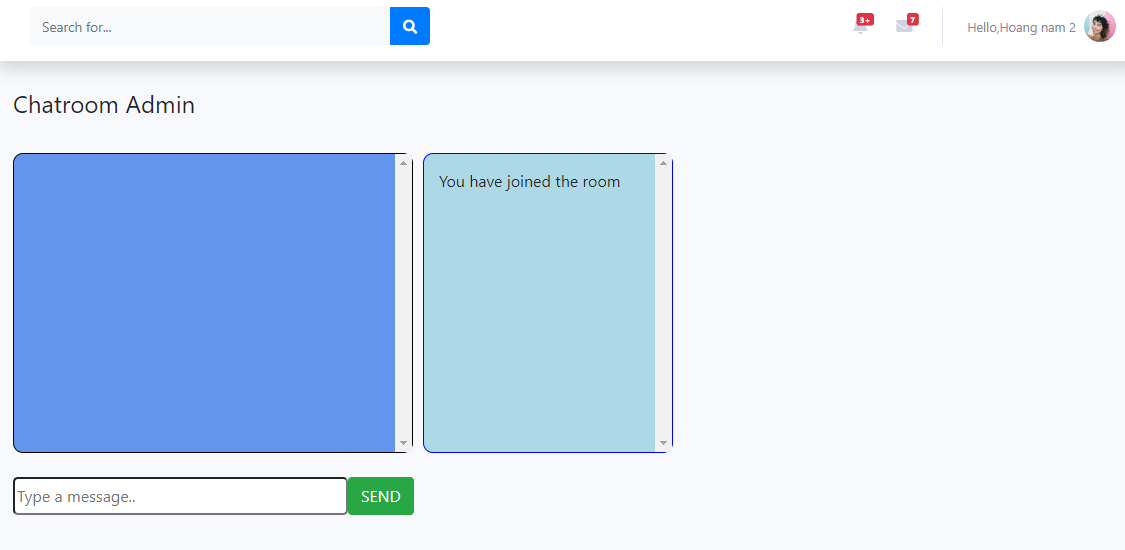
*Bên phía server*

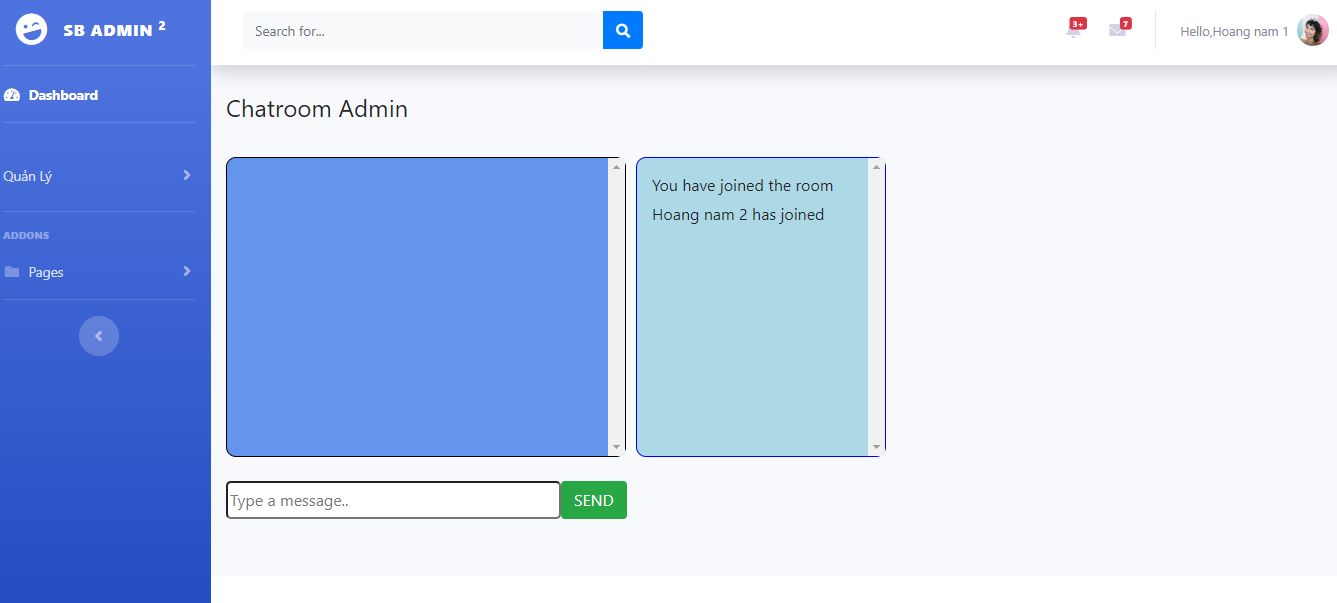
Server sẽ kết nối đến socket là lấy dữ liệu mà client đã emit lên ,sau khi lấy được server sẽ tạo một function và emit lại cho tất cả các client đang kết nối đoạn nội dụng mà user kia đã gửi.



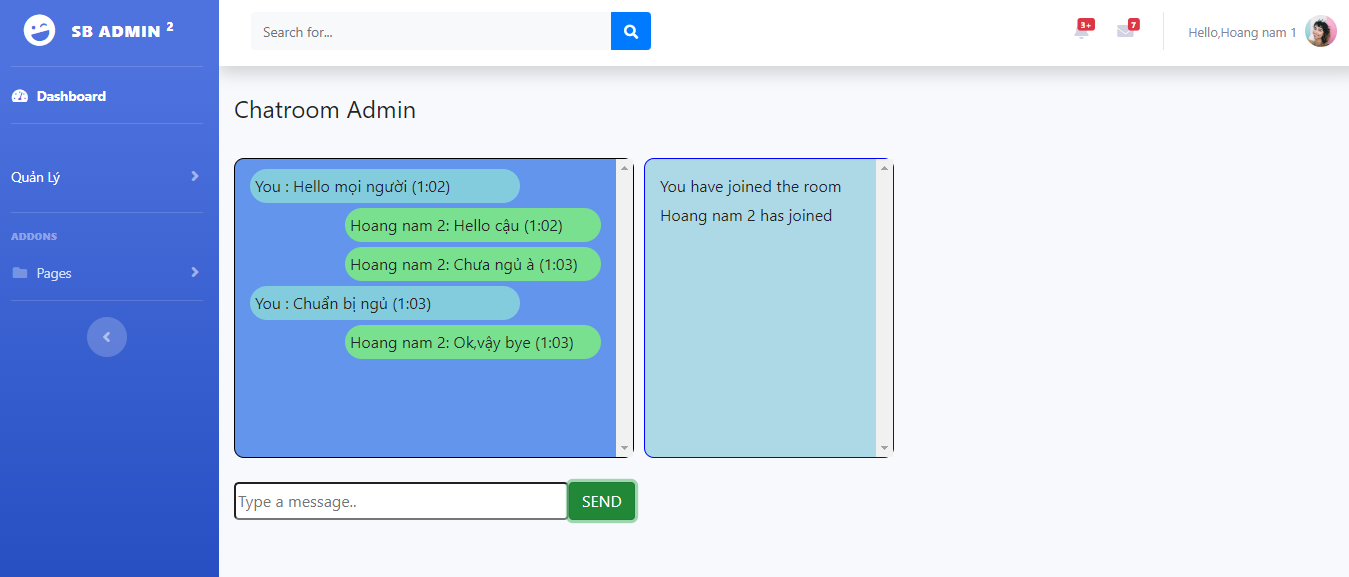
Tât cả client đều nhân được thông qua funtion phía trên và hiện thị dữ liêu từ server lên cho user.Đó chính là cách hoạt động giữa server và các client kết nối thông qua giao thức của socket.Tương tự như các hành động như connected hoặc connected.

Demo bằng hình ảnh.Chúng ta thể vào 1 browser khác và login với tài khoản khác như hình sau





*Thì bên trình duyệt này đã hiện thị thông báo 1 user đã join vào*



*2 user giao tiếp qua lại trong thời gian thực*

Như vây là mình đã hoàn thành xong khi kết hợp giữa 2 Framework được sử dụng rộng rãi trong Nodejs để tạo 1 demo hoàn chỉnh.Qua demo lần này bản thân mình đã học hỏi rất nhiều,để tạo ra được 1 project như thế nào,và hiểu rõ hơn về bản chât của Nodejs và cách mà nó hoạt động như thế nào.

Link source code cho bạn nào muốn tham khảo: https://github.com/Hoangphuongnam123/Final-project.git

# Tài liệu kham khảo

1. “Express FW nodejs”, <https://vietjack.com/nodejs/express_framework_trong_nodejs.jsp>
2. “Project NodeJS” , <https://github.com/nodejs/node>
3. “NodeJS Documentation” , <https://nodejs.org/en/download/>
4. “Package Manager for Window”, <https://chocolatey.org/>
5. “Package Manager for MacOS”, <https://brew.sh/>
6. “Node Package Manager”, <https://www.npmjs.com/>
7. “Node Package Manager”, <https://yarnpkg.com/>
8. “Protect module brain”, <https://github.com/BrainJS/brain.js>
9. “Module neural network”, <https://www.npmjs.com/package/neuralnetwork>
10. “Module Dnn”, <https://github.com/junku901/dnn>
11. “Module Sysnatic”, <https://github.com/cazala/synaptic>
12. “Module node neural network”, <https://www.npmjs.com/package/node-neural-network>
13. “Module mind”, <https://github.com/stevenmiller888/mind>
14. “Module converjs”, <https://github.com/karpathy/convnetjs>
15. “Module dn2a”, <https://github.com/antoniodeluca/dn2a>
16. “Project module node opencv”, <https://github.com/peterbraden/node-opencv>
17. “Module jimp”, <https://github.com/oliver-moran/jimp>
18. “Module neural Network Nodejs”, <https://github.com/BrainJS/brain.js>
19. “Nodejs Tutorial in w3schools”, https://w3schools.com