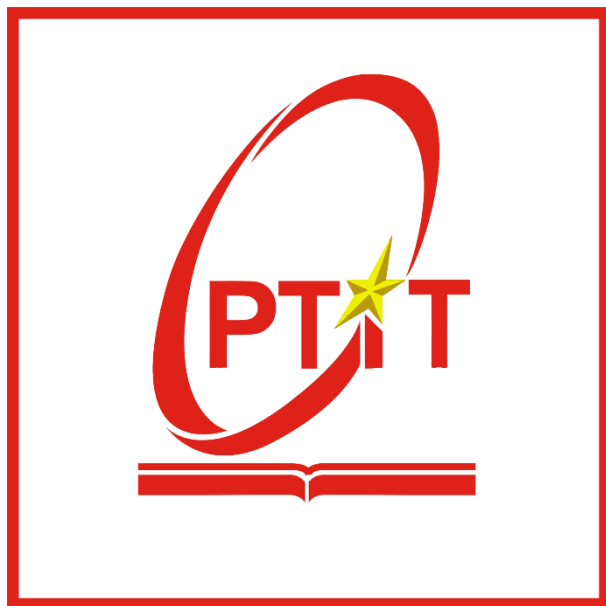


**BỘ THÔNG TIN VÀ TRUYỀN THÔNG  
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**



**BÁO CÁO THỰC TẬP CƠ SỞ TUẦN 1**

# **TÌM HIỂU VỀ NODE.JS VÀ EXPRESS FRAMEWORK**

**Giảng viên hướng dẫn:** TS. Kim Ngọc Bách

**Sinh viên thực hiện:**

Nguyễn Quang Trung – B22DCDT321

# MỤC LỤC

- I. Tổng quan về Node.js
  - I.A. Node.js là gì ?
  - I.B. Node REPL và CLI
  - I.C. Biến toàn cục (Global variables)
  - I.D. Modules
  - I.E. CommonJS và ECMAScript Modules
  - I.F. Các framework nổi tiếng và xem thêm
- II. Tìm hiểu về Express Framework
  - Khởi tạo 1 server Express
  - HTTP requests
  - Middleware
- III. Bài tập thực hành

# I. Tổng quan về Node.js

## I.A. Node.js là gì ?

→ Node.js là:

- Môi trường (runtime environment) để chạy Javascript phía server, cho phép thực thi Javascript bên ngoài trình duyệt.
- Được xây dựng trên Chrome's V8 Javascript Engine.

→ Node.js thường được dùng để ?

- Xây dựng API và Backend.
- Xây dựng Microservices.
- Xử lý file và hệ thống.
- Kết hợp với frontend như React, Angular, Vue để xây dựng ứng dụng Full-Stack (MERN Stack).

→ Package Manager (npm):

Node.js đi kèm với npm (Node Package Manager), một công cụ quản lý các thư viện và gói mã nguồn mở. Npm là kho lưu trữ lớn nhất thế giới cho các gói mã nguồn mở, giúp các nhà phát triển dễ dàng chia sẻ và sử dụng lại mã nguồn.

## I.B. Node REPL và CLI

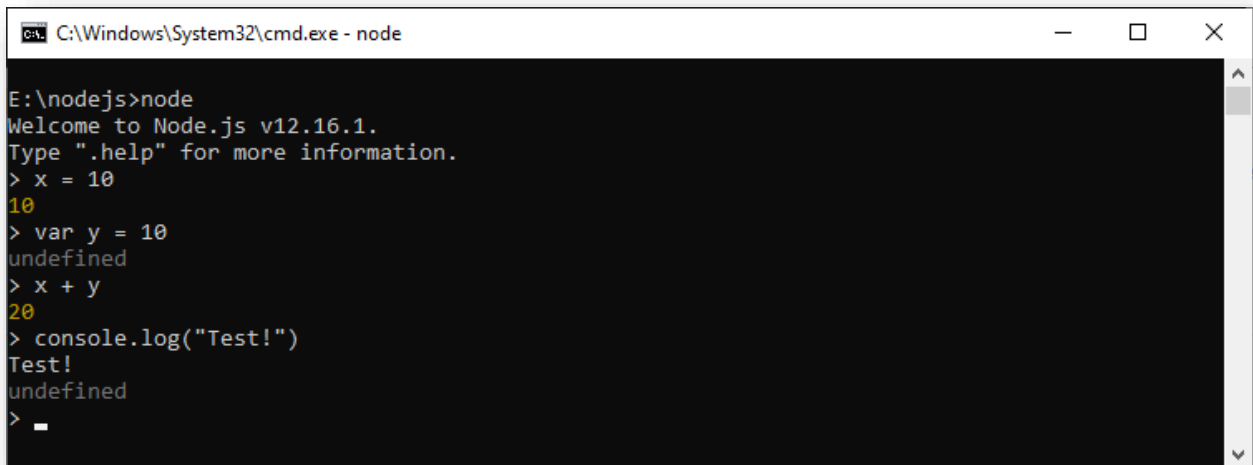
- Node REPL: Read – Eval – Print – Loop. Là một môi trường tương tác của Node.js, cho phép bạn nhập lệnh Javascript trực tiếp và nhận kết quả ngay lập tức. Nó hoạt động theo vòng lặp:

Read: Đọc đầu vào từ người dùng.

Eval: Thực thi mã Javascript.

Print: Hiển thị kết quả của mã đã thực thi.

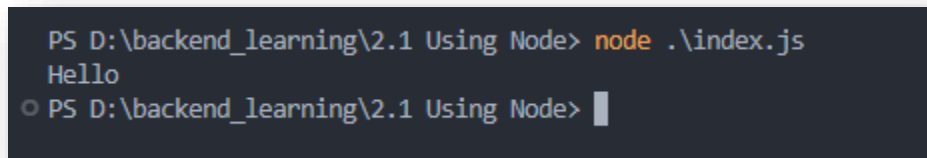
Loop: Quay lại bước đầu tiên và tiếp tục nhận lệnh mới.



```
C:\Windows\System32\cmd.exe - node
E:\nodejs>node
Welcome to Node.js v12.16.1.
Type ".help" for more information.
> x = 10
10
> var y = 10
undefined
> x + y
20
> console.log("Test!")
Test!
undefined
> _
```

*Ví dụ về Node REPL*

- Node CLI (Command Line Interface): của Node.js là một giao diện dòng lệnh cho phép bạn chạy các tập tin JavaScript hoặc thực thi các lệnh liên quan đến Node.js.



```
PS D:\backend_learning\2.1 Using Node> node .\index.js
Hello
PS D:\backend_learning\2.1 Using Node> █
```

*Ví dụ về Node CLI*

### **I.C. Biến toàn cục (Global variables)**

- Trong Node.js, biến toàn cục (global variables) là những biến có thể được truy cập từ bất kỳ đâu trong chương trình mà không cần import hoặc khai báo lại.

Biến/Đối tượng	Mô tả
global	Đối tượng toàn cục, tương tự `window` trong trình duyệt
__dirname	Trả về đường dẫn thư mục chứa file hiện tại
__filename	Trả về đường dẫn đầy đủ của file hiện tại
process	Đối tượng chứa thông tin về tiến trình Node.js
module	Đối tượng chứa thông tin về module hiện tại
exports	Được sử dụng để xuất module trong Node.js
setTimeout	Thực thi một hàm sau một khoảng thời gian
setInterval	Thực thi một hàm lặp đi lặp lại theo thời gian định kỳ
console	Đối tượng để in ra màn hình console

## I.D. Modules

- Trong Node.js, module là các file JavaScript độc lập chứa các đoạn mã có thể tái sử dụng. Node.js sử dụng hệ thống module để tổ chức và quản lý code dễ dàng hơn.

- Khi một script lớn đến một mức độ nào đó, chúng ta có thể tổng hợp các chức năng hay dùng thành các module.

- Module là một đoạn mã nguồn mà có thể tái sử dụng được.

- Tồn tại độc lập, không làm ảnh hưởng, xung đột với các đoạn mã nguồn khác.

- Trong JavaScript thì không có khái niệm này (trước đây).

Loại Module	Mô tả
Core Modules	Các module tích hợp sẵn trong Node.js như fs, http, path, os, v.v.
Local Modules	Các module do người dùng tự tạo để sử dụng trong dự án.
Third-party Modules	Các module được cài đặt từ npm (Node Package Manager), ví dụ: express, lodash, mongoose, v.v.

## I.E. CommonJS và ECMAScript Modules

### 1. CommonJS:

CommonJS (CJS) là một hệ thống module được sử dụng trong Node.js để quản lý và tổ chức mã nguồn. Đây là chuẩn module mặc định trong Node.js trước khi ES Modules (ESM) được hỗ trợ.

- Đặc điểm của CommonJS:

- Đồng bộ: Các module được tải ngay lập tức.
- Chạy trên môi trường Node.js: Không thể sử dụng trực tiếp trong trình duyệt (trừ khi dùng Webpack, Browserify, v.v.).
- Sử dụng `require()` để import module.
- Dùng `module.exports` hoặc `exports` để export module.

### 2. ECMAScript:

ECMAScript (ES) là một tiêu chuẩn dành cho các ngôn ngữ kịch bản (scripting languages), bao gồm JavaScript, JScript và ActionScript. Nó được biết đến nhiều nhất như một tiêu chuẩn của JavaScript, nhằm đảm bảo tính

tương thích của các trang web trên các trình duyệt khác nhau. Tiêu chuẩn này được Ecma International quy định trong tài liệu ECMA-262.

ECMAScript thường được sử dụng cho lập trình phía client trên World Wide Web, và ngày càng được áp dụng cho các ứng dụng phía server thông qua các môi trường chạy như Node.js, Deno và Bun.

Đặc điểm	CommonJS (CJS)	ES Modules (ESM)
Cách import	require()	import
Cách export	module.exports	export
Tính đồng bộ	Đồng bộ	Bất đồng bộ
Hỗ trợ trình duyệt	Không	Có (hỗ trợ trực tiếp)
Môi trường chính	Node.js	Trình duyệt & Node.js

### 3. Ưu và nhược điểm của CommonJS và ES Modules

#### - CommonJS (CJS):

##### Ưu điểm:

- Đơn giản, dễ sử dụng, và là chuẩn mặc định của Node.js trong nhiều năm.
- Hỗ trợ đồng bộ, phù hợp với các tác vụ I/O đồng bộ như đọc/ghi file.

##### Nhược điểm:

- Không hỗ trợ tree-shaking (loại bỏ mã không sử dụng), dẫn đến kích thước file bundle lớn hơn.
- Không tương thích trực tiếp với trình duyệt, cần sử dụng các công cụ như Webpack hoặc Browserify để chuyển đổi.

- ES Modules (ESM):

Ưu điểm:

- Hỗ trợ tree-shaking, giúp giảm kích thước file bundle bằng cách loại bỏ mã không sử dụng.
- Là chuẩn JavaScript hiện đại, được hỗ trợ trực tiếp trong trình duyệt và Node.js (từ phiên bản Node.js 12 trở lên).
- Hỗ trợ bất đồng bộ, phù hợp với các tác vụ I/O bất đồng bộ.

Nhược điểm:

- Cần cấu hình thêm khi sử dụng trong Node.js (ví dụ: đổi đuôi file từ .js sang .mjs hoặc cấu hình package.json).
- Một số thư viện cũ có thể chưa hỗ trợ ES Modules.

## **I.F. Các framework nổi tiếng và xem thêm**

- Express.js được sử dụng phổ biến để phát triển ứng dụng trên nền Nodejs.
- Electron.js được sử dụng để phát triển ứng dụng được sử dụng trên môi trường desktop.
- NPM - bộ quản lý đóng gói nổi bật của Node.js. Từ phiên bản Node.js 0.6.3, npm được cài tự động với Node.js.
- YARN - Bộ quản lý đóng gói mã nguồn mở với hiệu năng cao.
- JSAN, viết tắt của JavaScript Archive Network - một bộ quản lý gói khác ít dùng hơn.
- Opa, một hướng tiếp cận khác cho lập trình ứng dụng web, có nhiều đặc trưng của Node.js






## II. Tìm hiểu về Express Framework

Express là một framework được xây dựng trên nền tảng của Nodejs. Nó cung cấp các tính năng mạnh mẽ để phát triển web hoặc mobile. Express hỗ trợ các method HTTP và middleware tạo ra API vô cùng mạnh mẽ và dễ sử dụng.

Với Express.js, bạn có thể xây dựng các API RESTful dễ dàng hơn và quản lý dữ liệu hiệu quả mà không cần viết quá nhiều code phức tạp. Express.js giúp bạn tập trung vào logic ứng dụng thay vì xử lý chi tiết các yêu cầu HTTP.

Các tính năng nổi bật của Express:

- Thiết lập router cho phép sử dụng với các hành động khác nhau dựa trên phương thức HTTP và URL.
- Hỗ trợ xây dựng theo mô hình MVC
- Cho phép định nghĩa middleware giúp tổ chức và tái sử dụng code
- Hỗ trợ RESTful API

1 <code>const express = require('express' 4.16.2 )</code> 2 <code>const app = express()</code>		1- Hiring the manager
3 4 <code>app.use(function (req, res, next) {</code> 5 <code>  console.log('Request: ', req)</code> 6 <code>  console.log('Response: ', res)</code> 7 <code>  next()</code> 8 <code>})</code>		2-Got shirt and shoes?
9 10 <code>app.get('/', function (req, res) {</code> 11 <code>  res.send('Hello World!')</code> 12 <code>})</code>		3-Taking an order
13 14 <code>app.listen(3000, function () {</code> 15 <code>  console.log('Example app listening on port 3000!')</code> 16 <code>})</code>		4-Open for business

*Ví dụ về sử dụng Express Framework*

Khởi tạo 1 server Express:

1. Khởi tạo thư mục (Create directory)
2. Khởi tạo file index.js (Create index.js file)
3. Khởi tạo npm (Initialize npm)
4. Cài đặt gói Express (Install the Express package)
5. Viết ứng dụng server trong file index.js (Write server application in index.js)
6. Bắt đầu server (Start server)

- HTTP Requests:

GET → Yêu cầu tài nguyên

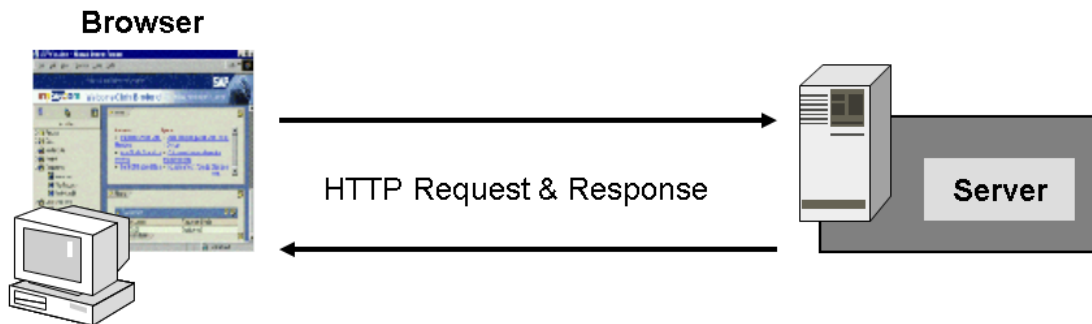
POST → Gửi đi tài nguyên

PUT → Thay thế tài nguyên (thay thế hoàn toàn)

PATCH → Thay thế tài nguyên 1 phần (Patch up a resource)

DELETE → Xóa tài nguyên

- Making requests:



- Express Middleware:

Các hàm middleware là những khối xây dựng quan trọng của bất kỳ máy chủ web nào, đặc biệt trong các framework như ExpressJS. Chúng đóng vai trò quan trọng trong chu trình xử lý yêu cầu - phản hồi. Middleware là các hàm có quyền

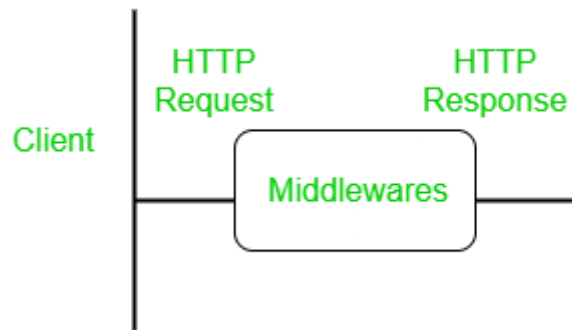
truy cập vào đối tượng yêu cầu (req), đối tượng phản hồi (res), và hàm next trong chu trình xử lý yêu cầu - phản hồi của ứng dụng.

Middleware trong Express đề cập đến các hàm xử lý yêu cầu trước khi đến trình xử lý tuyến (route handlers). Các hàm này có thể:

- Thay đổi đối tượng yêu cầu (req) và phản hồi (res).
- Kết thúc chu trình yêu cầu - phản hồi.
- Gọi hàm middleware tiếp theo trong chuỗi xử lý.

Đặc điểm của Middleware:

- Được thực thi theo thứ tự định nghĩa.
- Có thể thực hiện các tác vụ như xác thực, ghi log, hoặc xử lý lỗi.
- Giúp tách biệt các chức năng và quản lý các tuyến đường phức tạp hiệu quả hơn.



### Cách Middleware Hoạt Động Trong Express.js

Trong Express.js, các hàm middleware được thực thi tuần tự theo thứ tự chúng được thêm vào ứng dụng. Khi một yêu cầu được gửi đến máy chủ, nó sẽ đi qua từng middleware theo thứ tự định nghĩa. Mỗi middleware có thể thực hiện một nhiệm vụ và:

- Gửi phản hồi để kết thúc chu trình request-response.
- Gọi next() để chuyển quyền xử lý cho middleware tiếp theo.

Luồng xử lý của Middleware:

- Yêu cầu (request) đến máy chủ.
- Các middleware được áp dụng theo thứ tự.

- Mỗi middleware có thể:
- Gửi phản hồi và kết thúc chu trình.
- Gọi next() để tiếp tục xử lý với middleware tiếp theo.
- Nếu không có middleware nào kết thúc chu trình, yêu cầu sẽ đi đến route handler và phản hồi cuối cùng sẽ được gửi đi.

### Các Loại Middleware Trong Express.js

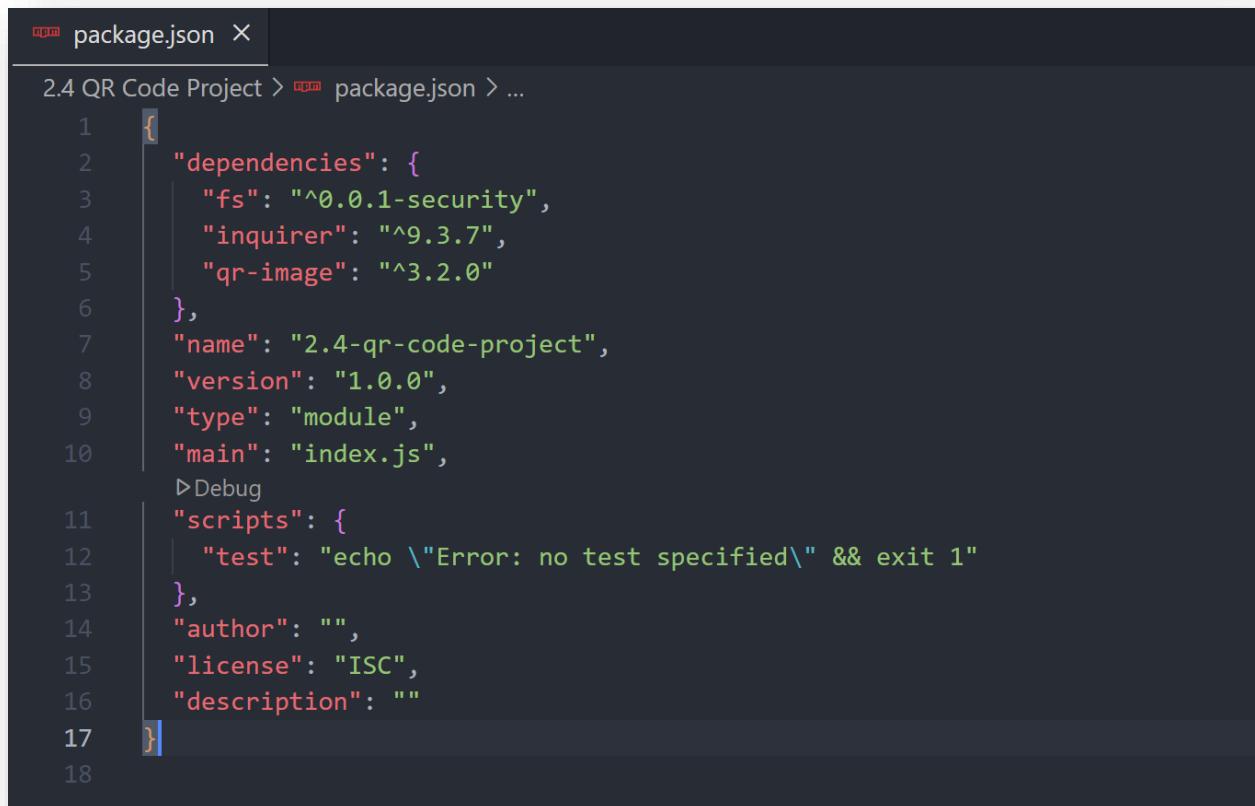
- Application-level middleware
  - Gắn với toàn bộ ứng dụng bằng app.use() hoặc app.METHOD().
  - Được thực thi cho tất cả các tuyến đường.
- Router-level middleware
  - Gắn với các tuyến đường cụ thể bằng router.use() hoặc router.METHOD().
  - Chỉ thực thi trên các tuyến đường được định nghĩa trong router đó.
- Error-handling middleware
  - Dùng để xử lý lỗi trong chu trình request-response.
  - Được định nghĩa với bốn tham số (err, req, res, next).
- Built-in middleware (Middleware có sẵn trong Express)
  - Được Express cung cấp sẵn, không cần cài đặt thêm.
- Third-party middleware (Middleware bên thứ ba)
  - Được phát triển bởi cộng đồng, cần cài đặt thêm bằng npm.

### III. Bài tập Thực hành

1. Lấy input (url) nhập từ bàn phím, biến đổi thành qr code và lưu trữ vào file.

- Sử dụng gói npm inquirer để lấy đầu vào từ người dùng.
- Sử dụng gói npm qr-image để chuyển đổi URL mà người dùng nhập vào thành hình ảnh mã QR.
- Tạo một tệp .txt để lưu đầu vào của người dùng bằng mô-đun fs (hệ thống tệp) của Node.js.

→ File package.json sau khi cài đặt npm



```
package.json X
2.4 QR Code Project > package.json > ...
1 {
2   "dependencies": {
3     "fs": "^0.0.1-security",
4     "inquirer": "^9.3.7",
5     "qr-image": "^3.2.0"
6   },
7   "name": "2.4-qr-code-project",
8   "version": "1.0.0",
9   "type": "module",
10  "main": "index.js",
11  "scripts": {
12    "test": "echo \"Error: no test specified\" && exit 1"
13  },
14  "author": "",
15  "license": "ISC",
16  "description": ""
17 }
18
```

→ File index.js lấy input từ người dùng và convert thành qr code

```
JS index.js x
2.4 QR Code Project > JS index.js > ...
7   import fs from "fs";
8   import inquirer from 'inquirer';
9   import qr from "qr-image";
10
11   inquirer
12     .prompt([
13       {
14         message: "Type in your URL:",
15         name: "URL",
16       },
17     ])
18     .then((answers) => {
19       const url = answers.URL;
20       var qr_svg = qr.image(url);
21       qr_svg.pipe(fs.createWriteStream('i_love_qr.png'));
22       fs.writeFile('messageURL.txt', url, (err) => {
23         if (err) throw err;
24         console.log('The file has been saved!');
25       });
26       console.log("OK done")
27     })
28     .catch((error) => {
29       if (error.isTtyError) {
30         // Prompt couldn't be rendered in the current environment
31       } else {
32         // Something else went wrong
33       }
34     });
35
36   // let userInput = prompt()
37
```

Khai báo các npm package sử dụng

prompt để lấy và lưu trữ input từ bàn phím

tạo qr code và dùng module fs để lưu output vào file

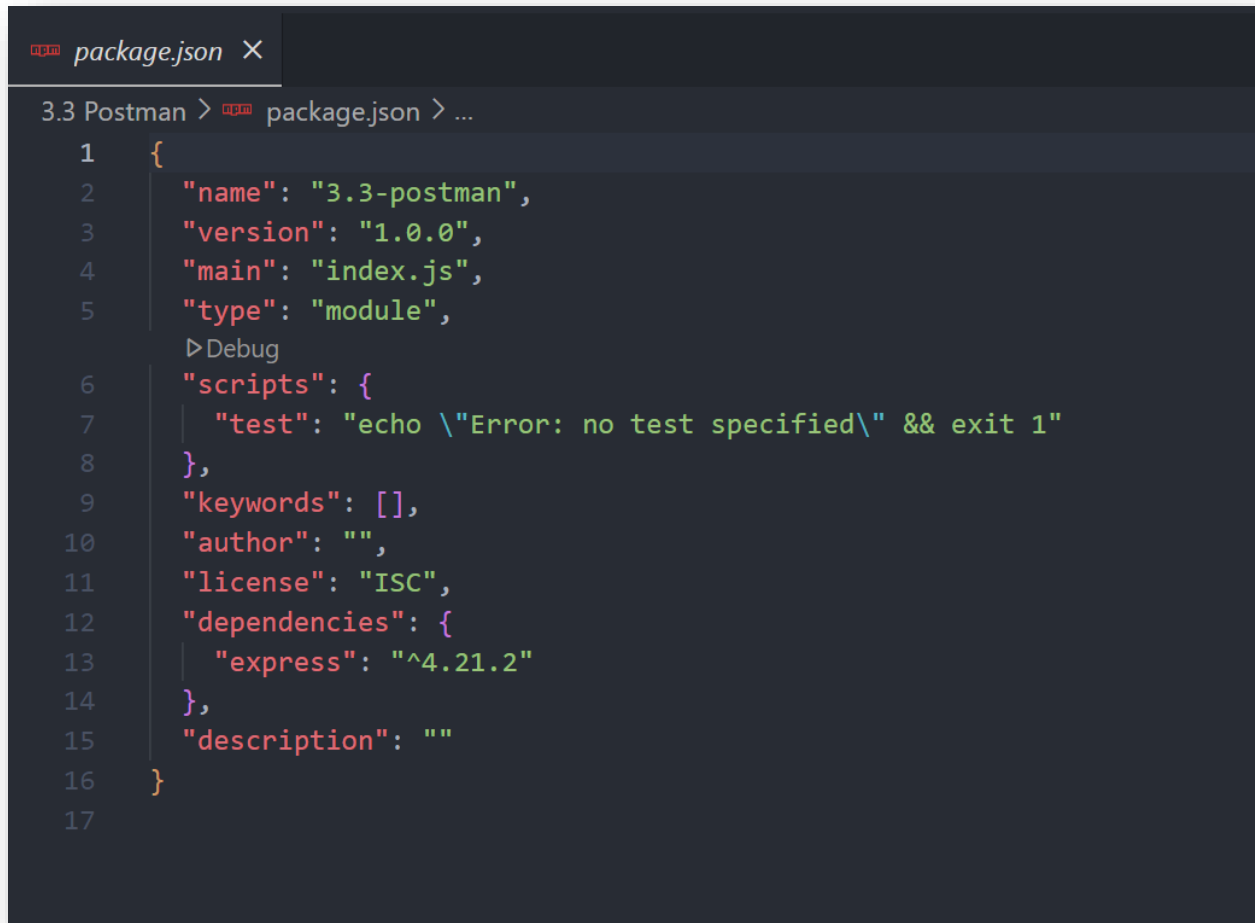
xử lý nếu có lỗi

→ Kết quả



## 2. Sử dụng Postman kiểm tra status code của server

→ File package.json sau khi cài đặt express



```
package.json X
3.3 Postman > package.json > ...
1  {
2    "name": "3.3-postman",
3    "version": "1.0.0",
4    "main": "index.js",
5    "type": "module",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC",
12   "dependencies": {
13     "express": "^4.21.2"
14   },
15   "description": ""
16 }
17
```

→ File index

```
JS index.js ×
3.1 Express Server > JS index.js > ...
1  import express from "express"
2  const app = express()
3  const port = 3000
4
5  app.get("/", (req, res) => {
6      res.send("<h1> Hello Trung </h1>")
7      // console.log(req.rawHeaders);
8  })
9
10 app.get("/about", (req, res) => {
11     res.send("<h1> Đây là about </h1>")
12     // console.log(req.rawHeaders);
13 })
14
15 app.listen(port, () => {
16     console.log(`Server running on port ${port}.`)
17 })
```

→ Khởi chạy server

```
● PS D:\backend_learning\learn> cd ..
● PS D:\backend_learning> cd '.\3.1 Express Server\'
○ PS D:\backend_learning\3.1 Express Server> node .\index.js
Server running on port 3000.
█
```

→ Kiểm tra bằng Postman



<https://localhost:3333> Save

GET

http://localhost:3000/

Send

ParamsAuthorizationHeaders (8)Body •Pre-request ScriptTestsSettingsCookies

Query Params

	Key	Value	Bulk Edit
	Key	Value	

BodyCookiesHeaders (7)Test Results

200 OK2 ms250 BSave Response

PrettyRawPreviewVisualizeHTML

1<h1> Hello Trung </h1>

GET http://localhost:3333/allS [CONFLICT] GET https://locali+...

<https://localhost:3333> Save

GET

http://localhost:3000/about

Send

ParamsAuthorizationHeaders (8)Body •Pre-request ScriptTestsSettingsCookies

Query Params

	Key	Value	Bulk Edit
	Key	Value	

BodyCookiesHeaders (7)Test Results

200 OK2 ms254 BSave Response

PrettyRawPreviewVisualizeHTML

1<h1> Đây là about </h1>