1. **Chuyển thư mục trong cmd để chạy JavaScript:**

**Cách 1:**

E:

cd PhatTrienUngDungWeb\BaiTap\JavaScript

node tinhToanCanBan.js

**Giải thích:**

* E: → Chuyển sang ổ **E**
* cd PhatTrienUngDungWeb\BaiTap\JavaScript → Di chuyển đến thư mục **chứa file**
* node tinhToanCanBan.js → Chạy file bằng **Node.js**

**Cách 2:**

node "E:\PhatTrienUngDungWeb\BaiTap\JavaScript\tinhToanCanBan.js"

**Lưu ý:**

* Đường dẫn có khoảng trắng thì đặt trong **dấu ngoặc kép** "..."
* Kiểm tra lại tên file có đúng không (phải là tinhToanCanBan.js, không có lỗi .js.js hoặc .txt)

***\*Xóa lịch sử code vửa chạy: cls***

**Hàm mũi tên:**

Có 3 cách để viết hàm mũi tên:

- [1] Không tham số: () => { ... }

- [2] Một tham số: x => { ... } (không cần dấu ngoặc cho một tham số)

- [3] Nhiều tham số: (x, y) => { ... }

1. **Nginx: Lắng nghe trên cổng 65536 được không? Tại sao**

Không thể lắng nghe trên cổng 65536

Lý do:

* Cổng hợp lệ trong TCP/UDP chỉ nằm trong khoảng 0 - 65535.
* 65536 là số không hợp lệ vì nó vượt quá giới hạn của hệ thống mạng.
* Khi cố gắng sử dụng listen 65536;, Nginx sẽ báo lỗi "invalid port".

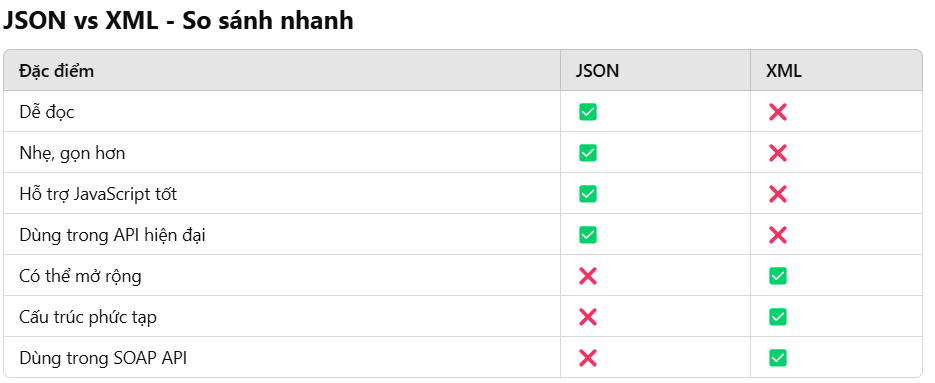
1. **Tổng quan về git:**

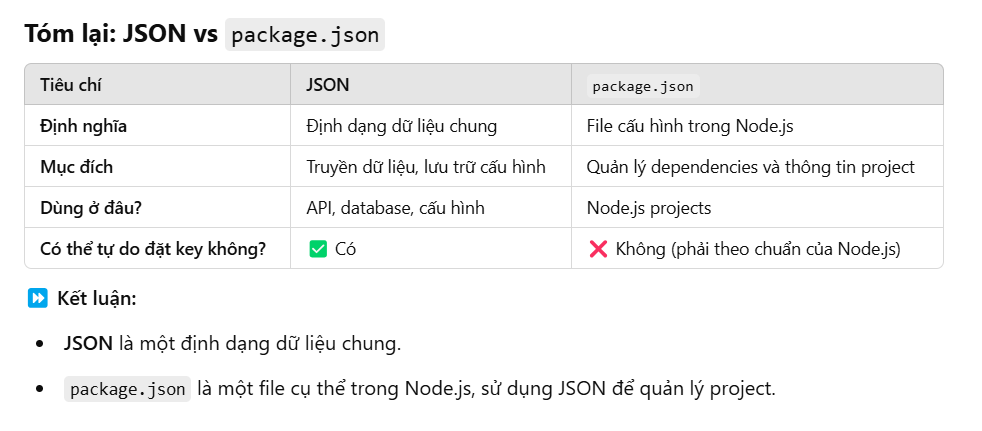
Khởi tạo git: git init (tạo kho chứa)

1. **Nodejs và NPM**

Xem phiên bản của Nodej: node -v

Xem phiên bản của NPM: npm -v







Tắt nodejs: Ctrl + C

Lệnh: ***npm init -y*** được sử dụng để tự động tạo một tập tin package.json trong dự án Node.js mà không cần phải trả lời các câu hỏi từng bước.

{

"name": "ptudw",

"version": "1.0.0",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"keywords": [],

"author": "",

"license": "ISC",

"dependencies": {

"express": "^5.1.0"

},

"devDependencies": {},

"description": ""

}

Trong đó:

* name: Tên thư mục hiện tại (dự án) sẽ được sử dụng làm tên của dự án.
* version: Phiên bản mặc định là 1.0.0.
* main: Tên file chính (tập tin đầu vào) của ứng dụng là index.js (nếu không có, bạn có thể thay đổi).
* scripts: Chứa một số script mặc định, ví dụ test, có thể tùy chỉnh sau.
* license: Mặc định là ISC.
* description: Trống.

Cài đặt lại Express bằng npm: *npm install express*

Kiểm tra lại thư mục node\_modules: *dir node\_modules*

Chạy lại ứng dụng: *node index.js* -> thành công: server dang chay tren cong 9000

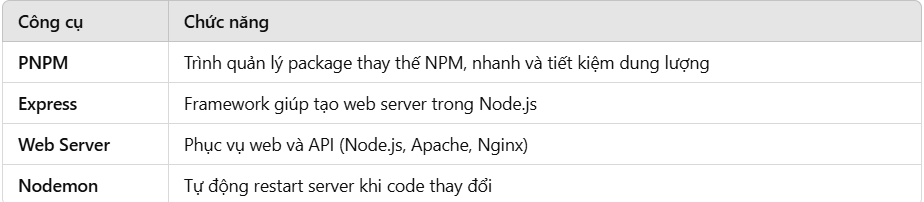
Chạy trên trình duyệt: [*http://localhost:9000/*](http://localhost:9000/)

1. **Nodemon**

Cài đặt Nodemon: *npm install --g nodemon*

Chạy nodemon: *nodemon*

**Kết luận:**



1. **Local và Global**
2. **Cài đặt một gói kiểu cục bộ (local)**

****

**VD:**

Cài đặt gói Express theo kiểu cục bộ: npm install express

Gỡ bỏ một gói cục bộ: npm uninstall express

**\*Chỉ có cài gói theo kiểu cục bộ thì mới được cập nhật vào package.json.**

1. **Cài đặt một gói kiểu toàn cục (global)**

Để cài đặt một gói kiểu toàn cục, sử dụng lệnh npm install có thêm tham số -g

npm install <package\_name> -g

Vd: Cài đặt gói nodemon theo kiểu toàn cục: install nodemon -g

+ Vị trí lưu trữ gói toàn cục: npm root -g

+ Xem các gói được cài đặt theo kiểu toàn cục: npm ls -g --depth 0

Trong đó,

- npm ls: để liệt kê các gói đã được cài đặt trên máy, ls là viết tắt của list

- -g là cờ báo, để chỉ hiển thị các gói kiểu toàn cục (global)

-  --depth 0: chỉ hiển thị các gói toàn cục cấp cao nhất (top-level), không hiển thị các gói là con-cháu (sub-dependencies) của gói cấp cao nhất

1. **Tổng quan về git**

Khởi tạo git, nhúng vào thư mục của dự án: git init

***Cấu hình định danh người dùng:***

Lệnh git config được sử dụng để đọc, ghi hoặc cập nhật các thiết lập cấu hình của git

*--system: Cấu hình toàn hệ thống (tất cả người dùng, tất cả repo), Áp dụng cho tất cả người dùng, mọi repo Git trên máy.*

VD: git config --system user.name “Quoc Lam”

*--global: cấu hình cho toàn bộ người dùng hiện tại. Áp dụng cho toàn bộ repo Git mà bạn sử dụng với tài khoản hiện tại.*

VD: git config --global user.name “Quoc Lam”

+ git config –global user.email “nguyenlequoclam@gmial.com”

Thư mực:  trong C:\Users\<tên\_người\_dùng> sau đó tìm file .gitconfig

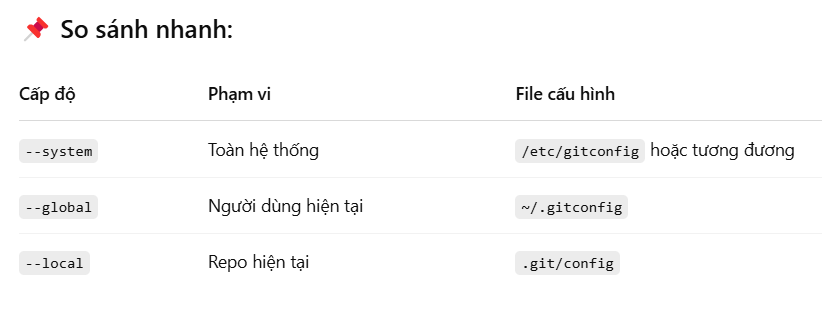
*--local: (mặc định) cấu hình cho repo hiện tại. Áp dụng chỉ trong repo Git hiện tại. Nếu không chỉ rõ cấp độ, Git sẽ mặc định dùng --local.*

VD: git config user.name "Tên riêng cho repo này"

+ git config user.email "email-rieng@example.com"

Thư mục: file .git trong thư mục dự án

***\*Lưu ý****: Trước khi chạy cần phải chạy dưới quyền admin: Windows -> cmd -> Bấm chuột phải vào tiện ích Command Prompt -> Chọn Run as administrator*

**



****

**Xem thông tin người dùng:**

Lệnh: git config –list

**Xóa định danh người dùng:**

git config --global --unset user.name

git config --global --unset user.email

Tương tự, bạn có thể xóa cấu hình ở phạm vi local hoặc system bằng cách bỏ qua hoặc thay thế --global bằng --local hoặc --system.

**Thay đổi định danh người dùng:**

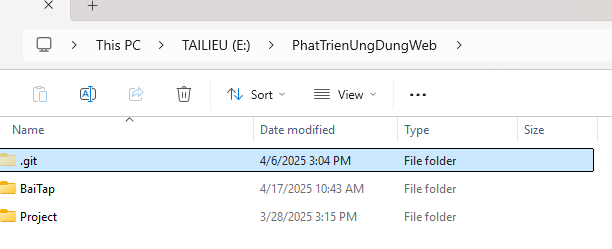
git config --global user.name "Tên mới của bạn"

git config --global user.email "email\_mới@example.com"

Lệnh này sẽ ghi đè lên cấu hình cũ.

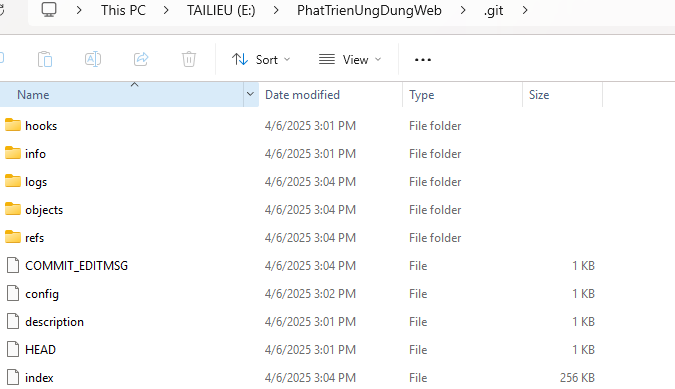
Tương tự, bạn có thể thay đổi cấu hình ở mức local hoặc system bằng cách bỏ qua hoặc thay thế --global bằng --local hoặc --system.

1. **Các khu vực làm việc của Git**

****

Kho chứa (repository): file .git

Thư mục làm việc (working directory): flie BaiTap, Project



Khu vực tổ chức tạm (staging area): file index

***\*Các lệnh của Git***

* Kiểm tra trạng thái: git status
* Thêm thay đổi vào staging area: git add .

- git add <tập\_tin>: thêm một tập tin cụ thể.

- git add .: thêm tất cả thay đổi.

- git add -p: thêm từng phần thay đổi (patch) trong tập tin.

* Commit thay đổi: git commit -m "text"

Để gỡ bỏ tập tin ra khỏi khu tạm:

git rm --cached <tên\_tập\_tin>

Hiển thị lích sử commit: git log

*'use strict'*

*const express = require('express')*

*const app = express();*

*const port = process.env.PORT || 9000*

*// xu ly khi nguoi dung gui request toi web server*

*app.get("/", (req, res) => {*

*const ten = req.query.name;*

*const lop = req.query.lop;*

*res.send(`Tên của tôi là ${ten} và học lớp ${lop}`);*

*});*

*// khoi dong web server*

*app.listen(port, () => {*

*console.log(`server dang chay tren cong ${port}`);*

*});*

http://localhost:9000/?name=NguyenLeQuocLam&lop=CTK46B

protocol (giao thức): http

domain (tên miền): localhost

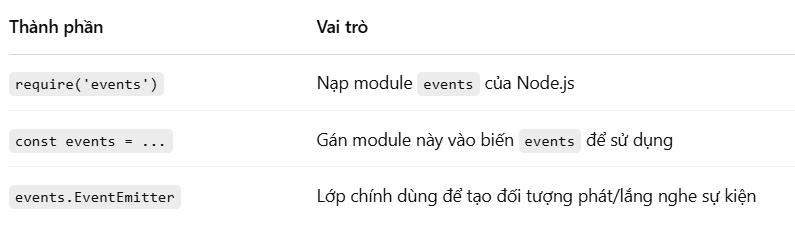
port (cổng): 9000

path (đường dẫn): / (là thư mục gốc).

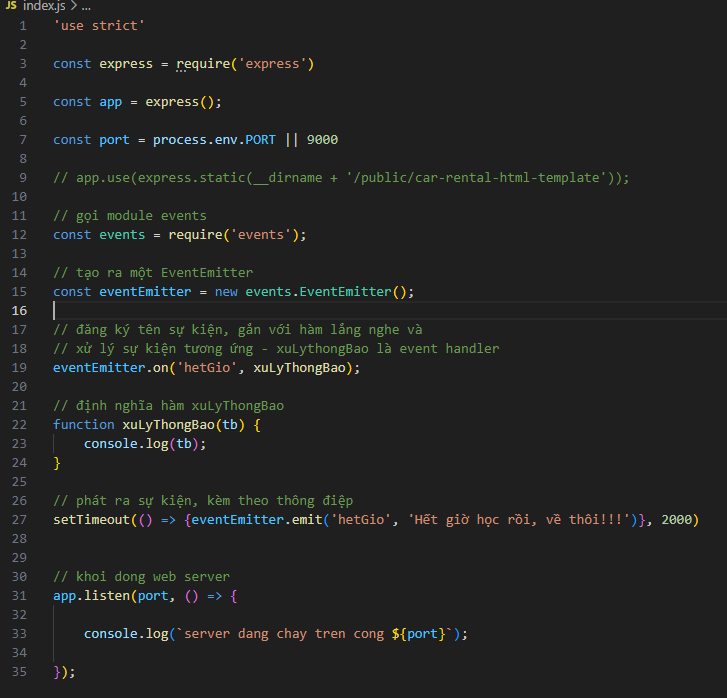
query (truy vấn): ?name=NguyenLeQuocLam&lop=CTK46B (chuỗi các tham số được sử dụng để truyền dữ liệu đến server).

1. **Lập trình hướng sự kiện**

const events = **require**('events');



Giải thích chi tiết



* **EventEmitter** – Bộ phát sự kiện

const events = require('events');

const eventEmitter = new events.EventEmitter();

* **Event Handler** – Bộ xử lý sự kiện

eventEmitter.on('hetGio', xuLyThongBao);

function xuLyThongBao(tb) {

console.log(tb);

}

* **Event** – Sự kiện xảy ra

setTimeout(() => {

eventEmitter.emit('hetGio', 'Hết giờ học rồi, về thôi!!!')

}, 2000)

* **Event Loop** – Vòng lặp sự kiện

Bạn không thấy rõ Event Loop trong code vì nó là **cơ chế nội tại của Node.js**, nhưng nó đang hoạt động ngầm để:

* Chờ sự kiện xảy ra
* Kích hoạt callback tương ứng

**Đây là mô hình hoạt động của Web Server theo kiến trúc Client-Server**

Nó chia làm 2 phía: Phía Client và phía Sever, phía Client dùng 3 ngôn ngữ để làm: HTML, CSS, JavaScript, phía Sever dùng 1 trong 6 ngôn ngữ như: PHP, Node.js, Python, Java, C#, Ruby

Bên sever: dịch và thực thi mã

B1: Người dùng dùng trình duyệt(Browser) gửi yêu cầu(Request) đến Web Server dựa trên giao thức HTTP và phương thức GET để gửi yêu cầu.

B2: Phân tích Request data: URL encoding, GET/POST Data, Cookies

* URL Encoding: Mã hóa ký tự đặc biệt trong URL để truyền dữ liệu an toàn.
* GET/POST Data: Gửi dữ liệu từ client lên server. GET hiển thị trên URL, POST ẩn trong body request.
* Cookies: Lưu thông tin trên trình duyệt để duy trì đăng nhập, giỏ hàng, hoặc theo dõi người dùng.

Sau khi lấy thông tin của gói Request data thì đẩy xuống Web Application (mã nguồn Backend) để xử lý

B3: Web Application biết được yêu cầu của Web Server và xử lý yêu cầu, truy xuất dữ liệu từ cơ sở dữ liệu gửi trở về cho Web Application

B4: Đổ dữ liệu sang định dạng HTML (vì trình duyệt không hiểu ngôn ngữ)

B5: Web Application gửi HTML đã xử lý sau đó trả về Web Server

B6: Trình duyệt người dùng hiển thị nội dung response

B7: Sau khi nhận và hiển thị HTML, trình duyệt tiếp tục gửi thêm request khác để lấy các nội dung tĩnh của ứng dụng web (HTML, CSS, JavaScript, hình ảnh, và các tài nguyên khác).