

**1. Em biết gì về web?**

**Web** (World Wide Web) là một hệ thống thông tin toàn cầu, trong đó tài liệu và tài nguyên được truy cập thông qua **trình duyệt web** bằng **giao thức HTTP/HTTPS**. Nội dung web được phân phối thông qua các **web server** và hiển thị cho người dùng thông qua **web browser**.

**2. Hạ tầng web (Web infrastructure)**

Hạ tầng web là tập hợp các thành phần giúp một website hoặc ứng dụng web hoạt động. Bao gồm:

* **Trình duyệt web** (Chrome, Firefox, Safari…) – giúp người dùng truy cập web.
* **Web server** (Apache, Nginx…) – nhận yêu cầu từ trình duyệt và phản hồi.
* **Cơ sở dữ liệu** (MySQL, PostgreSQL…) – lưu trữ dữ liệu web.
* **Backend** – xử lý logic, tương tác với cơ sở dữ liệu (PHP, Python, Node.js…).
* **Frontend** – giao diện người dùng (HTML, CSS, JavaScript).
* **Hosting & domain** – nơi lưu trữ web và địa chỉ truy cập.

**3. Phân biệt webpage, website và web app**

| **Thuật ngữ** | **Định nghĩa** |
| --- | --- |
| **Webpage** | Một trang đơn lẻ trên web (ví dụ: trang chủ, trang liên hệ). |
| **Website** | Một tập hợp nhiều webpage, liên kết với nhau dưới cùng một tên miền. |
| **Web App** | Là ứng dụng chạy trên trình duyệt, có chức năng tương tác với người dùng. |

📌 Ví dụ:

* Trang tin tức là **website** gồm nhiều **webpage**.
* Gmail, Facebook là **web app** vì có tính năng động, xử lý dữ liệu, đăng nhập…

**4. Để làm web app, cần 5 công cụ chính:**

**1. Ngôn ngữ lập trình (Programming Languages)**

Dùng để xây dựng frontend & backend:

* **Frontend**: HTML, CSS, JavaScript
* **Backend**: PHP, Python, Java, Node.js, Ruby, etc.

**2. Công cụ viết mã (Code Editor / IDE)**

* **Visual Studio Code** (phổ biến nhất)
* Sublime Text, Atom, WebStorm

**3. Công cụ quản lý cơ sở dữ liệu (Database tools)**

* **MySQL**, **PostgreSQL**, **MongoDB**
* Dùng cùng với **phpMyAdmin**, **pgAdmin**, **Mongo Compass** để quản lý.

**4. Công cụ quản lý mã nguồn (Source Control)**

* **Git** – công cụ quản lý mã nguồn phân tán.
* **GitHub**, **GitLab**, **Bitbucket** – nền tảng lưu trữ và quản lý dự án.

**5. Web server**

* **Apache**, **Nginx** – máy chủ web xử lý và phân phối nội dung.
* Với Node.js, thường dùng **Express.js** như một web server framework.

**[2] NPM: (thành phần, local, global, dependencies, devDependencies)**

* Node.js: môi trường thực thi JavaScript đa nền tảng, mã nguồn mở, và miễn phí, dùng để tạo web server, ứng dụng web và các công cụ dòng lệnh
* NPM: Công cụ quản lý các thư viện (package) trong ứng dụng Node.js..
* Thành phần của NPM:
* npm CLI: Công cụ dòng lệnh
* npm Registry: CSDL trực tuyến chứa các gói JavaScript mã nguồn mở
* package.json: Tệp cấu hình quan trọng nằm ở thư mục gốc của dự án

**1. Local packages (Gói cục bộ)**

* Cài đặt **cục bộ** (local) nghĩa là gói sẽ được cài trong thư mục node\_modules của dự án hiện tại.
* Được định nghĩa trong file package.json của dự án.
* Chỉ dùng cho **dự án hiện tại**, không ảnh hưởng đến hệ thống hoặc dự án khác.

**2. Global packages (Gói toàn cục)**

* Cài đặt **toàn cục** (global) nghĩa là có thể dùng gói đó **ở bất kỳ đâu trên máy tính**.
* Thường dùng cho các **công cụ dòng lệnh (CLI)** như nodemon, typescript, eslint, v.v

**3. dependencies**

* Là **các gói cần thiết khi ứng dụng chạy** (runtime).
* Được định nghĩa trong phần "dependencies" của package.json

**4. devDependencies**

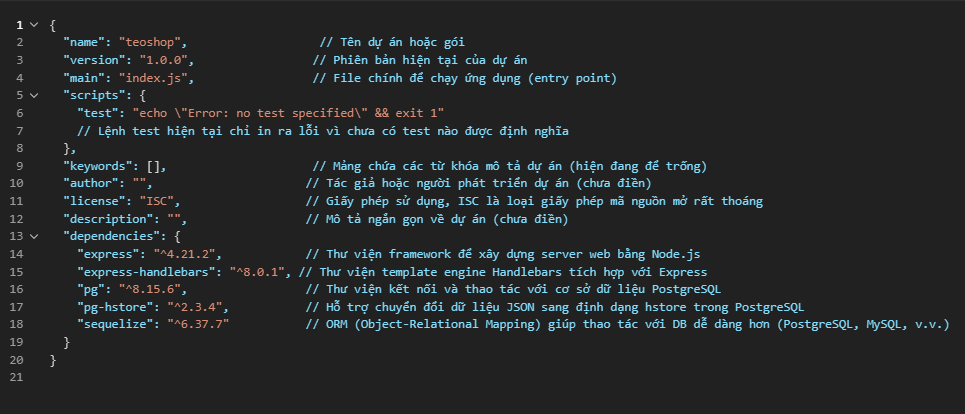
* Là **các gói chỉ dùng khi phát triển** (development), không cần khi chạy ứng dụng thực tế.
* Ví dụ: Babel, ESLint, Webpack, Jest, Prettier, TypeScript, v.v.
* Được định nghĩa trong "devDependencies"

Sự khác biệt chính giữa dependencies và devDependencies là mục đích sử dụng của gói:

* **dependencies:** Các gói cần thiết cho ứng dụng hoạt động bình thường khi đã được triển khai (production). Khi người khác cài đặt gói của bạn (ví dụ: nếu bạn phát triển một thư viện), họ sẽ cài đặt các gói trong dependencies.
* **devDependencies:** Các gói chỉ cần thiết cho quá trình phát triển và kiểm thử. Chúng không cần thiết cho ứng dụng chạy ở môi trường production. Khi người khác cài đặt gói của bạn, mặc định họ sẽ không cài đặt các gói trong devDependencies. Điều này giúp giảm kích thước của gói cài đặt cuối cùng.

Sự khác biệt chính giữa cài đặt gói NPM theo kiểu **local** và **global** nằm ở **phạm vi sử dụng** của gói đó:

* **Local:** Dành cho các thư viện là *phụ thuộc của dự án cụ thể*, chỉ dùng trong dự án đó.
* **Global:** Dành cho các *công cụ* dùng chung trên toàn hệ thống, có thể gọi từ dòng lệnh ở bất kỳ đâu

1. **Một gói nằm trong dependencies có dùng được trong quá trình phát triển (development) không?**

* **Có.** Chắc chắn dùng được.
* **Giải thích:** dependencies là các gói mà mã nguồn chính của dự án cần để hoạt động đúng. Quá trình phát triển bao gồm việc chạy thử, debug và kiểm thử ứng dụng, tất cả đều yêu cầu các gói trong dependencies phải có mặt và hoạt động. Nếu không có chúng, ứng dụng sẽ không thể chạy ngay cả trong môi trường phát triển.
* **Ví dụ:** Gói express trong file package.json của bạn. Bạn cần express để chạy server web của mình. Khi bạn phát triển (ví dụ: viết code xử lý request, test API), bạn chạy server này, do đó express là bắt buộc và được sử dụng trực tiếp trong quá trình phát triển.

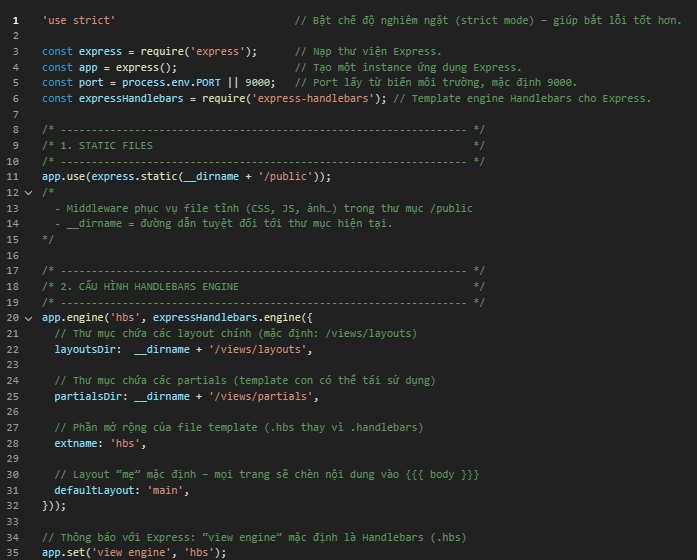
2. **Tại sao lại chia thành dependencies và devDependencies?**

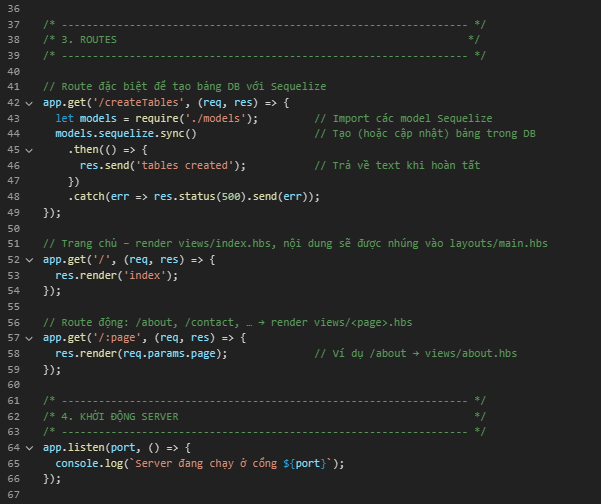
* Phân loại mục đích sử dụng: dependencies dành cho các gói cần thiết khi ứng dụng chạy ở môi trường production; devDependencies dành cho các gói chỉ cần thiết cho quá trình phát triển, kiểm thử, build.
* Tối ưu hóa khi triển khai (deploy): Cho phép cài đặt chỉ các gói cần thiết cho production (dùng npm install --production), giúp giảm kích thước và thời gian cài đặt ứng dụng khi deploy.
* Làm rõ cấu hình dự án: Giúp nhà phát triển khác dễ dàng nhận biết vai trò của từng gói trong dự án

**3. Một gói được cài đặt theo kiểu global có nằm trong package.json hay không?**

* **Không.** Các gói cài đặt global không nằm trong file package.json của dự án.
* **Giải thích:** File package.json là file cấu hình và khai báo phụ thuộc **của một dự án cụ thể**. Nó được dùng để tái tạo môi trường làm việc giống nhau cho dự án đó trên các máy khác nhau (chỉ cần clone code và chạy npm install). Các gói global lại được cài đặt cho **toàn bộ hệ thống** để sử dụng ở bất kỳ đâu, chúng không phải là phụ thuộc của riêng dự án đó.
* **Ví dụ:** Gói nodemon. Bạn thường cài nodemon global (npm install -g nodemon) để có thể dùng lệnh nodemon từ bất kỳ thư mục dự án nào để chạy file Node.js và tự động reload khi file thay đổi. nodemon là một công cụ *hỗ trợ* quá trình phát triển của bạn trên máy *cá nhân*, chứ không phải là thứ mà code của dự án bạn cần require() để chạy. Do đó, nó không thuộc về cấu hình của dự án và không được đưa vào package.json.

**[3] Lập trình với Express**





**[4] Callback, hàm trong js**

* **Hàm ẩn danh**: Hàm không có tên khi định nghĩa, được gán vào biến, truyền làm tham số, hoặc thực thi ngay lập tức
* **Hàm mũi tên:** Cú pháp ngắn gọn, không có ngữ cảnh riêng, không thể làm constructor, không có biến arguments (…) => {…}
* **Callback:** Là một **hàm** được **truyền làm đối số** cho một hàm khác, để hàm đó **gọi lại (thực thi) sau** khi hoàn thành công việc nào đó (thường là tác vụ bất đồng bộ).Lập trình đồng bộ: Các lệnh được thực thi tuần tự từ trên xuống dưới, có thể chặn luồng khi tác vụ mất nhiều thời gian

Lập trình bất đồng bộ: Cho phép thực hiện nhiều tác vụ mà không cần chờ đợi

Các kĩ thuật cho LTBĐB:

* Callback: Hàm được thực thi khi tác vụ hoàn tất
* Promise: Kết quả cuối cùng của 1 tác vụ bất đồng bộ
* Async/await:Viết mã bất đồng bộ giống như đồng bộ, giúp dễ đọc. Trong đó, async đặt trước function, await đặt trước lời gọi hàm

**[5]Lập trình Client – server**

Ứng dụng cục bộ: Ứng dụng chạy trực tiếp trên thiết bị, không cần internet

Ứng dụng client-server: Ứng dụng mà trong đó client hiển thị, server xử lí logic và dữ liệu, cần kết nối mạng giữa chúng

Ứng dụng web là ứng dụng client-server, trong đó server là web server và client là trình duyệt

Vai trò:

* + **Front-end**: Thiết kế giao diện
  + **Back-end**: Xử lý logic và dữ liệu
  + **Full-stack**: Làm được cả hai

**protocol://domain:port/path?query#fragment**

* protocol: Cách giao tiếp (ví dụ: http, https).
* domain: Tên máy chủ/trang web (ví dụ: google.com).
* :port: Cổng kết nối trên máy chủ (thường ẩn nếu là mặc định).
* /path: Vị trí tài nguyên cụ thể trên máy chủ (ví dụ: /trang/gioithieu).
* ?query: Dữ liệu bổ sung gửi lên server (sau ?, ví dụ: ?id=1).
* #fragment: Vị trí cụ thể trong trang/tài nguyên (sau #, chỉ xử lý ở trình duyệt).

**[6]Lập trình hướng sự kiện**

Khái niệm: Là mô hình lập trình mà luồng thực thi của chương trình được quyết địng bởi các sự kiện

Thành phần chính:

* Event: Sự kiện (nhấn chuột, gõ phím,…)
* Event Emitter: Bộ quản lí phát ra sự kiện gửi đến nơi cần nghe
* Event Handler: Hàm xử lí sự kiện, được gọi khi sự kiện xảy ra
* Event Loop: Vòng lặp sự kiện kiểm tra hàng đợi và xử lí callback

Mô hình sự kiện:

Event Emitter phát ra sự kiện, đưa vào hàng đợi (Events), Event Loop kiểm tra, lấy sự kiện chuyển cho Event Handler xử lí, Event Loop tiếp tục vòng lặp, chờ sự kiện tiếp theo

**[7]CSDL**

Dữ liệu thô: Chuỗi các kí hiệu được lưu dưới dạng bit 0 và 1 trong máy, không có ngữ nghĩa, cần được xử lí

Tháp xử lí thông tin: Data -> Information -> Knowledge -> Wisdom

Cơ sở dữ liệu: Tập các dữ liệu được tổ chức, lưu trữ và quản lí có hệ thống trong máy, giúp truy cập dễ dàng, giảm trùng lặp, bảo mật cai và hỗ trợ đa người dùng

Thành phần CSDL: Gồm dữ liệu, mô hình quan hệ hoặc phi quan hệ, hệ quản trị CSDL

Hệ quản trị CSDL: Phần mềm quản lí CSDL

RDBMS: DBMS quản lí CSDL quan hệ

Thành phần của SQL Server:

* Database Engine: Thành phần cốt lõi
* Sqlcmd: Công cụ dòng lệnh
* SSMS: Giao diện đồ hoạ

Lập trình với CSDL:

Code First: Định nghĩa mô hình dữ liệu trước trong mã nguồn

Database First: Thiết kế CSDL trước

Code First linh hoạt, phát triển nhanh, nhưng khó tích hợp với CSDL có sẵn, phú hợp dự án chưa có CSDL, phát triển nhanh, tập trung logic ứng dụng

Database First phù hợp với hệ thống sẵn có, tối ưu hoá hiệu suất nhưng ít linh hoạt

**[8]Git**

Khái niệm: Git là hệ thống quản lí phiên bản phân tán sử dụng rộng rãi trong phát triển phần mềm, cho phép theo dõi và quản lí các thay đổi trong mã nguồn một cách hiệu quả

Tính năng:

* Theo dõi lịch sử thay đổi
* Hỗ trợ đa người dùng
* Phân nhánh và hợp nhất
* Lưu trữ dưới dạng bản ghi, đảm bảo bảo mật
* Hỗ trợ làm việc không cần kết nối mạng và đồng bộ hóa sau đó

Một số lệnh cmd:

* Kiểm tra: git –version hoặc git –v
* Nhúng Git: git init
* Đinh danh người dùng: git config –[system/global] user.name=”…”
* Xem cấu hình: git config –list
* Xoá: git config –[system/global] –unset user.name

Khu vực làm việc:

* Thư mục làm việc: Nơi chỉnh sửa trực tiếp các tập tin trên máy tính, lưu các thay đổi được Git theo dõi hoặc chưa.

Cách khởi tạo:

* Tạo mới: git init
* Sao chép: git clone
* git add: Đưa các thay đổi từ Thư mục làm việc vào Khu tạm
* git status: Hiển thị trạng thái hiện tại của Kho lưu trữ và Thư mục làm việc
* Khu tạm: Nơi chọn lọc và tổ chức các thay đổi đưa vào commit tiếp theo.
* gitignore: Chỉ định các tập tin hoặc thư mục mà Git sẽ bỏ qua, giúp tránh commit rác, giữ lịch sử sạch, giảm kích thước kho chứa, tránh xung đột
* Kho chứa: Nơi lưu trữ tất cả các tập tin và lịch sử thay đổi của dự án. Thư mục .git chính là kho chứa này.
* git commit: Lưu các thay đổi từ Khu tạm vào Kho chứa