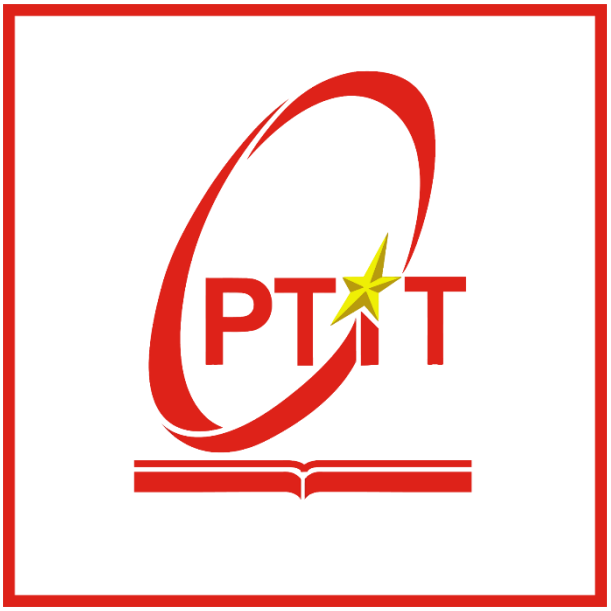
BỘ THÔNG TIN VÀ TRUYỀN THÔNG
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



BÁO CÁO THỰC TẬP CƠ SỞ TUẦN 2

TÌM HIỂU VỀ NODE.JS

Giảng viên hướng dẫn: TS. Kim Ngọc Bách

Sinh viên thực hiện:

Lê Quang Thanh – B22DCVT509

MỤC LỤC

I. Node.js là gì?
II. Nodejs hoạt động ra sao?
    1. Kiến trúc của Node.js
    2. Cơ chế hoạt động của Node.js
III. Thành phần của Node.js
    1. Module
    2. Console
    3. Cluster
    4. Global
    5. Luồng (Streaming)
    6. Đệm (Butter)
    7. Miền (Domain)
    8. DNS
    9. Trình gỡ lỗi
IV. Tại sao nên sử dụng Node.js?
V. Ứng dụng của Node.js

I. Node.js là gì?

- NodeJS là một nền tảng được xây dựng trên V8 Javascript engine được viết bằng C++ và Javascript. Nền tảng này được phát triển bởi Ryan Lienhart Dahl vào năm 2009.

- Node.js ra đời khi các developer đời đầu của JavaScript mở rộng nó từ một thứ bạn chỉ chạy được trên trình duyệt thành một thứ bạn có thể chạy trên máy của mình dưới dạng ứng dụng độc lập.

1. **Nguồn mở (Open-source)**: Mã nguồn của Node.js được công bố công khai, điều này có nghĩa là bất kỳ ai cũng có thể truy cập, sử dụng, và đóng góp vào mã nguồn.
2. **Đa nền tảng (Cross-platform)**: Node.js không phụ thuộc vào bất kỳ hệ điều hành nào cụ thể nào, nghĩa là nó có thể chạy trên Linux, macOS hoặc Windows. Điều này làm cho Node.js trở thành một lựa chọn linh hoạt cho các nhà phát triển muốn xây dựng các ứng dụng có thể hoạt động trên nhiều nền tảng khác nhau mà không cần thay đổi mã nguồn.
3. **Môi trường thực thi JavaScript (JavaScript runtime environment)**: Để mã JavaScript có thể được thực thi, nó cần một môi trường chạy phù hợp. Trong khi trình duyệt như Chrome và Firefox cung cấp một môi trường thực thi cho JavaScript, Node.js mở rộng khả năng này ra ngoài trình duyệt.
4. **Dựa trên V8 JavaScript Engine**: Node.js được xây dựng dựa trên V8, động cơ JavaScript được phát triển bởi Google cho trình duyệt Chrome. Điều này giúp Node.js có khả năng thực thi JavaScript nhanh và hiệu quả, đồng thời hỗ trợ các tính năng mới nhất của ngôn ngữ JavaScript.

⇨ Node.js đã mở rộng khả năng của JavaScript từ việc chỉ phát triển front-end trong trình duyệt để bao gồm cả phát triển back-end. Điều này có nghĩa là các lập trình viên có thể sử dụng cùng một ngôn ngữ lập trình, JavaScript, để phát triển toàn bộ ứng dụng, từ front-end đến back-end, qua đó tạo điều kiện cho việc học tập và phát triển ứng dụng nhanh chóng và hiệu quả hơn.

II. Nodejs hoạt động ra sao?

1. Kiến trúc của Node.js

- Non-blocking I/O và Event-Driven:

- Node.js sử dụng một mô hình non-blocking I/O (input/output) và event-driven, nghĩa là các hoạt động như đọc file, truy vấn cơ sở dữ liệu, hoặc giao tiếp mạng được thực hiện mà không chặn tiến trình chính. Điều này cho phép xử lý nhiều yêu cầu cùng lúc mà không cần tạo nhiều luồng (thread), giúp giảm bớt chi phí liên quan đến quản lý luồng và tối ưu hóa hiệu suất.
- Khi một hoạt động I/O được khởi tạo, nó sẽ được gửi đến thực thi trong hệ thống hoặc cơ sở dữ liệu mà không làm chậm tiến trình chính. Sau khi hoạt động hoàn tất, một sự kiện sẽ được phát đi và xử lý bằng các hàm gọi lại (callback).

- **V8 JavaScript Engine:**

Node.js được xây dựng trên động cơ JavaScript V8 của Google Chrome, đây là một động cơ rất nhanh cho phép biên dịch mã JavaScript thành mã máy để thực thi trực tiếp trên phần cứng, làm tăng hiệu suất thực thi.

- **Single-Threaded:**

Mặc dù Node.js hoạt động trên một luồng duy nhất cho logic ứng dụng của người dùng, nó vẫn sử dụng nhiều luồng ở tầng thấp hơn thông qua thư viện libuv để xử lý các hoạt động I/O. Tuy nhiên, những chi tiết này được ẩn giấu khỏi người dùng, giúp việc lập trình đơn giản hơn mà vẫn đảm bảo hiệu suất.

- **Event Loop:**

Đây là vòng lặp sự kiện mà ở đó Node.js tiếp tục lắng nghe sự kiện và thực hiện các hàm gọi lại khi một sự kiện được kích hoạt. Vòng lặp sự kiện cho phép Node.js xử lý hàng nghìn kết nối đồng thời mà không cần phải tạo ra chi phí quản lý luồng.

- **Trigger Callback:**

Khi thao tác I/O hoàn tất, hệ điều hành thông báo cho Node.js, và Node.js sau đó thực thi hàm callback tương ứng để xử lý kết quả hoặc tiếp tục xử lý logic.

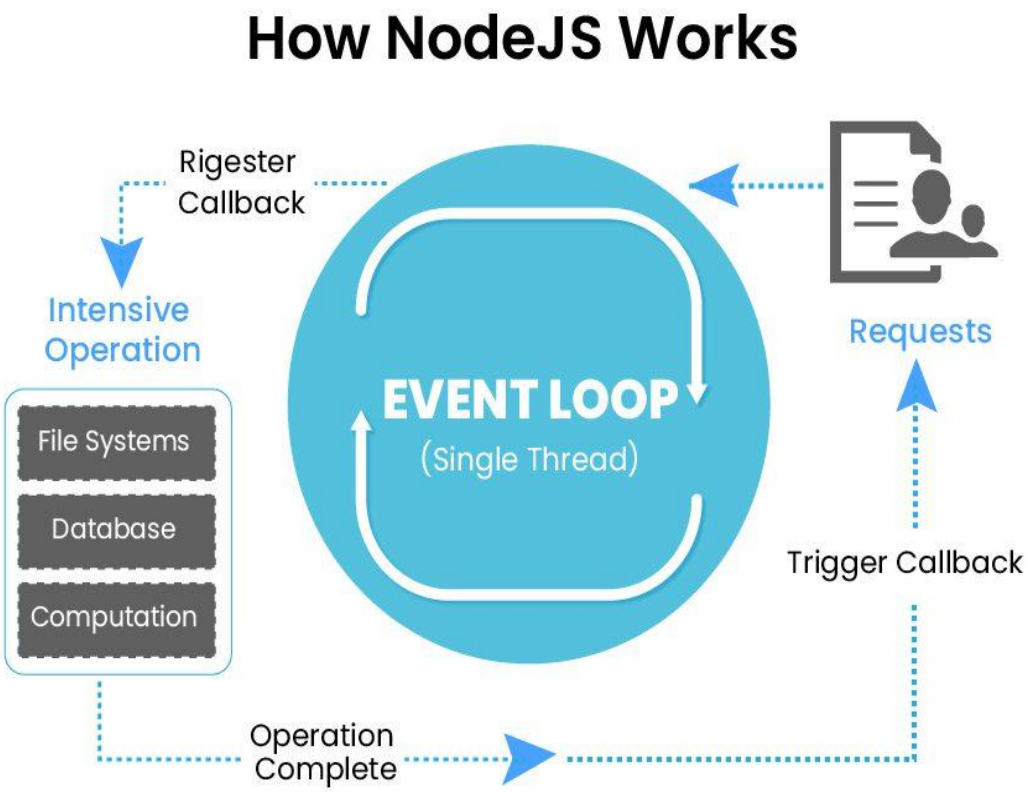
- **NPM (Node Package Manager):**

NPM là hệ thống quản lý gói cho Node.js, cho phép các nhà phát triển dễ dàng chia sẻ và sử dụng mã nguồn từ nhau. NPM là một trong những kho lưu trữ mã nguồn mở lớn nhất thế giới và chứa hàng ngàn module có thể được tích hợp vào ứng dụng của bạn.

- **Require:** Require là 1 chức năng, và nó nhận tham số path tinh chỉnh và trả về module.export .

- Tải module đi kèm với Node.js như hệ thống file và HTTP từ Node.js API.
- Tải thư viện thứ 3 như Express và Mongoose mà bạn cài đặt từ npm.
- Giúp require file và mo-đun hoá project.

2. Cơ chế hoạt động của Node.js



- Node.js được sinh ra để phát triển server-side. Node.js hoạt động dựa trên một số nguyên tắc cơ bản giúp nó hiệu quả trong việc xử lý các ứng dụng có nhiều hoạt động nhập/xuất (I/O) mà không bị chặn, đồng thời giảm đáng kể sự phức tạp trong quản lý các luồng thực thi.

- Ý tưởng chính của Node js là sử dụng non-blocking, hướng sự vào ra dữ liệu thông qua các tác vụ thời gian thực một cách nhanh chóng. Bởi vì, Node js có khả năng mở rộng nhanh chóng, khả năng xử lý một số lượng lớn các kết nối đồng thời bằng thông lượng cao. Nếu như các ứng dụng web truyền thống, các request tạo ra một luồng xử lý yêu cầu mới và chiếm RAM của hệ thống thì việc tài nguyên của hệ thống sẽ được sử dụng không hiệu quả. Chính vì lẽ đó giải pháp mà Node js đưa ra là sử dụng luồng đơn (Single-Threaded), kết hợp với non-blocking I/O để thực thi các request, cho phép hỗ trợ hàng chục ngàn kết nối đồng thời.

- **Ví dụ:** Khi có request đến server, nó đẩy event vào Event Queue, event loop sẽ nhận lần lượt các event để xử lý. Nếu event đó không Block lại chương trình quá lâu, nó tự xử lý rồi trả response lại. Rồi có những request có thao tác truy vấn database, việc truy vấn sẽ tốn rất nhiều thời gian, Event Loop sẽ đưa công việc qua Thread Pool (thông qua thư viện Libuv đã nói phía trên), khi thread đó đã thực hiện xong thì có kết quả truy vấn database, Node.js đẩy callback của event đó về Event Queue để xử lý việc này. Callback chính là function Javascript xử lý sau khi có kết quả truy vấn database, có thể là response ngay về client.

III. Thành phần của Node.js

1. Module

-Module giống như thư viện JavaScript có thể được sử dụng trong ứng dụng Node.js để bao gồm một tập hợp các hàm. Để bao gồm một module trong ứng dụng Node.js, hãy sử dụng hàm require() với dấu ngoặc đơn chứa tên của module.

```
const http = require("http");
```

2. Console

-Console là một module cung cấp phương pháp gỡ lỗi tương tự như console JavaScript cơ bản do trình duyệt internet cung cấp. Nó in thông báo ra stdout và stderr.

```
console.log('hello world')
```

3. Cluster

-Node.js được xây dựng dựa trên khái niệm lập trình đơn luồng. Cluster là một mô-đun cho phép đa luồng bằng cách tạo các tiến trình con chia sẻ cùng một cổng máy chủ và chạy đồng thời.

4. Global

-Các đối tượng toàn cục trong Node.js có sẵn trong tất cả các mô-đun. Các đối tượng này là hàm, mô-đun, chuỗi, v.v. Một số đối tượng toàn cục của Node.js được đề cập trong bảng dưới đây:

Đối tượng toàn cục	Mô tả
global	Đối tượng toàn cục chung, tương tự như `window` trong trình duyệt.
process	Cung cấp thông tin và điều khiển về quá trình thực thi Node.js.

Đối tượng toàn cục	Mô tả
console	Cung cấp các phương thức để ghi log thông tin, lỗi và các thông báo khác.
Buffer	Lớp toàn cục giúp làm việc với các dữ liệu nhị phân (binary data).
setTimeout	Lên lịch thực thi một hàm sau một khoảng thời gian nhất định.
clearTimeout	Hủy một hàm đã được lên lịch bởi setTimeout.
setInterval	Lên lịch thực thi một hàm nhiều lần, với khoảng thời gian cố định giữa các lần thực thi.
clearInterval	Hủy một hàm đã được lên lịch bởi setInterval.
setImmediate	Thực thi một hàm ngay lập tức sau khi vòng lặp sự kiện hiện tại kết thúc.
clearImmediate	Hủy một hàm đã được lên lịch bởi setImmediate.
__dirname	Chứa đường dẫn tới thư mục của tệp hiện tại.
__filename	Chứa đường dẫn tuyệt đối tới tệp hiện tại.
module	Đối tượng liên quan đến mô-đun hiện tại, chứa thông tin về mô-đun.
require()	Hàm dùng để nhập các mô-đun khác vào tệp hiện tại.

5. Luồng (Streaming)

- Luồng là các đối tượng cho phép bạn đọc dữ liệu hoặc ghi dữ liệu liên tục. Có bốn loại luồng:

- Có thể đọc được: Đây là loại luồng mà dữ liệu có thể được đọc
- Có thể ghi: Đây là loại luồng mà dữ liệu có thể được ghi vào
- Duplex: Đây là cả luồng có thể đọc và ghi
- Chuyển đổi: Các luồng có thể thao tác dữ liệu trong khi nó đang được đọc hoặc ghi.

6. Đệm (Butter)

-Buffer là một module cho phép xử lý các luồng chỉ chứa dữ liệu nhị phân. Một bộ đệm rỗng có chiều dài ’10’ có thể được tạo bằng phương pháp này:

```
var buf = Butter.alloc(10)
```

7. **Miền (Domain)**

-Mô-đun miền chặn các lỗi vẫn chưa được xử lý. Có hai phương pháp được sử dụng để chặn các lỗi này:

- Liên kết nội bộ: Trình phát lỗi thực thi mã của nó bên trong phương thức chạy.
- Liên kết bên ngoài: Bộ phát lỗi được thêm rõ ràng vào miền thông qua phương thức add của nó.

8. **DNS**

-Mô-đun DNS được sử dụng để kết nối với máy chủ DNS và thực hiện phân giải tên bằng phương pháp sau:

```
dns.resolve()
```

-Mô-đun DNS cũng được sử dụng để thực hiện phân giải tên mà không cần kết nối mạng bằng cách sử dụng phương pháp sau:

```
dns.lookup()
```

9. **Trình gỡ lỗi**

-Node.js bao gồm một tiện ích gỡ lỗi có thể được truy cập bằng một trình gỡ lỗi tích hợp. Trình gỡ lỗi Node.js không có nhiều tính năng nhưng hỗ trợ kiểm tra mã đơn giản. Trình gỡ lỗi có thể được sử dụng trong thiết bị đầu cuối bằng cách sử dụng từ khóa 'inspect' trước tên tệp JavaScript. Để kiểm tra tệp — ví dụ: myscript.js—bạn có thể làm theo phương pháp này:

```
$ node inspect myscript.js
```

IV. Tại sao nên sử dụng Node.js?

1. **Hiệu suất cao**: Node.js được xây dựng trên động cơ JavaScript V8 của Google Chrome, cho phép biên dịch mã JavaScript thành mã máy nhanh chóng. Nhờ đó, thời gian thực thi của Node.js rất nhanh, làm tăng hiệu suất của các ứng dụng.
2. **Hệ sinh thái phong phú**: Với hơn 50,000 gói có sẵn trong Node Package Manager (NPM), các nhà phát triển có thể dễ dàng tìm và sử dụng các thư viện theo nhu cầu của họ mà không cần phải viết lại từ đầu, tiết kiệm đáng kể thời gian và công sức.

3. **Xử lý bất đồng bộ và không chặn (Asynchronous and Non-blocking)**: Node.js hoạt động một cách bất đồng bộ và không chặn các hoạt động I/O, nghĩa là nó không cần chờ đợi API trả về dữ liệu trước khi tiếp tục xử lý yêu cầu tiếp theo. Điều này làm cho Node.js trở nên lý tưởng cho việc xây dựng các ứng dụng web thời gian thực và xử lý dữ liệu lớn.
4. **Tính nhất quán trong mã nguồn**: Node.js cho phép sử dụng cùng một ngôn ngữ lập trình (JavaScript) cho cả phía máy chủ và máy khách. Điều này không chỉ giúp giảm thiểu sự không đồng bộ giữa client và server mà còn làm cho việc bảo trì và quản lý mã nguồn trở nên dễ dàng hơn.
5. **Khả năng mở rộng**: Node.js hỗ trợ xây dựng các ứng dụng có khả năng mở rộng cao thông qua mô hình sự kiện và bất đồng bộ của mình. Điều này cho phép xử lý hàng ngàn kết nối đồng thời mà không làm giảm hiệu suất.
6. **Ngôn ngữ quen thuộc**: Vì Node.js là một khung làm việc JavaScript, nó trở thành lựa chọn lý tưởng cho những nhà phát triển đã quen thuộc với JavaScript. Điều này làm cho quá trình học tập và phát triển dự án với Node.js trở nên dễ dàng hơn nhiều.

⇨ Nhờ những đặc điểm này, Node.js đã trở thành một trong những lựa chọn hàng đầu cho việc phát triển phía máy chủ, đặc biệt là trong các dự án yêu cầu hiệu suất cao và khả năng xử lý đồng thời lớn.

V. Ứng dụng của Node.js

- Tất cả các tính năng mạnh mẽ của NodeJS đã làm cho nó trở thành lựa chọn hàng đầu cho nhiều kỹ sư phần mềm. Dưới đây là những ứng dụng mà NodeJS thường được sử dụng:

- Ứng dụng trò chuyện thời gian thực: NodeJS rất phù hợp để xây dựng các ứng dụng trò chuyện thời gian thực như ứng dụng trò chuyện trực tuyến. NodeJS cho phép máy chủ xử lý các yêu cầu từ người dùng mà không gây trễ trong giao diện người dùng chính.
- NodeJS cung cấp một cơ sở hạ tầng mạnh mẽ để xây dựng các nền tảng truyền thông xã hội. Nên NodeJS là một lựa chọn lý tưởng cho việc xây dựng các ứng dụng như mạng xã hội.
- NodeJS phát triển các ứng dụng Internet of Things (IoT): Nó cho phép xử lý dữ liệu trong thời gian thực và phản hồi nhanh chóng.
- NodeJS hỗ trợ xây dựng các ứng dụng phát trực tuyến như live-streaming, ứng dụng tin tức theo thời gian thực. NodeJS cho phép việc phát trực tuyến nhanh chóng và mượt mà.

- Công cụ quản lý quan hệ khách hàng (CRM): NodeJS có thể được sử dụng để xây dựng các công cụ CRM linh hoạt và hiệu quả. NodeJS giúp tạo ra các ứng dụng CRM mạnh mẽ và ổn định.
- Nền tảng thương mại điện tử: NodeJS cung cấp khả năng mở rộng cao nên là lựa chọn phổ biến cho việc xây dựng các trang web thương mại điện tử. NodeJS giúp tạo ra các trang web thương mại điện tử nhanh chóng và ổn định.
- Nền tảng E-learning: NodeJS là một lựa chọn lý tưởng để xây dựng các nền tảng E-learning. Với khả năng xử lý lượng truy cập lớn và mở rộng linh hoạt trên những ứng dụng khác. NodeJS đảm bảo rằng E-learning có thể cung cấp cho người học những trải nghiệm tốt nhất.