BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC PHENIKAA

ĐỒ ÁN CƠ SỞ

TÌM HIỂU NGÔN NGỮ LẬP TRÌNH NODEJS VÀ XÂY DỰNG ỨNG DỤNG

Giảng viên hướng dẫn: **Nguyễn Thị Thùy Liên**

Sinh viên thực hiện

1. Nguyễn Thị Nga MSSV: 20010809
2. Hoàng Minh Thu MSSV: 20010831

HÀ NỘI, 12/2022

LỜI MỞ ĐẦU

Ngày nay, ngành công nghệ thông tin đang phát triển vô cùng mạnh mẽ. Những thành tựu của ngành CNTT được ứng dụng ở mọi lĩnh vực và dần trở thành một phần quan trọng trong cuộc sống hiện đại. Vì vậy để theo kịp xu hướng và có thể phát triển hơn nữa thì việc tìm hiểu thêm về các loại ngôn ngữ lập trình khác nhau là một điều cần thiết. Trong đồ án cơ sở này chúng em tìm hiểu về ngôn ngữ lập trình NodeJS. NodeJS là một nền tảng được phát triển độc lập được xây dựng trên javascript runtime của chrome's V8 Engine có thể xây dựng được ứng dụng mạng nhanh chóng và dễ dàng mở rộng. NodeJS được xây dựng và phát triển từ năm 2009 và được bảo trợ từ công ty Joyent. Khả năng đáp ứng của NodeJS là rất nhanh bởi phần core của NodeJS được viết hầu hết bởi ngôn ngữ C++. NodeJS đang trở thành một xu hướng của giới lập trình back-end. Có rất nhiều ứng dụng lớn của các công ty lớn đang sử dụng NodeJS. Có thể kể tên như: Paypal, Netflix, LinkedIn…

Mục tiêu của báo cáo này là giúp hiểu thêm về ngôn ngữ lập trình NodeJS và có thể áp dụng để xây dựng một Website và triển khai nó trên Internet.

Trong quá trình thực hiện đồ án, nhóm em vẫn còn nhiều thiếu sót, rất mong nhận được sự góp ý của cô và các bạn. Để hoàn thành tốt để tài và bài báo cáo này, chúng em xin gửi lời cảm ơn chân thành đến giảng viên hướng dẫn, cô Nguyễn Thị Thùy Liên, người đã trực tiếp hỗ trợ chúng em trong suốt quá trình làm đồ án. Chúng em cảm ơn cô đã đưa ra những lời khuyên từ kinh nghiệm thực tiễn của mình để định hướng cho chúng em đi đúng với yêu cầu của đề tài đã chọn, luôn giải đáp thắc mắc và đưa ra những góp ý, chỉnh sửa kịp thời giúp chúng em khắc phục nhược điểm và hoàn thành tốt cũng như đúng thời hạn đã đề ra.

MỤC LỤC

Bảng phân chia công việc nhóm 3		
STT	**HỌ VÀ TÊN**	**CÔNG VIỆC ĐẢM NHẬN**
1	Nguyễn Thị Nga	Tìm hiểu tổng quan về NodeJS Chạy thực nghiệm chức năng CRUD cơ bản Xây dựng cơ bản website bán hàng kết hợp hệ quản trị cơ sở dữ liệu MySql
2	Hoàng Minh Thu	Tìm hiểu tổng quan về NodeJS Chạy thực nghiệm MongoDB Xây dựng cơ bản website bán hàng kết hợp hệ quản trị cơ sở dữ liệu MySql

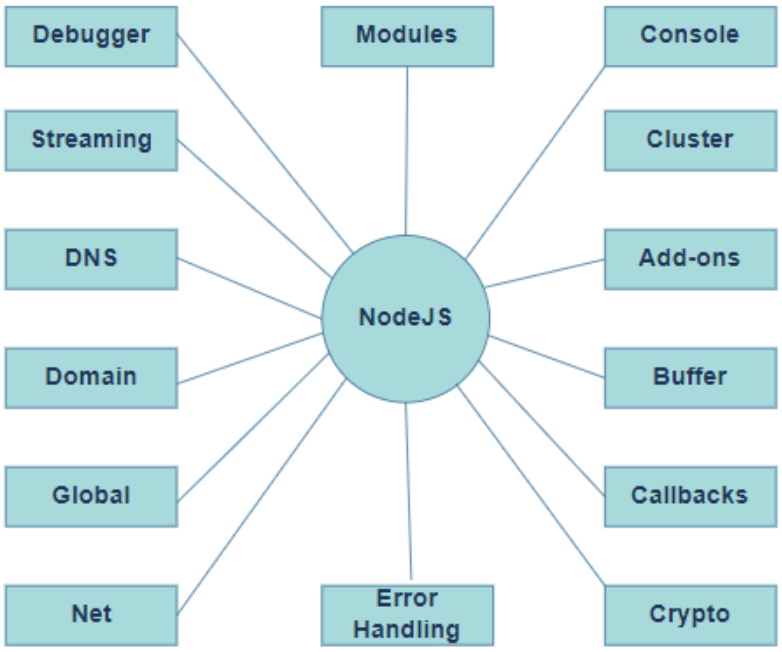
Bảng 1: Phân chia công việc nhóm

PHẦN I: TÌM HIỂU TỔNG QUAN VỀ NODEJS

1. TỔNG QUAN

NodeJS là một mã nguồn được xây dựng dựa trên nền tảng Javascript V8 Engine, nó được sử dụng để xây dựng các ứng dụng web như các trang video clip, các forum và đặc biệt là trang mạng xã hội phạm vi hẹp. NodeJS là một mã nguồn mở được sử dụng rộng rãi trên toàn thế giới. Nó có thể chạy trên nhiều nền tảng hệ điều hành khác nhau từ Linux, Microsoft Windows, Mac OS X. NodeJS cung cấp các thư viện phong phú ở dạng Javascript Module khác nhau giúp đơn giản hóa việc lập trình và giảm thời gian ở mức thấp nhất.

NodeJS có rất nhiều đặc tính. Một trong những đặc tính quan trọng của NodeJS phải kể đến là *không đồng bộ*: Tất cả các API của NodeJS đều không đồng bộ (none-blocking), nó chủ yếu dựa trên nền của NodeJS Server và chờ đợi Server trả dữ liệu về (Realtime). Realtime ở đây chính là xử lý giao tiếp từ client tới máy chủ theo thời gian thực. Đặc tính tiếp theo là *chạy rất nhanh:* NodeJS NodeJS được xây dựng dựa vào nền tảng V8 Javascript Engine nên việc thực thi chương trình rất nhanh. NodeJS là *đơn luồng nhưng khả năng mở rộng cao*: NodeJS sử dụng một mô hình luồng duy nhất với sự kiện lặp. Cơ chế tổ chức sự kiện giúp các máy chủ để đáp ứng một cách không ngăn chặn và làm cho máy chủ cao khả năng mở rộng. NodeJS sử dụng một chương trình đơn luồng và các chương trình tương tự có thể cung cấp dịch vụ cho một số lượng lớn hơn nhiều so với yêu cầu máy chủ truyền thống như Apache HTTP Server. NodeJS đã được cấp giấy phép MIT License.



Sơ đồ về các thành phần quan trọng trong NodeJS

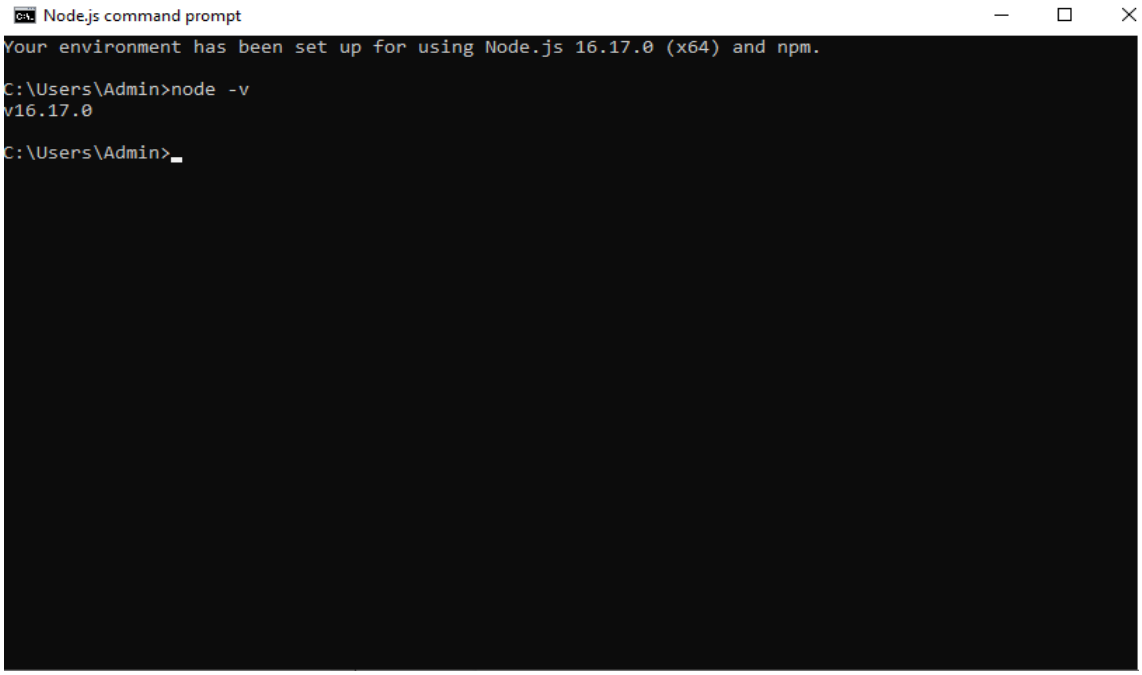
Node cho phép thực hiện các giao thức mạng ở cấp độ thấp một cách dễ dàng. Chẳng hạn như Node có module HTTP cho phép xây dựng một webserver chỉ với vài dòng code, tuy nhiên vì thế mà sẽ phải học nhiều thứ hơn như học về các header của một gói tin HTTP. Tuy nhiên vì NodeJS là một framework mã nguồn mở, do đó cũng có một số thư viện hỗ trợ viết webserver nhanh hơn và dễ hơn cho coder. Hai framework sử dụng phổ biến là Express và SocketIO. ExpressJS dùng để xây dựng API và website. Còn SocketIO cho phép xây dựng một các ứng dụng realtime như chatbot, tickers, dashboard APIs, và nhiều ứng dụng khác. SocketIO có lợi ích hơn so với NodeJS thông thường ở chỗ: hỗ trợ route URL tùy chỉnh cho web socket; tích hợp dễ dàng hơn với ExpressJS; hỗ trợ clustering với Redis.

Ý tưởng chính của NodeJS là sử dụng môt hình non-blocking, event-driven để tạo ra các ứng dụng web thời gian thực nhẹ tải, hiệu suất cao đồng thời có thể chạy trên rất nhiều thiết bị khác nhau. NodeJS chỉ thực sự mạnh ở các ứng dụng cần tốc độ, khả năng mở rộng vì điểm mạnh của nó là khả năng xử lí một lượng rất lớn các connection với throughput cao, đồng nghĩa với khả năng mở rộng (scalability) là rất lớn. NodeJS dùng một thread duy nhất cùng với các câu lệnh I/O non-blocking cho phép nó phục vụ hàng chục ngàn connection cùng một lúc.

2. CÀI ĐẶT NODEJS TRÊN LOCAL

Xem video hướng dẫn: https://youtu.be/CcSuYLjKW3g

Sau khi cài xong thì kiểm tra trạng thái cài đặt bằng cách sử dụng Command Prompt gõ node -v kiểm tra version của NodeJS mà bạn đã cài đặt, nếu có kết quả trả về tức là bạn đã cài đặt thành công.



3. XỬ LÝ JSON TRONG NODEJS

3.1. Cấu trúc của chuỗi JSON

JSON là chữ viết tắt của Javascript Object Notation, đây là một dạng dữ liệu tuân theo một quy luật nhất định mà hầu hết các ngôn ngữ lập trình hiện nay đều có thể đọc được, bạn có thể sử dụng lưu nó vào một file, một record trong CSDL rất dễ dàng. JSON có định dạng đơn giản, dễ dàng sử dụng và truy vấn hơn XML rất nhiều nên tính ứng dụng của nó hiện nay rất là phổ biến.

Cấu trúc của chuỗi JSON: “Key”:”Value” . Giá trị mỗi key có thể là một chuỗi, một số hoặc là một chuỗi JSON khác nữa.

Ví dụ:

```
1. {
2. "username" : "node",
```

```
3. "email" : "node@gmail.com",
4. "title" : "Học lập trình với JSON"
5. }
```

Cấu trúc JSON của file package.json được truy xuất như sau:

```
1. obj_json.name; //lấy key name
2. obj_json.version; // lấy key version
3. obj_json.description; //lấy key description
```

3.2. Xử lý chuỗi JSON bằng Javacript

Chuyển JSON snag Object: *JSON.parse(json_string)*

Chuyển Object sang JSON: *JSON.stringify(json_object)*

4. NODE-PERSIST-LOCAL STORAGE

4.1. Modue node-persist

Node-persist là một module được xây dựng dành cho NodeJS, đây là Module có công dụng tương tự như LocalStorage trong HTML5 và Session trong PHP. Node-persist không sử dụng database để lưu trữ dữ liệu mà thay vào đó nó sẽ lưu vào một file trong hệ thống hoặc trong bộ nhớ với nội dung là chuỗi JSON hoặc file (có tên theo key). Vì dữ liệu lưu trữ trong bộ nhớ RAM hoặc ổ đĩa (disk) nên tốc độ xử lý dữ liệu lưu trữ của node-persist không kém phần lưu trữ trong database. Node-persist sử dụng phương thức localStorage trong HTML5 nên việc tiếp xúc nó rất dễ dàng.

4.2. Cài đặt node-persist

Mở cmd và nhập lệnh: *npm install node-persist*

4.3. Cách sử dụng node-persist

Sử dụng hàm require để tạo đối tượng module: var storage = require('node-persist');

Hàm khởi tạo

```
storage.initSync();
// hoặc
storage.init().then(promise);
```

Hàm khởi tạo này sẽ có một tham số truyền vào ở dạng Object, trong đó có nhiều key:

```
storage.init({
    dir : "path/to/save",
    ttl : false
});

storage.initSync({
    dir : "path/to/save",
    ttl : false
});
```

Trong đó key *dir* là đường dẫn lưu trữ dữ liệu. Còn *ttl* (time to live) là thời gian sống của dữ liệu, nếu thiết lập false thì nó sẽ sống vĩnh viễn.

Đối với hàm init() thì tất cả những phần xử lý có sử dụng các hàm của node-persist đều phải đặt trong promise, nếu không sẽ bị lỗi.

```
storage.init().then(function(){
    // Gọi các hàm setItem, getItem
});
```

Hàm get:

Hàm get dùng để lấy giá trị của một key nào đó, nếu key không tồn tại thì nó sẽ trả về undefined.

```
storage.getItem('domain');
// hoặc
storage.getItemSync('domain');
```

Hàm set: Dùng để thiết lập giá trị cho một key nào đó

```
storage.setItem('domain', 'name');
// hoặc
storage.setItemSync('domain',’name);
```

Hàm remove: dùng để xóa một key nào đó

```
storage.removeItem('domain');
// hoặc
storage.removeItemSync('domain');
```

Hàm clear: dùng để xóa tất cả các key trong bộ nhớ và ổ đĩa

```
storage.clear();
// hoặc
storage.clearSync();
```

5. MODULE, CALLBACK, EVENT, BUFFER TRONG NODEJS

5.1. MODULE

5.1.1. Module Yargs

Yargs là một module có tác dụng tách chuỗi của các request và lấy các tham số + giá trị của các tham số, module này rất hữu ích trong việc giao tiếp lấy dữ liệu từ client.

Cài đặt yargs: npm install yargs

Cách sử dụng Yargs căn bản: Truyền tham số dạng chỉ có value: Tất cả các tham số sẽ được lưu trữ trong thuộc tính argv của Yargs hoặc truyền tham số dạng key => value: Để truyền tham số dạng key => value thì ta sử dụng cú pháp sau:

```
node yargs-demo.js --key=value
hoặc
node yargs-demo.js --key=value
```

5.1.2. Module Yargs Options

Hàm Command trong Yargs: Dùng để thiết lập các cấu hình dành cho một action nào đó (Tham số truyền vào, kiểu dữ liệu của các tham số,...) và có thể sử dụng nhiều lần để thiết lập cho nhiều hành động

Cú pháp:

```
require('yargs').command('action', 'description', function(yargs){
    // Do something
});
```

Trong đó tham số thứ ba là một function và function này có một tham số truyền vào và đó cũng chính là đối tượng đại diện Yargs. Vì vậy sẽ dựa vào đối tượng này để cấu hình các Options.

Một số options hay sử dụng trong Yargs:

Xác định kiểu dữ liệu: *array*, *boolean*, *string*, *number*. Theo mặc định nếu không xác định kiểu dữ liệu và không truyền dữ liệu vào thì nó sẽ lấy kiểu boolean.

Sử dụng alias: Để tạo alias thì thêm key *alias: value* vào danh sách các options.

Lưu ý: Nếu alias chỉ có một ký tự thì ở lệnh chạy chỉ cần một dấu gạch ngang. Ví dụ nếu alias là u thì lúc chạy sẽ là -u chứ không phải là –u.

Xác định giá trị mặc định: *default: value*

5.1.3. Module CryptoJS

Cài đặt: *npm install crypto-js*

Quá trình mã hóa/ giải mã:

Link git: Link github: https://github.com/nguyenthingahd/do_an_co_so

Danh sách các loại mã hóa của Crypto:

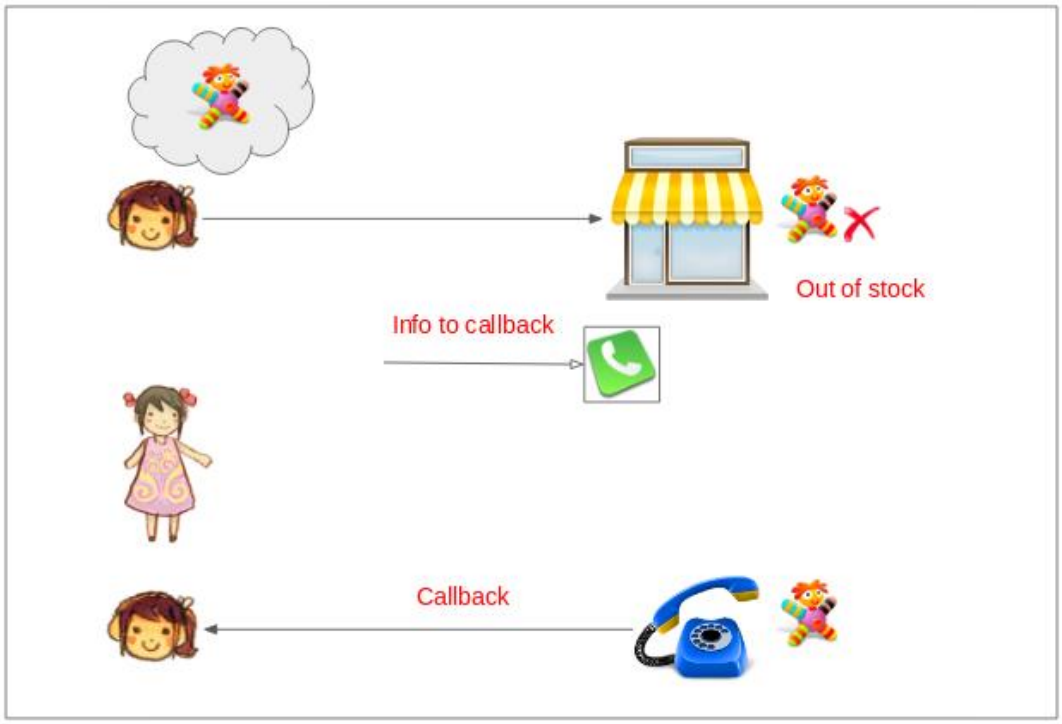
- crypto-js/AES
- crypto-js/core
- crypto-js/x64-core
- crypto-js/lib-typedarrays
- crypto-js/md5
- crypto-js/sha1
- crypto-js/sha256
- …

5.2. CALLBACK

Callback trong NodeJS tương tự như khái niệm về callback trong javascript (là một hàm sẽ được thực hiện sau khi một hàm khác đã thực hiện xong) vì NodeJS

sử dụng Google V8 JavaScript engine để thực thi mã và phần lớn các module được viết bằng javascript.

Ví dụ: Bạn tới một cửa hàng để mua một món đồ mà bạn yêu thích, nhân viên cửa hàng nói với bạn rằng hiện tại món đồ đó đã hết, bạn để lại số điện thoại và yêu cầu họ gọi lại ngay sau khi có hàng. Sau đó bạn có thể đi chơi hoặc làm một công việc nào đó và không cần quan tâm tới cửa hàng đó nữa, cho tới khi bạn nhận được điện thoại thông báo của hàng đã có món đồ mà bạn yêu thích.



Callback trong NodeJS

5.3. EVENTS

5.3.1. Một số khái niệm

Javascript Engine là chương trình hoặc trình thông dịch mã javascript. Một Javascript Engine có thể biên dịch như bình thường hoặc biên dịch just-in-time từ Javascript thành bytecode.

Heap Memory là vùng nhớ để chứa kết quả tạm thực thi các hàm trong stack, heap được cấp phát bộ nhớ càng cao thì chương trình chạy càng nhanh.

Stack là một cấu trúc dữ liệu, có nhiệm vụ ghi nhớ vị trí của các lệnh trong quá trình chạy. Vì Javascript là một ngôn ngữ đơn luồng nên nó chỉ có 1 stack.

Ngoài ra về stack còn có các vấn đề cần biết như :

- StackOverFlow : khi mà gọi một hàm đệ quy lặp đi lặp lại không kết thúc sẽ gây ra hiện tượng tràn stack.
- Stacktrace : Thứ tự, đường đi của các câu lệnh chứa trong stack

5.3.2. Event

Event loop là một vòng lặp vô tận có nhiệm vụ lắng nghe các event. Nhiệm vụ của nó là đọc các stack và event queue rồi đưa vào lại stack.

Event thao tác với các sự kiện trong NodeJS bằng cách require module event và khởi tạo một đối tượng mới là EventEmitter() (có chức năng chính là phát sinh sự kiện)

```
1. //Require module events
2. const event = require('events');
3. //Khởi tạo đối tượng eventEmitter
4. const eventEmitter = new event.EventEmitter();
```

Để thêm sự kiện mới chúng ta sử dụng hàm phương thức *on()* :

```
1. // Bind event and event handler as follows
2. eventEmitter.on(eventName, eventHandler);
```

Thực thi bằng phương thức:

```
1. eventEmitter.emit('eventName');
```

5.4. BUFFER

Global Objects là các object mà NodeJS cung cấp sẵn trong nhân của NodeJS và trong các module. Không cần phải khai báo hay include nó từ các thư viện khác mà chỉ cần gọi trực tiếp nó trong dự án. Các objects này có thể là một module, một hàm, object, string,...

Buffer là một trong những Global Objects trong NodeJS và nó sẽ được sử dụng mà không cần yêu cầu bất cứ module nào. Buffer để lưu trữ dữ liệu thô thành các mảng số nguyên tố tương ứng với việc cấp phát bộ nhớ heap V8 (vùng nhớ).

Ghi giá trị vào Buffer:

Cú pháp: *buffer.write(string[, offset][, length][, encoding])*

- string: chuỗi giá trị mà bạn muốn ghi, đây là tham số bắt buộc.
- offset: Vị trí bắt đầu mà bạn muốn ghi trong Buffer, mặc định sẽ là 0.
- length: Số bytes mà bạn muốn ghi lên, mặc định sẽ là độ dài của Buffer.
- encoding: kiểu mã hóa mà bạn muốn truyền vào, mặc định sẽ là "utf8".

Kiểu mã hóa mặc định là "utf-8", buffer hỗ trợ mã hóa theo nhiều kiểu khác nhau như : "ascii", "utf8", "utf16le", "ucs2", "base64", "hex",...

Đọc giá trị của Buffer:

Cú pháp: *buffer.toString([encoding][, start][, end])*

Phương thức toString() trong module Buffer cho phép chúng ta truyền vào 3 tham số và cả 3 tham số đều không bắt buộc.

- encoding: kiểu dữ liệu dùng để giải mã (mặc định là "utf8")
- start: vị trí bắt đầu đọc, mặc định là 0
- end: vị trí kết thúc đọc, mặc định sẽ là độ dài của buffer

Chuyển Buffer sang JSON

Cú pháp: *buffer.toJSON()*

Gộp nhiều Buffers với nhau

Cú pháp: *buffer.concat(list[, totalLength])*

Trong đó:

- list: một mảng các buffer cần gộp lại.
- totalLength: độ dài tối đa của buffers mới khi được gộp

Phương thức	Miêu tả
compare()	So sánh 2 Buffer
copy()	Sao chép Buffer
length	Trả về độ dài của Buffer, bytes
equal()	So sánh 2 buffers với nhau và trả về giá trị true hoặc false
fill()	Điền vào Buffer một giá trị cụ thể
from()	Tạo ra buffer mới từ buffer cũ
indexOf()	Kiểm tra xem buffer có chứa giá trị nào đó không ? Nếu có thì trả về vị trí của giá trị đó, ngược lại trả về -1
isBuffer()	Kiểm tra xem đối tượng đó có phải Buffer
isEncoding()	Kiểm tra xem Buffer object có kiểu mã hóa

5.5. TIMER FUNCTION

Set Timer function:

- setImmediate() : chạy ngay lập tức
- setTimeout() : chạy trong một khoảng thời gian.
- setInterval(): lặp đi lặp lại trong khoảng thời gian

Clear timer function:

● clearImmediate() : dừng một setImmediate objects, tạo bởi hàm setImmediate()

● clearTimeout() : dừng một setTimeout objects, tạo bởi hàm setTimeout()

● clearInterval() : dừng một setInterval objects, tạo bởi hàm setInterval()

5.6. STREAMS

Streams là một collections của dữ liệu giống như strings hay arrays, sự khác nhau duy nhất đó là các streams không tồn tại cùng một lúc do đó không cần chiếm nhiều bộ nhớ.

Streams cho phép đọc dữ liệu từ một nguồn hoặc viết dữ liệu đến một đích đến nào đó. Trong NodeJS, có 4 loại streams khác nhau: Readable, Writable, Duplex, Transform (giống Duplex nhưng khác nhau bởi kết quả của đầu ra dựa vào đầu vào). Mỗi streams đều được cấp một EventEmitter cho phép chúng ta bắt sự kiện theo từng thời điểm cụ thể (data, end, error, finish).

Streams cho phép đọc các dữ liệu lớn bằng cách chia nhỏ dữ liệu ra và đọc giá trị theo từng phần. Đọc dữ liệu với Streams: sử dụng phương thức *createReadStream()* tham số là đường dẫn file muốn đọc. Ghi dữ liệu với Streams: sử dụng phương thức *createWriteStream()* trong module file system.

Kỹ thuật Piping Stream trong NodeJS: Piping (đường ống) cho phép chúng ta lấy dữ liệu đầu ra từ một stream làm đầu vào trong streams khác. Nó hoạt động như một đường ống giúp chuyển dữ liệu giữa các streams với nhau. Và sử dụng phương thức *pipe()*.

Kỹ thuật Piping Chaining trong NodeJS: là kĩ thuật để kết nối đầu ra của các streams lại với nhau, nối đầu ra của streams này với streams khác tạo thành một chuỗi bao gồm nhiều các streams. Nó được sử dụng với cách hoạt động của piping. Dùng kỹ thuật này để lấy đầu ra của một file, nén nó lại sau đó tiến hành ghi file nén đó ra. Sử dụng thư viện zlib và phương thức *createGzip()* để nén file.

5.6. HTTP WEB SERVER TRONG NODEJS

Để khởi tạo một HTTP web server trong NodeJS chúng ta có thể dùng các thư viện bên thứ ba hoặc sử dụng module HTTP có sẵn trong NodeJS. HTTP module được NodeJS xây dựng cho phép Nodejs có thể truyền dữ liệu theo phương thức HTTP *(Hypertext Transport Protocol)*. Module HTTP là một nền tảng của NodeJS giúp khởi tạo để hỗ trợ các protocol phổ biến mà theo các thông thường rất khó để sử dụng, đặc biệt là các tin nhắn là dữ liệu lớn. Module HTTP được xây dựng để không chiếm quá nhiều vùng nhớ Buffer bằng cách stream data. Để sử dụng module này thì sử dụng khai báo: *require('http')*

Module HTTP có thể khởi tạo một web server cái mà lắng nghe trên port các yêu cầu của người dùng và phản hồi lại. Có thể tạo một web server trong module http bằng cách sử dụng phương thức createServer() :

```
1.  const http = require("http");
2.  //Sử dụng phương thức createServer
3.  http.createServer(function(req, res) {
4.     //Phản hồi của server
5.     res.write("Hello World !");
6.     //Kết thúc phản hồi
7.     res.end();
8.  }).listen(3000); //Sử dụng port 3000 để lắng nghe
```

Có thể tùy chỉnh header của HTTP server bằng cách thêm phương thức *writeHead()*:

```
1.  res.writeHead(statusCode, [header])
```

Trong đó:

- statusCode: mã phản hồi, trình duyệt sẽ dựa vào mã phản hồi để kiểm tra trạng thái của phản hồi.
- header: truyền vào một object chứa các giá trị header mà bạn muốn trả về.

Callback function của phương thức server.createServer() trả lại một tham số đó là *req*. Đây là một obejct chứa dữ liệu mà client gửi đến server. Nó có chứa một thuộc tính là req.url() chứa chuỗi truy vấn từ người dùng gửi đến:

```
• const http = require("http");
• //Sử dụng phương thức createServer
• http.createServer(function(req, res) {
•    //Thêm header vào trong response
```

```
    res.writeHead(200, {'Content-Type': 'text/html'});
    //Phản hồi của server
    res.write(req.url);
    //Kết thúc phản hồi
    res.end();
}).listen(3000); //Sử dụng port 3000 để lắng nghe
```

PHẦN II: CƠ SỞ DỮ LIỆU TRONG NODEJS

1. MYSQL

Cài đặt: *npm install mysql*

Để kết nối với MySql sử dụng cú pháp:

```
const mysql = require('mysql');
const con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword"
});
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
});
```

Thực hiện câu truy vấn:

```
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Result: " + result);
  });
});
```

2. MONGODB

MongoDB là một cơ sở dữ liệu tài liệu với khả năng mở rộng và linh hoạt với truy vấn và lập chỉ mục. Nó được sử dụng rất rộng rãi bởi khả năng phát triển ứng dụng dễ dàng hơn. Mô hình lưu trữ dữ liệu của MongoDB rất đơn giản để các nhà phát triển tìm hiểu và sử dụng, trong khi vẫn cung cấp tất cả các khả năng cần thiết để đáp ứng các yêu cầu phức tạp nhất ở mọi quy mô.

MongoDB lưu trữ dữ liệu trong các tài liệu linh hoạt, giống như JSON, có nghĩa là các trường có thể thay đổi từ tài liệu này sang tài liệu khác và cấu trúc dữ liệu có thể được thay đổi theo thời gian. Mô hình dữ liệu tương đồng với các object dữ liệu trong quá trình lập trình, giúp dữ liệu dễ dàng làm việc với các dữ liệu. Truy vấn đặc biệt, lập chỉ mục và tổng hợp thời gian thực cung cấp các cách mạnh mẽ để truy cập và phân tích dữ liệu. MongoDB là một cơ sở dữ liệu phân tán ở nhân của nó, vì vậy tính sẵn sàng cao, khả năng mở rộng cao (theo chiều ngang), có thể mổ rộng mô hình lưu trữ bằng cách thêm các máy chủ khác nhau ở nhiều địa điểm mà không phải nâng cấp phần cứng của server duy nhất như các hệ quản trị cơ sở dữ liệu MySQL.

Ưu điểm của MongoDB là: Dữ liệu lưu trữ phi cấu trúc, không có tính ràng buộc, toàn vẹn nên tính sẵn sàng cao, hiệu suất lớn và dễ dàng mở rộng lưu trữ. Dữ liệu được caching (ghi đệm) lên RAM, hạn chế truy cập vào ổ cứng nên tốc độ đọc và ghi cao.

Nhược điểm: Không ứng dụng được cho các mô hình giao dịch nào có yêu cầu độ chính xác cao do không có ràng buộc. Không có cơ chế transaction (giao dịch) để phục vụ các ứng dụng ngân hàng. Dữ liệu lấy RAM làm trọng tâm hoạt động vì vậy khi hoạt động yêu cầu một bộ nhớ RAM lớn. Mọi thay đổi về dữ liệu mặc định đều chưa được ghi xuống ổ cứng ngay lập tức vì vậy khả năng bị mất dữ liệu từ nguyên nhân mất điện đột xuất là rất cao.

Kết nối MongoDB:

```
1. var mongoose = require('mongoose');
2. mongoose.connect('mongodb://localhost:3000/laptop_store ', {useNewUrlParser: true, useNewUrlParser: true,  useUnifiedTopology: true});
```

Kiểm tra kết nối

```
1. var db = mongoose.connection;
2. db.on('error', console.error.bind(console, 'connection error:'));
3. db.once('open', function() {
4.   // we're connected!   });
```

PHẦN III: XÂY DỰNG ỨNG DỤNG

1. TÊN ĐỀ TÀI

Xây dựng website quản lý bán hàng

2. MÔ TẢ VẤN ĐỀ

Hiện nay trên thế giới thương mại điên tử đang phát triển rất mạnh mẽ. Kỹ thuật số giúp chúng ta tiết kiếm các chi phí nhờ chi phí vận chuyển trung gian, chi phí giao dịch và đặt biệt là giúp tiết kiệm thời gian để con người đầu tư vào các hoạt động khác. Hơn nữa thương mại điện tử còn giúp con người có thể tìm kiếm tự động theo nhiều mục đích khách nhau, tự động cung cấp thông tin theo nhu cầu và sở thích của con người. Giờ đây, con người có thể

ngồi tại nhà để mua sắm mọi thứ theo ý muốn và các website bán hàng trên mạng sẽ giúp ta làm được điều đó. Chính vì vậy các công nghệ mã nguồn mở trở lên được chú ý vì các tính năng của nó. Giá thành rẻ và được hỗ trợ rất nhiều trên mạng sẽ giúp ta nhanh chóng xây dựng các website bán hàng thân thiện và dễ sử dụng với người dùng. Chính vì vậy trong đồ án này em chọn đề tài về " Xây dựng website bán hàng" sử dụng NodeJS kết hợp với hệ quản trị cơ sở dữ liệu MySql.

3. DEMO

Link github: https://github.com/nguyenthingahd/do_an_co_so

PHẦN IV: KẾT LUẬN

1. KẾT LUẬN

- Tìm hiểu tổng quan về NodeJS

- Xây dựng được ứng dụng cơ bản sử dụng NodeJS kết hợp với cở sở dữ liệu MySQL.

2. HƯỚNG PHÁT TRIỂN

- Tìm hiểu sâu những framework của NodeJS

- Phát triển ứng dụng có nhiều chức năng hơn

3. SOURCE CODE

Link github: https://github.com/nguyenthingahd/do_an_co_so