Bài 1
Tổng quan NodeJS

Module: Web Backend Development with NodeJS

Mục tiêu

- Tìm hiểu tổng quan về NodeJS.
- Cài đặt thành công NodeJS và NPM.
- Tìm hiểu về module trong nodeJS.
- Phân biệt được Blocking và Non-Blocking.
- Debug được trong môi trường NodeJS.
- Phân biệt được Javascript trong trình duyệt và trong NodeJS.

Thảo luận

NodeJS là gì?

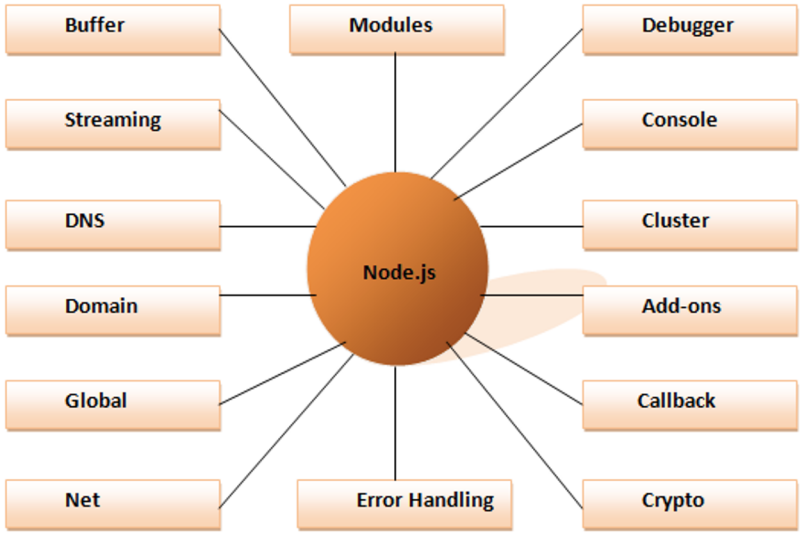
NodeJS là gì?

- Node.js là một platform và môi trường thời gian chạy đa nền tảng để chạy các ứng dụng JavaScript bên ngoài trình duyệt. Nó được sử dụng để tạo các ứng dụng web phía máy chủ.
- Nhiều mô-đun cơ bản của Node.js được viết bằng JavaScript. Node.js chủ yếu được sử dụng để chạy các ứng dụng máy chủ thời gian thực (real-time server applications).
- Node.js sử dụng mô hình I/O hướng sự kiện, hoàn hảo cho các ứng dụng thời gian thực sử dụng nhiều dữ liệu chạy trên các thiết bị phân tán.
- Node.js cũng cung cấp một thư viện phong phú gồm các mô-đun JavaScript khác nhau để đơn giản hóa việc phát triển các ứng dụng web.

Sơ đồ thành phần của NodeJS

> Node.js = Runtime Environment + JavaScript Library

Sơ đồ sau chỉ định một số phần quan trọng của Node.js:



Các tính năng của NodeJS

- Cực nhanh: Node.js được xây dựng trên Công cụ JavaScript V8 của Google Chrome, do đó, thư viện của nó thực thi mã rất nhanh.
- I/O là không đồng bộ và theo hướng sự kiện: Tất cả các API của thư viện Node.js là không đồng bộ, tức là không chặn (non-blocking). Vì vậy, máy chủ dựa trên Node.js không bao giờ đợi API trả về dữ liệu. Máy chủ chuyển đến API tiếp theo sau khi gọi nó và cơ chế thông báo Sự kiện của Node.js giúp máy chủ nhận được phản hồi từ lệnh gọi API trước đó. Nó cũng là một lý do mà nó là rất nhanh.

- Một luồng: Node.js tuân theo một mô hình một luồng với vòng lặp sự kiện.
- Khả năng mở rộng cao: Node.js có khả năng mở rộng cao vì cơ chế sự kiện giúp máy chủ phản hồi theo cách không bị chặn(non-blocking).
- Không có bộ đệm: Node.js cắt giảm thời gian xử lý tổng thể trong khi tải lên các tệp âm thanh và video. Các ứng dụng Node.js không bao giờ đệm bất kỳ dữ liệu nào. Các ứng dụng này chỉ đơn giản xuất dữ liệu theo từng phần.
- Mã nguồn mở: Node.js có một cộng đồng mã nguồn mở đã tạo ra nhiều mô-đun tuyệt vời để bổ sung cho các ứng dụng Node.js.
- Giấy phép: Node.js được phát hành theo giấy phép MIT.

Ứng dụng viết bằng NodeJS

- Websocket server: Các máy chủ web socket như là Online Chat, Game Server...
- Fast File Upload Client: là các chương trình upload file tốc độ cao.
- Ad Server: Các máy chủ quảng cáo.
- Cloud Services: Các dịch vụ đám mây.
- RESTful API: đây là những ứng dụng mà được sử dụng cho các ứng dụng khác thông qua API.
- Any Real-time Data Application: bất kỳ một ứng dụng nào có yêu cầu về tốc độ thời gian thực. Micro Services: Ý tưởng của microservices là chia nhỏ một ứng dụng lớn thành các dịch vụ nhỏ và kết nối chúng lại với nhau. Nodejs có thể làm tốt điều này.

NodeJS Modules

Thảo luận.

NodeJS module là gì?

Khái niệm NodeJS module.

Trong Node.js,Module là các khối mã đóng gói giao tiếp với ứng dụng bên ngoài trên cơ sở chức năng liên quan của chúng. Mô-đun có thể là một tệp đơn hoặc một tập hợp nhiều tệp/thư mục. Lý do mà các lập trình viên phụ thuộc nhiều vào các mô-đun là vì khả năng tái sử dụng của chúng cũng như khả năng chia nhỏ một đoạn mã phức tạp thành các đoạn có thể quản lý được.

Các loại Module trong NodeJS

Mô-đun có 3 loại:

- Core Modules
- local Modules
- Third-party Modules

Core module.

Core Modules: Node.js có nhiều mô-đun tích hợp là một phần của nền tảng và đi kèm với cài đặt Node.js. Các mô-đun này có thể được tải vào chương trình bằng cách sử dụng hàm require.

Cú pháp:

```
const module = require('module_name');
```

Ví dụ: Tạo máy chủ web bằng module http.

```
const http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write('Welcome to this page!');
  res.end();
}).listen(3000);
```

Local module.

- Local Modules: Mô-đun cục bộ được tạo cục bộ trong ứng dụng Node.js của bạn.
- Local modules được cung cấp các thuộc tính cho bên ngoài thông qua export hoặc module.export.
- Các tệp khác có thể sử dụng chức năng của local module thông qua hàm require().

Third-party Modules

Third-party Modules: là các mô-đun trực tuyến có sẵn bằng cách sử dụng Node Package Manager(NPM). Các mô-đun này có thể được cài đặt trong thư mục dự án. Một số mô-đun third-party phổ biến là mongoose, express, angular, và react

Đây chính là modules làm nên tên tuổi của NodeJS. Nó có cộng đồng và sự support rộng lớn.

Ví dụ:

- npm install express
- npm install mongoose
- npm install -g @angular/cli

Blocking và Non-Blocking trong NodeJS

Blocking

Blocking: Đề cập đến việc ngăn chặn hoạt động tiếp theo cho đến khi hoạt động hiện tại kết thúc. Các phương thức chặn được thực thi đồng bộ. Đồng bộ có nghĩa là chương trình được thực thi từng dòng một. Chương trình đợi cho đến khi hàm được gọi hoặc trả về.

Ví dụ:

```
function(){
    console.log('A')
    const a = () => {
        console.log('B')
    }
         a();
    console.log('C')
})()
```

Output:

```
A
B
C
```

Non-Blocking

Non-Blocking: Đề cập đến chương trình không chặn việc thực hiện các hoạt động tiếp theo. Các phương thức Non-Blocking được thực thi không đồng bộ. Không đồng bộ có nghĩa là chương trình có thể không nhất thiết phải thực thi từng dòng một. Chương trình gọi hàm và chuyển sang thao tác tiếp theo và không đợi nó trả về.

Ví dụ:

```
(function(){
    console.log('A')
    const a = () => {
        console.log('B')
    }
    setTimeout(a,1000)
    console.log('C')
})()
```

Output:

A

C

B

Demo

Demo về blocking và bon-blocking trong nodeJs.

NodeJS và Brower.

Thảo luận.

NodeJS và Brower giống và khác nhau như thế nào?

Khái niệm.

- Nodejs là một nền tảng (Platform) phát triển độc lập được xây dựng ở trên Javascript Runtime của Chrome mà chúng ta có thể xây dựng được các ứng dụng mạng một cách nhanh chóng và dễ dàng mở rộng.
- Browser hay còn gọi là trình duyệt web, bất cứ phần mềm nào cho phép bạn truy cập website, biên dịch được mã HTML, CSS, Javascript và cho phép lưu cookie thì được gọi là Browser.

Giống nhau

- Cả hai đều là môi trường để thực thi code Javascript. nhưng node thực hiện việc đó ở phía máy chủ (server), còn trình duyệt thực hiện ở phía client. Và cả hai đều cùng sử dụng Javascript engine.

Khác nhau

- Ở Browser, bạn có thể tương tác với DOM hoặc các API (localStorage, sessionStorage, ...). Tuy nhiên những thứ đó không tồn tại ở Node.
- Bên cạnh đó Node cũng không có các object “document” và “window” và những đối tượng khác có sẵn trong Browser.

```
var name = 'Code Learn'
console.log(window.name);
```

Đoạn code trên sẽ báo lỗi window is not defiend nếu chạy owe Node.

- Browser không có các API mà Node Js cung cấp trong npm (node package manager).
- Ở NodeJs, bạn có thể kiểm soát được môi trường khi phát triển, và bạn có thể chọn version cho NodeJs. Ở môi trường Browser thì bạn không có nhiều quyền lựa chọn để sử dụng.
- Với NodeJs, bạn có thể sử dụng từ khóa “require” để sử dụng các module nhưng ở Browser thì không. Khi bạn sử dụng “require” trong Browser thì trình duyệt sẽ báo lỗi.
- Browser có giao diện đồ họa còn Node thì không.

- Ở Node, bạn có toàn quyền truy cập hệ thống người dùng. Không giống Browser, NodeJs có thể truy cập hệ thống của bạn như các ứng dụng khác. Có nghĩa là bạn có thể đọc và ghi trực tiếp từ file system hoặc là thực thi những phần mềm khác.
- Bên cạnh đó bạn còn có thể viết một ứng dụng desktop hoàn toàn có thể với NodeJs bao gồm UI thông qua các modules.
- Một điểm khác biệt nữa là Node.js sử dụng hệ thống module CommonJS, trong khi trong trình duyệt, chúng ta bắt đầu thấy tiêu chuẩn ES modules đang được triển khai.