1. Giới thiệu

Node.js là gì?

NodeJS là một nền tảng Server side được xây dựng dựa trên Javascript Engine (V8 Engine). Node.js được phát triển bởi Ryan Dahl năm 2009. NodeJs được định nghĩa như sau:

Node.js là một nền tảng (platform) dựa vào Chrome Javascript runtime để xây dựng các ứng dụng nhanh, có độ lớn. Node.js sử dụng các thành phần hướng sự kiện (event-driven), mô hình non-blocking I/O để tạo ra các ứng dụng nhẹ và hiệu quả cho các ứng dụng về dữ liệu thời gian thực chạy trên các thiết bị phân tán.

Node.js = Môi trường Runtime + Các thư viện Javascript

NodeJs là một platform mã nguồn mở, đa nền tảng cho phát triển các ứng dụng phía Server và các ứng dụng liên quan đến mạng. Ứng dụng Node.js được viết bằng Javascript và có thể chạy trong môi trường Node.js trên hệ điều hành Window, Linux...

Node.js cũng cung cấp cho chúng ta các module Javascript đa dạng, có thể đơn giản hóa sự phát triển của các ứng dụng web sử dụng Node.js với các phần mở rộng.

Đặc điểm của Node.js

Dưới đây là vài đặc điểm quan trọng biến Node.js trở thành sự lựa chọn hàng đầu trong phát triển phần mềm:

* Không đồng bộ và hướng sự kiện (Event Driven): Tất các các APIs của thư viện Node.js đều không đồng bộ, nghĩa là không blocking (khóa). Nó rất cần thiết vì Node.js không bao giờ đợi một API trả về dự liệu. Server chuyển sang một API sau khi gọi nó và có cơ chế thông báo về sự kiện của Node.js giúp Server nhận được phản hồi từ các API gọi trước đó.
* Chạy rất nhanh: Dựa trên V8 Javascript Engine của Google Chrome, thư viện Node.js rất nhanh trong các quá trình thực hiện code.
* Các tiến trình đơn giản nhưng hiệu năng cao: Node.js sử dụng một mô hình luồng đơn (single thread) với các sự kiện lặp. Các cơ chế sự kiện giúp Server trả lại các phản hồi với một cách không khóa và tạo cho Server hiệu quả cao ngược lại với các cách truyền thống tạo ra một số lượng luồng hữu hạn để quản lý request. Nodejs sử dụng các chương trình đơn luồng và các chương trình này cung cấp các dịch vụ cho số lượng request nhiều hơn so với các Server truyền thống như Apache HTTP Server.
* Không đệm: Ứng dụng Node.js không lưu trữ các dữ liệu buffer.
* Có giấy phép: Node.js được phát hành dựa vào MIT License.

Với Node.js, bạn phải làm mọi thứ

Node.js chỉ là một môi trường – điều này có nghĩa bạn tự phải làm mọi thứ. Sẽ chẳng có bất kỳ máy chủ mặc định nào cả. Một đoạn script xử lý tất cả các kết nối với Client. Điều này làm giảm đáng kể số lượng tài nguyên được sử dụng trong ứng dụng.

Các thành phần quan trọng trong Node.js

Hình ảnh dưới đây mô tả các thành phần (component) quan trọng của Node.js mà chúng ta sẽ học ở các bài tiếp theo.



Khi nào sử dụng Node.js

Dưới đây là các lĩnh vực mà Node.js được sử dụng như là một sự lựa chọn hoàn hảo:

* Các ứng dụng về I/O
* Các ứng dựng về luồng dữ liệu
* Các ứng dụng về dữ liệu hướng đến thời gian thực
* Các ứng dụng dựa vào JSON APIs
* Các ứng dụng Single Page Application

Khi nào không nên dùng Node.js

Nó không nên sử dụng trong các ứng dụng đòi hỏi về CPU.

Ứng dụng hello world trong nodejs

Trước khi tạo ứng dụng "Hello world" trong Node.js, hãy cùng xem các phần chính của chương trình Node.js. Một chương trình Node.js bao gồm các phần quan trọng đưới đây:

* Các Module cần thiết: Chúng ta sử dụng require directive để tải một Node.js module.
* Tạo Server: Một Server lắng nghe yêu cầu từ phía client tương tự như các Server Apache HTTP.
* Đọc request và trả về các response: Server được tạo một các dễ dàng bên trên sẽ đọc các HTTP request bởi client từ các trình duyệt hay màn hình console để trả về các phản hồi.

## Tạo ứng dụng Node.js

Bước 1: Import các Module cần thiết

Chúng ta sử dụng require directive để tải các Module http và trả về các sự thể hiện đến biến http như sau:

|  |  |
| --- | --- |
|  | var http = require("http"); |

Bước 2: Tạo Server

Bước tiếp theo, chúng ta sẽ tạo http và gọi phương thức http.createServer() để tạo mới một Server và trả về một Server Instance và sau đó gắn kết nó trên cổng 8081. Truyền vào nó với các tham số request và response. Viết đoạn ví dụ sau đây về chương trình "Hello World".

http.createServer(function (request, response) {

response.writeHead(200, {'Content-Type': 'text/plain'});

response.end('Hello World\n');

}).listen(8080, "127.0.0.1");

Đoạn code này đủ cho việc tạo một HTTP Server để lắng nghe và đợi các phản hồi trên 8081 của máy local.

Bước 3: Kiểm tra Request & Response

Cùng đặt bước 1 và 2 với nhau trong 1 file với tên gọi main.js và bật HTTP Server như sau:

console.log('Server running at http://127.0.0.1:80801');

Start Server và xem kết quả như sau:

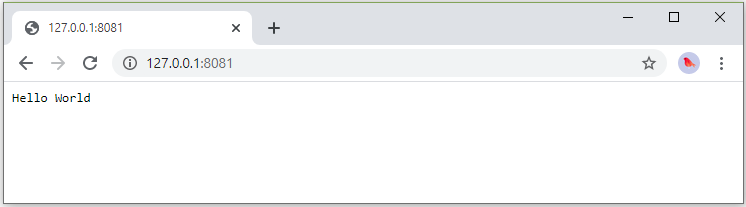
$ node main.js

Kiểm tra kết quả. Server đã được bật.

Server running at <http://127.0.0.1:8081/>

## Tạo một request đến một Node.js Server

Mở http://127.0.0.1:8080/ trong bất kỳ trình duyệt nào và xem kết quả:.



Chúc mừng bạn đã tạo thành công ứng dụng Node.js đầu tiên. Theo dõi tiếp các chương tiếp theo để hiểu sâu hơn về Node.js nhé.

## REPL là gì?

REPL là từ viết tắt của Read Eval Print Loop (Đọc - Đánh giá - In - Lặp) và nó biểu diễn môi trường máy tính như màn hình console trong Linux shell nơi bạn có thể gõ các dòng lệnh và hệ thống sẽ trả về các kết quả. Node.js cũng có môi trường REPL. Nó để thực hiện các tác vụ mong muốn:

* Read: Đọc các thông tin input của người dùng, chuyển đổi thành các dữ liệu Javascript và lưu trữ trong bộ nhớ.
* Eval: Đánh giá các cấu trúc dữ liệu này.
* Print: In các kết quả.
* Loop: Lặp các dòng lệnh đến khi người dùng gõ ctrl-c hai lần.

Đặc tính REPL của Node js rất hữu ích khi bạn sử dụng Node.js với mục đích debug code.

## Bắt đầu với REPL trong Node.js

REPL có thể bắt đầu bằng cách đơn giản trên màn hình shell/console mà không sử dụng bất cứ tham số nào như sau:

> node

Bạn sẽ thấy màn hình dòng nhắc lệnh REPL >. Tại đây, bạn có thể gõ bất cứ lệnh nào trong Node.js.

>

### Các biểu thức đơn giản trong Node.js

Dưới đây mình giới thiệu một số biểu thức đơn giản có thể được sử dụng tại dòng nhắc lệnh REPL trong Node.js:

$ node

> 1 + 3

4

> 1 + ( 2 \* 3 ) - 4

3

>

### Sử dụng biến trong Node.js

Bạn có thể sử dụng các biến để lưu trữ giá trị và sau đó in giá trị của biến giống như trong các đoạn mã truyền thống. Nếu không sử dụng từ khóa var, giá trị sẽ được lưu trữ trong biến và in ra. Trong khi đó nếu từ khóa var được sử dụng, giá trị được lưu trữ và có thể không được in. Bạn có thể in biến sử dụng console.log().

$ node

> x = 10

10

> var y = 10

undefined

> x + y

20

> console.log("Hello World")

Hello World

undefined

### Biểu thức trên nhiều dòng trong Node.js

Node REPL hỗ trợ dạng biểu thức nằm trên nhiều dòng tương tự như trong Javascript. Cùng kiểm tra biểu thức do-while trong hành động sau đây:

Ví dụ:

$ node

> var x = 0

undefined

> do {

... x++;

... console.log("x: " + x);

... }

while ( x < 5 );

x: 1

x: 2

x: 3

x: 4

x: 5

undefined

>

Các dấu gạch chấm ... hiển thị tự động khi bạn nhấn phím Enter sau dấu ngoặc mở. Node.js sẽ tự động kiểm tra xem biểu thức có được tiếp tục nữa hay không.

### Biến có dấu gạch dưới trong Node.js

Bạn có thể sử dụng dấu gạch dưới \_ để nhận kết quả cuối cùng của phép tính:

Output:

$ node

> var x = 10

undefined

> var y = 20

undefined

> x + y

30

> var sum = \_

undefined

> console.log(sum)

30

undefined

>

## Giới thiệu một số lệnh REPL trong Node.js

* ctrl + c - Kết thúc lệnh hiện tại.
* Nhấn ctrl + c hai lần - Kết thúc Node REPL.
* ctrl + d - Kết thúc Node REPL.
* Các phím mũi tên Up/Down - Xem lịch sử các lệnh, kiểm tra lệnh trước đó và có thể sửa đổi các lệnh đã soạn trước đó.
* Phím tab - Danh sách các lệnh hiện tại.
* .help - Danh sách tất cả các lệnh.
* .break - Thoát khỏi một biểu thức nằm trên nhiều dòng (ví dụ: do-while).
* .clear - Thoát khỏi một biểu thức nằm trên nhiều dòng
* .save ten\_file - Lưu phiên làm việc của Node.js REPL hiện tại vào một ten\_file nào đó.
* .load ten\_file - Tải nội dung của ten\_file của phiên làm việc Node.js REPL hiện tại.

## Kết thúc REPL trong Node.js

Như đã đề cập bên trên, bạn cần nhấn ctr + c 2 lần để kết thúc Node.js REPL.

Output:

> node

>

(^C again to quit)

>

# **NPM trong Node.js**

## NPM là gì?

NPM viết tắt của Node package manager là một công cụ tạo và quản lý các thư viện lập trình Javascript cho Node.js.

Trong các dự án với javascript thì chắc hẳn các bạn sẽ cần nhiều thư viện của javascript. Điển hình như jquery, bootstrap, express, vue, react, angular, ... thì việc quản lý các thư viện này như thế nào, dùng phiên bản bao nhiêu, cần require thêm những module gì. Những công việc này nếu làm thủ công thì quả là cực, đặc biệt là những dự án lớn.

NPM được cài đặt cùng với Node.js từ phiên bản v0.6.3. Để xác nhận việc này, mở cửa sổ dòng lệnh, gõ những lệnh sau và xem kết quả:

Kiểm tra phiên bản npm: npm --version

$ npm --version

2.7.1

Có thể dễ dàng cập nhật bản mới nhất. Sử dụng lệnh sau đây dưới quyền Administrator: npm install npm –g

## Cài đặt các Module bởi sử dụng npm trong Node.js

Dưới đây là cú pháp đơn giản để cài đặt bất kỳ Node.js Module nào:

> npm install [Module Name]

Ví dụ: lệnh để cài đặt một module rất phổ biến trong Node.js, đó là **express** Module:

> npm install express

Để sử dụng module express trong file js của bạn, sử dụng cú pháp sau đây:

var express = require('express');

## Cài đặt Global và cài đặt local

Mặc định, npm cài đặt bất kì phụ thuộc nào trên máy tính local. Ở đây, module này hướng đến các gói cài đặt trong thư mục node\_modules trong thư mục ứng dụng của Node. Gói local này có thể triển khai dựa vào phương thức require(). Ví dụ, khi bạn cài đặt Module express, bạn tạo thư mục node\_modules trong thư mục hiện tại nơi bạn có thể cài đặt Module này.

Cách khác, bạn có thể sử dụng lệnh npm ls để liệt kê các thư mục Module cài đặt.

Gói cài đặt toàn cục được lưu trữ trong các file hệ thống. Các phụ thuộc có thể được sử dụng với các hàm CLI (Command Line Interface) của node.js nhưng không thể import sử dụng require() của ứng dụng Node một cách trực tiếp.

Bây giờ cùng thử cài đặt Module express sử dụng cài đặt toàn cục như sau: npm install express -g

> npm install express -g

Điều này sẽ cho kết quả tương tự nhưng Module được cài đặt toàn cục. Ở đây, dòng đầu tiên nói với chúng ta về Module version và nơi chứa các bộ cài khi bắt đầu.

Bạn cũng có thể sử dụng lệnh sau đây để kiểm tra các phiên bản cài đặt toàn cục: npm ls -g

> npm ls -g

## Sử dụng gói package.json

Package.json hiển thị trong thư mục root của bất cứ ứng dụng Node nào và được sử dụng để định nghĩa thuộc tính của gói. Tập tin package.json của gói express hiển thị trong **node\_modules/express/**:

## Các thuộc tính của Package.json

* **name** - tên của package
* **version** - phiên bản của package
* **description** - phần mô tả của package
* **homepage** - trang chủ của package
* **author** - tác giả của package
* **contributors** - tên người đóng góp cho package
* **dependencies** - Danh sách các gói phụ thuộc, tự động được cài theo.
* **repository** - loại repository và url của package
* **main** - điểm vào của package
* **keywords** - các từ khóa

Gỡ cài đặt một module trong Node.js

Sử dụng các dòng lệnh dưới đây để gỡ một Module trong Node.js

> npm uninstall express

Khi gỡ cài đặt gói này, bạn có thể xác nhận nội dung trong thư mục /node\_modules/ bằng cách sử dụng lệnh sau đây:

> npm ls

Cập nhật một Module trong Node.js

Cập nhật gói package.json và thay đổi phiên bản của các ràng buộc mà có thể cập nhật sử dụng lệnh sau đây:

> npm update express

ìm kiếm một Module trong Node.js

Sử dụng npm, bạn có thể tìm kiếm một Module như sau:

> npm search express

Tạo một Module trong Node.js

Quá trình tạo Module yêu cầu gói package.json được tạo ra.

> npm init

This utility will walk you through creating a package.json file.

It only covers the most common items, and tries to guess sane defaults.See 'npm help json' for definitive documentation on these fields

and exactly what they do.Use 'npm install --save' afterwards to install a package and

save it as a dependency in the package.json file.Press ^C at any time to quit.

name: (webmaster)

Bạn cần cung cấp tất cả các thông tin được yêu cầu trong việc sử dụng Module. Bạn có thể sử dụng các bước bên trên trong file package.json để hiểu về ý nghĩa của các thông tin yêu cầu. Khi gói package.json được tạo ra, sử dụng các lệnh dưới đây đăng ký với trang npm bởi sử dụng địa chỉ email hợp lệ.

> npm adduser

Username: mcmohd

Password:

Email: (this IS public) mcmohd@gmail.com

Bây giờ publish Module của bạn:

> npm publish

Nếu mọi thứ ok, nó sẽ được publish và thư mục này sẽ được truy cập bởi những người khác sử dụng Module Node.js