NodeJS From Scratch

開發指南

Release 1.0

by NodeJS Taiwan

October 04, 2011

CONTENTS

CHAPTER

ONE

NODEJS TAIWAN 開源電子書計畫

這份電子書是由 NodeJS Taiwan 開發社群共同製作

我們的 GitHub 網址是： https://github.com/nodejs-tw/nodejs-from-scratch

您可以利用 Google Doc Viewer 預覽本電子書最新版： http://goo.gl/Y7nSh

使用 ContPub 服務打包： http://contpub.org/

1.1 如何加入 NodeJS Taiwan 開源書寫計畫？

1. 請先申請一組 GitHub 帳號
2. 請瀏覽本專案於 GitHub 的網頁 https://github.com/nodejs-tw/nodejs-from-scratch
3. 您的系統必須裝有 Git 軟體
4. 本專案的 Repo 位址是 git@github.com:nodejs-tw/nodejs-from-scratch.git

1.2 如何使用 reStructuredText 格式排版文字內容？

本書採用 Sphinx 擴充的 reStructuredText 格式撰寫。

http://sphinx.pocoo.org/rest.html

如果您有興趣加入內容寫作小組，只需要參考以下的語法說明。

本書的架構以「章」為單位切割檔案，副檔名一律採用.rst 結尾。例如 nodejs_intro.rst 、nodejs_hello.rst 分別是兩個不同 Chapter，在 index.rst 主文件中，會依次序連結到這些檔案（但定義時不含副檔名）。

每一章的.rst 檔案會有以下的文件結構：

```
這是大標題（章）
===============

這是小標題（節）
----------------

這是更小標題（子節）
^^^^^^^^^^^^^^^^^^^^
```

內文的部份，為了方便編輯，可以在在合適的文字長度之後換行，連續換行兩次則會建立新的段落。

```
這是第一段落，
第二行還是同一個段落；
第三行也是。

隔一個空行，
就會產生一個新的段落，
到這裡還是第二個段落。
```

如果要顯示一行或多行指令，例如 Shell 命令，則使用兩個冒號，隔一行之後再以 Tab 起始。

```
執行一個 NodeJS 程式檔。 ::

        node example.js
```

請注意兩個冒號前面若有其它文字，必須插入「空白」隔開。

如果要顯示程式碼，有兩種寫法。

第一種，直接寫在文件裡面（適合片段、部分節錄）：

```
.. code-block:: javascript

        function say_hello(who) {
                return 'Hello, ' + who + '!';
        }
```

第二種，寫在外部程式碼檔案（適合大型、可執行程式）：

```
.. literalinclude:: src/example.py
   :language: javascript
```

為了方便測試程式碼，如果是可被執行的完整程式，請放在 src 資料夾下，再使用第二種方法嵌入文件中。

CHAPTER

TWO

NODEJS 介紹

NodeJS 是一個高效能、易擴充的網站應用程式開發框架 (Web Application Framework) 。它誕生的原因，是為了讓開發者能夠更容易開發高延展性的網路服務，不需要經過太多複雜的調校、效能調整及程式修改，就能滿足網路服務在不同發展階段對效能的要求。

Ryan Dahl 是 NodeJS 的催生者，目前任職於 Joyent (主機託管服務公司) 。他開發 NodeJS 的目的，就是希望能解決 Apache 在連線數量過高時，緩衝區 (buffer) 和系統資源會很快被耗盡的問題，希望能建立一個新的開發框架以解決這個問題。因此嘗試使用效能十分優秀的 V8 JavaScript Engine ，讓網站開發人員熟悉的 JavaScript 程式語言，也能應用於後端服務程式的開發，並且具有出色的執行效能。

JavaScript 是功能強大的物件導向程式語言，但是在 JavaScript 的官方規格中，主要是定義網頁 (以瀏覽器為基礎) 應用程式需要的應用程式介面 (API) ，對 JavaScript 程式的應用範圍有所侷限。為使 JavaScript 能夠在更多用途發展，CommonJS 規範一組標準函式庫 (standard library) ，使 JavaScript 的應用範圍能夠和 Ruby 、Python 及 Java 等語言同樣豐富。撰寫 NodeJS 的 JavaScript 程式碼，符合 CommonJS 規範，可以使用 CommonJS API 為基礎開發程式，並且在不同的 CommonJS 兼容 (compliant) JavaScript 執行環境中，程式碼具有可攜性。

瀏覽器的 JavaScript 與實現 CommonJS 規範的 NodeJS 有何不同呢？瀏覽器的 JavaScript 提供 XMLHttpRequest ，讓程式可以和網頁伺服器建立資料傳輸連線，但這通常只能適用於網站開發的需求，因為我們只能用 XMLHttpRequest 與網頁伺服器通訊，卻無法利用它建立其他類型如 Telnet / FTP / NTP 的伺服器通訊。如果我們想開發網路服務程式，例如 SMTP 電子郵件伺服器，就必須使用 Sockets 建立 TCP (某些服務則用 UDP) 監聽及連線，其他程式語言如 PHP 、Java 、Python 、Perl 及 Ruby 等，在標準開發環境中皆有提供 Sockets API ，而瀏覽器的 JavaScript 基於安全及貼近網站設計需求的考量下，並未將 Sockets 列入標準函式庫之中。CommonJS 的規範填補這種基礎函式庫功能的空缺，遵循 CommonJS 規範的 NodeJS 可以直接使用 Sockets API 建立各種網路服務程式。

JavaScript 語言本身支援 Lambda 的特性，因此一個匿名函式 (anonymous function) 可以

被儲存成一個變數，並當作參數傳遞給另一個函式。

```
var proc1 = function(op, x) {
    return op(x);
}

var op1 = function(x) { return x+1; }
var op2 = function(x) { return x*x; }

proc1(op1, 3);  // result is 4
proc1(op2, 5);  // result is 25
```

另一個 JavaScript 開發者必須掌握的語言特性稱為 Closure 。

Callback 這部份需要在查文件確認 NodeJS 符合 CommonJS 的規範，使得 Callback 方式易於實現，也能夠讓更多同好基於 JavaScript 開發符合 NodeJS 的外掛模組 (Module)。

回想以前要寫一個能夠同時容納上百人的上線的網路服務，需要花費多大的苦工，可能 10 人多就需要經過一次程式調整，而 NodeJS 就是為了解決這個困境，NodeJS 因此誕生，它是一種利用 V8 Javascript 編譯器，所開發的產品，利用 V8 編譯器的高效能，與 Javascript 的程式開發特性所產生的網路程式。

開發人員所編寫出來的 Javascript 腳本程式，怎麼可能會比其他語言寫出來的網路程式還要快上許多呢？以前的網路程式原理是將使用者每次的連線 (connection) 都開啟一個執行緒 (thread)，當連線爆增的時候將會快速耗盡系統效能，並且容易產生阻塞 (block) 的發生。

NodeJS 對於資源的調校有所不同，當程式接收到一筆連線 (connection)，會通知作業系統透過 epoll, kqueue, /dev/poll, 或 select 將連線保留，並且放入 heap 中配置，先讓連線進入休眠 (Sleep) 狀態，當系統通知時才會觸發連線的 callback。這種處理連線方式只會佔用掉記憶體，並不會使用到 CPU 資源。另外因為採用 Javascript 語言的特性，每個 request 都會有一個 callback，如此可以避免發生 Block 的狀況發生。

基於 Callback 特性，目前 NodeJS 大多應用於 Comet(long pulling) Request Server，或者是高連線數量的網路服務上，目前也有許多公司將 NodeJS 設為內部核心網路服務之一。在 NodeJS 也提供了外掛管理 (Node package management)，讓愛好 NodeJS 輕易開發更多有趣的服務、外掛，並且提供到 npm 讓全世界使用者快速安裝使用。

相關連結：NodeJS 官方網站 NodeJS 官方部落格 NodeJS github NodeJS 中文社群

CHAPTER

THREE

轉角遇到 NODE

3.1 Hello World

An example of a web server written in Node which responds with ``Hello World'' for every request.

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(1337, "127.0.0.1");
console.log('Server running at http://127.0.0.1:1337/');
```

CHAPTER

FOUR

共同作者名冊

依照名字筆劃次序排列

Clonn

Hsu Ping Feng

Lyhcode