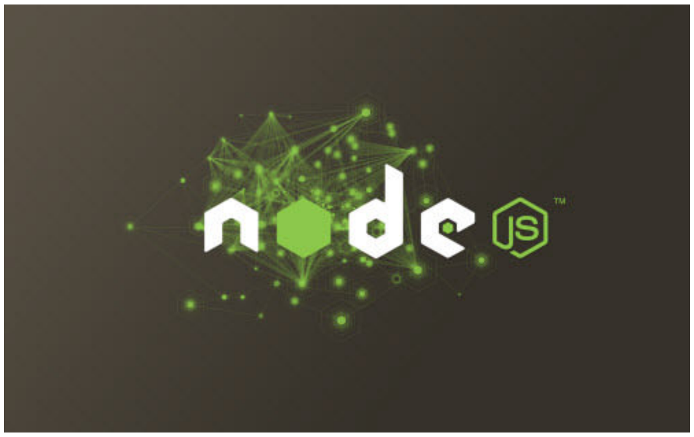
Geo Paul — Coding (Http://Www.Hongkiat.Com/Blog/Category/Coding/)

Beginner's Guide to Node.js (Server-side JavaScript)

Node.js (http://en.wikipedia.org/wiki/Nodejs) – in simple words – is **server-side JavaScript**. It has been getting a lot of buzz these days. If you've heard of it or you're interested in learning and getting some hands on it – this post is for you.



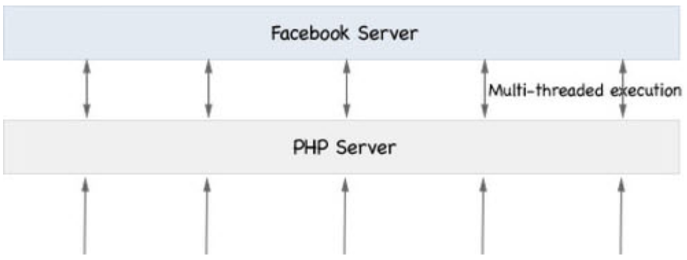
(Image Source: node.js (http://nodejs.org/logos/))

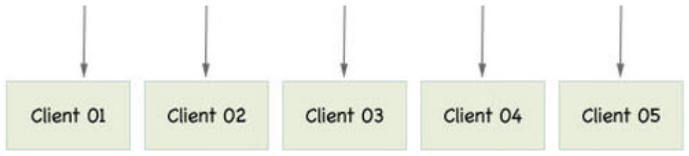
So what exactly is the need of using JavaScript in the server? To make the concept of Node.js clear I would like to compare it with the ordinary server-side languages such as PHP. Node.js uses an **event-based server execution procedure** rather than the multithreaded execution in PHP.

To explain it further, we'll be talking about the idea of **what Node.js is** along with some **hosting provider suggestions and installation tips**. Intermediate level knowledge of JavaScript, jQuery and Ajax are required, but we'll also provide examples for you to understand the entire thing easier and even work on it, so let's get to know more about Node.js!

Let's consider a case:

Consider a website in which you need to load content dynamically from another web server which is slow. In PHP you can do it in 2 ways – **coding it in a simple file** and **coding it as another script**, then **executing it in a multithreaded approach**.



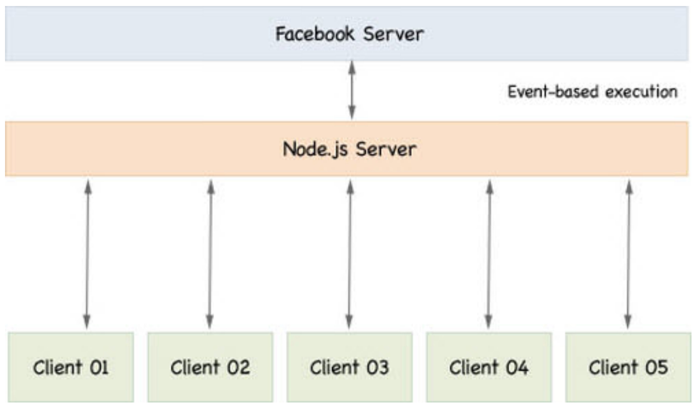


In the first method even though the code is simple **the execution pauses for a while** at the point where the slow web server is accessed. The second method is **more optimized in case of performance** but it's hard to code and it has a multithread management overhead. The case is similar for most of the web programming languages other than the server-side JavaScript, i.e., Node.js.

What's the difference in Node.js? In order to understand Node.js you must keep in mind the **JavaScript's event based programming in the browser**. We utilize the same technology here. Instead of using a separate thread, a **function is attached to the finish event** of the "slow web server access" mentioned above, thus you get the **full functionality** in the optimized second option mentioned above without any multithread overhead.

Getting started with Node.js

Node.js is JavaScript. Why can't we utilize the event based functionality of JavaScript in the client to a server? This thought might have led to the development of Node.js.



That said, the main highlight of Node.js – **it is event-based asynchronous functions**. It uses an **event loop** instead of waiting for I/O operations (accessing external web service, accessing hardware).

Node.js could still **make use of its processing power** when the server is waiting for any other operation. This makes Node.js **scalable** to millions of concurrent connections.

How Does JavaScript Run On A Server?

Node.js works on a v8 environment (http://code.google.com/p/v8/) – it is a **virtual machine** or a **JavaScript engine** that runs the JavaScript code, so for hosting you can't use the ordinary web hosts. You will need the ones that have the **v8 environment**.

Here are some provider suggestions for Node.js hosting:

1. bejes.us (http://bejes.us/)
2. Cloud Foundry (http://www.cloudfoundry.com/)
3. Cloudnode (http://cloudno.de/)
4. DotCloud (https://www.dotcloud.com/)
5. Duostack (http://duostack.com)

Installing Node.js

Node will **work perfectly on Linux, Macintosh, and Solaris operating systems**. On Windows you can install it using the Cygwin emulation layer (http://www.cygwin.com/install.html). None of the builds in Windows are satisfactory but it is still possible to get something running.

Option 1: Building Node from source.

Use `make` to build and install node.js (execute the following on the command line). **Git (http://book.git-scm.com/2_installing_git.html) is required**.

```
git clone --depth 1 git://github.com/joyent/node.git
cd node
git checkout v0.4.11
export JOBS=2
mkdir ~/local
./configure --prefix=$HOME/local/node
make
make install
echo 'export PATH=$HOME/local/node/bin:$PATH' >> ~/.profile
echo 'export NODE_PATH=$HOME/local/node:$HOME/local/node/lib/node_modules' >> ~/.pro
source ~/.profile
```

Option 2: Installing Node.js from Package

For Mac users, you can install Node.js as a package from https://sites.google.com/site/nodejsmacosx/ (https://sites.google.com/site/nodejsmacosx/) which is pretty self-explanatory.

Testing Node Installation

In order to check your successful Node installation we can try out a very simple console "Hello World" program. Create a file named "*test.js*" and write the following code into it.

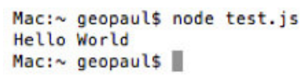
```
var sys = require("sys");
sys.puts("Hello World");
```

Code explanation:

It loads the `sys` class into a variable `sys`. It then uses the `sys` object to perform the console tasks. The `sys.puts` is a command similar to the `cout` in C++, so in order to run the script above go to the command prompt and run it by the command below:

```
node test.js
```

If your installation is successful then you will get a hello world output in the screen.



```
Mac:~ geopaul$ node test.js
Hello World
Mac:~ geopaul$ 
```

Creating a HTTP Server

Now it’s time to create a "Hello World" via web server using Node.js. Here’s what we are going to do – we **create a server that outputs a “Hello World” to the localhost on the port 8080** no matter what the URL is, giving you an idea what **event** is.

The codes:

```
var sys = require("sys"),
my_http = require("http");
my_http.createServer(function(request,response){
    sys.puts("I got kicked");
    response.writeHeader(200, {"Content-Type": "text/plain"});
    response.write("Hello World");
    response.end();
}).listen(8080);
sys.puts("Server Running on 8080");
```

Code explanation:

The most interesting part in Node.js is its event-based programming. In order to create a HTTP server we need the **HTTP library**, so we go forward and add it using `my_http`. We create server by the function:

```
my_http.createServer(function(request,response){}).listen(8080);
```

The function given as the first argument is executed **every time an event is triggered in port 8080**, so the function itself **suggests the node to listen for an event in port 8080**. In order to detect this, I have added a “*I got kicked*” message which will be displayed on the console screen whenever a request is received.

The `request` object contains **all the information about the request that has been made to the server**. For example it contains the URL string. The `response` object is the object that **handles the response from the server**. First we set the header of the response as a `text/plain` content, then outputs “*Hello World*“, then end the output stream. *200* is the status response.

Well, the above one is a very simple example but we can see that whatever URL we give in the browser for the same server we get the same output like “Hello World”.

Creating simple static file server

Let's create a simple static file server in the next tutorial.

The codes:

```
var sys = require("sys"),
my_http = require("http"),
path = require("path"),
url = require("url"),
filesys = require("fs");
my_http.createServer(function(request,response){
    var my_path = url.parse(request.url).pathname;
    var full_path = path.join(process.cwd(),my_path);
    path.exists(full_path,function(exists){
        if(!exists){
            response.writeHeader(404, {"Content-Type": "text/plain"});
            response.write("404 Not Found\n");
            response.end();
        }
        else{
            filesys.readFile(full_path, "binary", function(err, file) {
                 if(err) {
                     response.writeHeader(500, {"Content-Type": "text/plain"});
                     response.write(err + "\n");
                     response.end();

                 }
                 else{
                    response.writeHeader(200);
                    response.write(file, "binary");
                    response.end();
                }

            });
        }
    });
}).listen(8080);
sys.puts("Server Running on 8080");
```

Codes explanation:

The above code is pretty simple, we will discuss it as blocks.

```
var sys = require("sys"),
my_http = require("http"),
path = require("path"),
url = require("url"),
filesys = require("fs");
```

All these libraries are required for the program. Its usage will be clear in the following code.

```
var my_path = url.parse(request.url).pathname;
var full_path = path.join(process.cwd(),my_path);
```

The `request` object has the request details as we have discussed earlier. We use the `parse` function of the URL class which we have included to get the `pathname` of the request URL. After getting the pathname we concatenate it with the path of the current working directory in order to get the full path of the file.

For joining URLs we have a function called `join` in the path library.

```
path.exists(full_path,function(exists){
```

After getting the full path we check whether the path exists by the function **`exists`** . After the check is done the callback function is called and passed as the second argument.

```
if(!exists){
    response.writeHeader(404, {"Content-Type": "text/plain"});
    response.write("404 Not Found\n");
    response.end();
}
else{
    filesys.readFile(full_path, "binary", function(err, file) {
         if(err) {
             response.writeHeader(500, {"Content-Type": "text/plain"});
             response.write(err + "\n");
             response.end();

         }
         else{
            response.writeHeader(200);
            response.write(file, "binary");
            response.end();
        }

    });
}
```

Now in the callback function if the file does not exist we send a "*404 Page Not Found*" error.

If the page is found then we read the file by the **`readFile`** function in file system. We can also see the callback function for the **`readFile`** function defined there itself. If there is no error in reading the file then it will be displayed. If there is an error then a status 500 is returned with the error text.

I also recommend wrapping codes of the previous tutorial into a function so that you can use it in the next tutorial or for future use.

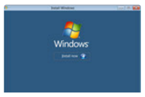
```
var sys = require("sys"),
my_http = require("http"),
path = require("path"),
url = require("url"),
filesys = require("fs");
my_http.createServer(function(request,response){
    var my_path = url.parse(request.url).pathname;
    var full_path = path.join(process.cwd(),my_path);
    path.exists(full_path,function(exists){
        if(!exists){
            response.writeHeader(404, {"Content-Type": "text/plain"});
            response.write("404 Not Found\n");
            response.end();
        }
        else{
            filesys.readFile(full_path, "binary", function(err, file) {
                 if(err) {
                     response.writeHeader(500, {"Content-Type": "text/plain"});
                     response.write(err + "\n");
                     response.end();

                 }
                 else{
```

Popular In Hongkiat

(http://www.hon right-javascript-mvc-framework/)

Choosing the Right JavaScript MVC Framework [Infographic] (http://www.hongkiat.com/blog/choosing-right-javascript-mvc-framework/)
Coding (/Blog/Category/Coding)


(http://www.hon bootcamp-without-optical-drive/)

How to Install Windows Boot Camp Without An Optical Drive (http://www.hongkiat.com/blog/install-bootcamp-without-optical-drive/)
Desktop (/Blog/Category/Desktop)

(http://www.hon design-to-codes/)

10+ Services & Tools To Convert Designs To Codes (http://www.hongkiat.com/blog/convert-design-to-codes/)
Toolkit (/Blog/Category/Toolkit)

(http://www.hon library-interactive-charts/)

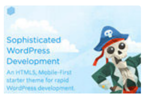
9 Javascript Libraries To Build Interactive Charts (http://www.hongkiat.com/blog/js-library-interactive-charts/)
Toolkit (/Blog/Category/Toolkit)

(http://www.hon elements-jquery-plugins/)

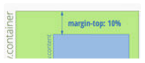
15 jQuery Plugins To Make Smart Sticky Elements (http://www.hongkiat.com/blog/sticky-elements-jquery-plugins/)
Toolkit (/Blog/Category/Toolkit)

(http://www.hon playing-cards/)

10 Unique Playing Card Designs You'd Rather Keep than Play (http://www.hongkiat.com/blog/unique-playing-cards/)
Artwork (/Blog/Category/Artwork)


(http://www.hon wordpress-themes/)

20 WordPress Starter Themes For Developers (http://www.hongkiat.com/blog/starter-wordpress-themes/)
WordPress (/Blog/Category/Wordpress)



What You Don't Know About Calculating

```
                 else{
                    response.writeHeader(200);
                    response.write(file, "binary");
                    response.end();
                }

            });
        }
    });
}
my_http.createServer(function(request,response){
    var my_path = url.parse(request.url).pathname;
    load_file(my_path,response);
}).listen(8080);
sys.puts("Server Running on 8080");
```

Stay Tuned !

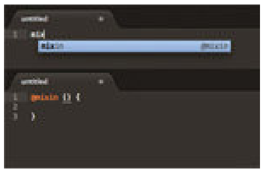
That's all. Hope this gives you a good idea of Node.js. In the next article, I'll be showing you how to **load and display number of Facebook likes using Node.js**. Stay tuned!

Editor's note: *This post is written by* ***Geo Paul*** *for Hongkiat.com (http://www.hongkiat.com/). Geo is an independent Web/iPhone developer who enjoys working with PHP, Codeigniter, WordPress, jQuery and Ajax. He has 4 years of experience in PHP and 2 years experience iniPhone application Development.*

Readers also read:

Wrapping Content In Shapes With CSS3 (http://www.hongkiat.com/blog/css3-content-wrapping/)

Working with Text in Scalable Vector Graphics (SVG) (http://www.hongkiat.com/blog/scalable-vector-graphics-text/)



Working with Code Snippets in Sublime Text (http://www.hongkiat.com/blog/sublime-code-snippets/)

Read more posts on: javascript (http://www.hongkiat.com/blog/tag/javascript/) , node.js (http://www.hongkiat.com/blog/tag/node-js/) , tutorials (http://www.hongkiat.com/blog/tag/tutorials/)

Comments for this thread are now closed. ×

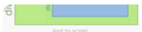
14 Comments   Hongkiat.com   1 Login

Recommend 12   Share   Sort by Newest

Radhemohan327 · 4 years ago
Awesome Article...
5 ^ | v · Share ›

moscato · 4 years ago
Thanks for grate information.
7 ^ | v · Share ›

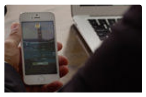
adumpaul · 4 years ago
Great tutorial.Thank you.
^ | v · Share ›


(http://www.hon css-percentage-margins/)

About Calculating Percentage Margins in CSS (http://www.hongkiat.com/blog/calculate-css-percentage-margins/)
Coding (/Blog/Category/Coding)

(http://www.hon responsive-website-webydo/)

Create a Responsive Site Without a Single Line of Code Using Webydo (http://www.hongkiat.com/blog/create-responsive-website-webydo/)
Web Design (/Blog/Category/Design)


(http://www.hon productivity-apps-freelancers/)

10 Must-Have Mobile Apps for Productivity (http://www.hongkiat.com/blog/mobile-productivity-apps-freelancers/)
Freelance (/Blog/Category/Freelance)

(http://www.hon apple-docks/)

20+ Apple Watch Docks – The Best So Far (http://www.hongkiat.com/blog/best-apple-docks/)
Gadgets (/Blog/Category/Gadgets)

(http://www.hon resume-template/)

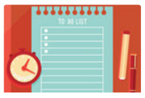
30 Free & Beautiful Resume Templates To Download (http://www.hongkiat.com/blog/editable-resume-template/)
Toolkit (/Blog/Category/Toolkit)

(http://www.hon lessons-from-avengers/)

20 Lessons The Avengers Can Teach You About Design (http://www.hongkiat.com/blog/design-lessons-from-avengers/)
Web Design (/Blog/Category/Design)

(http://www.hon freelance-work-with-full-time-job/)

5 Tips To Tackle Freelancing With a Full-time Job (http://www.hongkiat.com/blog/manage-freelance-work-with-full-time-job/)
Freelance (/Blog/Category/Freelance)


(http://www.hon ebooks-designers-

15 Free Ebooks for Designers & Developers in 2015 (http://www.hongkiat.com/blog/free-ebooks-designers-

Nora Reed • 4 years ago

Very nice review of this javascripts! it will gonna help me to learn n make more improvement in my self!
thanks :)

Share ›

WebpageLottery • 4 years ago

I am kinda confused.
Is that mean Node.js is powerful enough to replaced PHP?

21 • Share ›

adumpaul • 4 years ago

So useful post.I love this post.

Share ›

alicia • 4 years ago

nice write up ..really explained well :) thank youuu!!

Share ›

Brilliant Use of HTML Lists in Web Design (http://www.hongkiat.com/blog/html-lists-styling/)

Showcase of Web Designs with Beautiful Typography (http://www.hongkiat.com/blog/showcase-web-design-beautiful-typography/)

RohanAdvani • 4 years ago

Great POST!!!

Share ›

Fontana homes for sale • 4 years ago

Thanks for this post. Javascript is actually one of my favorite programming language since it's kind of simple and easy to apply. Actually that label might go to HTML but you get what I mean. :) Keep up the good work on this blog.

8 • Share ›

Kizi • 4 years ago

Thanks for share a very great articles.I love this post.

Share ›

Tom_pott • 4 years ago

ok thank for this write up. but i dont know what you are talking about.

22 • Share ›

Aijin John • 4 years ago

Very nice article , actually i was searching for detailed description on node.js .. this one really helps .. thanks geo

1 • Share ›

Subscribe Add Disqus to your site Privacy

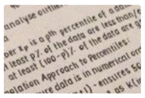
(http: u=htt js- serve side- javas side%

(http: text=I side% js- serve side- javas

(https url=h js- serve side- javas side%

developers-2015/)

developers-2015/)
Web Design (/Blog/Category/Design)


(http://www.hon handwritings/)

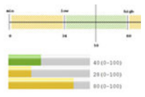
44 Handwriting So Beautiful You'd Be Jealous (http://www.hongkiat.com/blog/beautiful-handwritings/)
Artwork (/Blog/Category/Artwork)

(http://www.hon css-animations/)

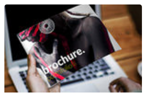
30 Cool CSS Animations You Have To See (http://www.hongkiat.com/blog/creative-css-animations/)
Coding (/Blog/Category/Coding)

(http://www.hon blogging-ebooks/)

12 Free Ebooks to Teach You Blogging and Content Marketing (http://www.hongkiat.com/blog/free-blogging-ebooks/)
Blogging (/Blog/Category/Blogging)


(http://www.hon html5-meter/)

Using and Styling HTML5 Meter [Guide] (http://www.hongkiat.com/blog/style-html5-meter/)
Coding (/Blog/Category/Coding)


(http://www.hon brochure-psds/)

20 Free Brochure PSDs You Can Download (http://www.hongkiat.com/blog/free-brochure-psds/)
Graphics (/Blog/Category/Graphics)