The Server-side JavaScript

Vikash Singh

What to expect ahead….

- Introduction
- Some (Confusing) Theory
- 5 Examples
- A couple of weird diagrams
- 2 Pics showing unbelievable benchmarks
- Some stuff from Internet
- And Homer Simpson

Background

- **V8** is an open source JavaScript engine developed by Google. Its written in C++ and is used in Google Chrome Browser.
- **Node.js** runs on V8.
- It was created by **Ryan Dahl** in 2009.
- Is still in Beta phase. Latest version is **0.6.11**
- Is **Open Source**. It runs well on Linux systems, can also run on Windows systems.
- If you have worked on EventMachine (Ruby) or Python's Twisted or Perl's AnyEvent framework then following presentation is going to be very easy.

Introduction: Basic

- In simple words Node.js is **‘server-side JavaScript’**.
- In *not-so-simple* words Node.js is a high-performance **network applications framework**, well optimized for high concurrent environments.
- It’s a **command line** tool.
- In ‘Node.js’ , ‘**.js’** doesn’t mean that its solely written JavaScript. It is 40% JS and 60% C++.
- From the official site:

  ‘Node's goal is to provide an easy way to build scalable network programs’ **-** (from nodejs.org!)

Introduction: Advanced (& Confusing)

- Node.js uses an **event-driven**, **non-blocking I/O** model, which makes it lightweight. (from nodejs.org!)
- It makes use of **event-loops** via JavaScript's **callback** functionality to implement the non-blocking I/O.
- Programs for Node.js are written in JavaScript but not in the same JavaScript we are use to. There is no DOM implementation provided by Node.js, i.e. you **can not** do this:

```
var element = document.getElementById("elementId");
```

- Everything inside Node.js runs in a **single-thread**.

Example-1: Getting Started & Hello World

- Install/build Node.js.
  - (Yes! Windows installer is available!)
- Open your favorite editor and start typing JavaScript.
- When you are done, open cmd/terminal and type this:

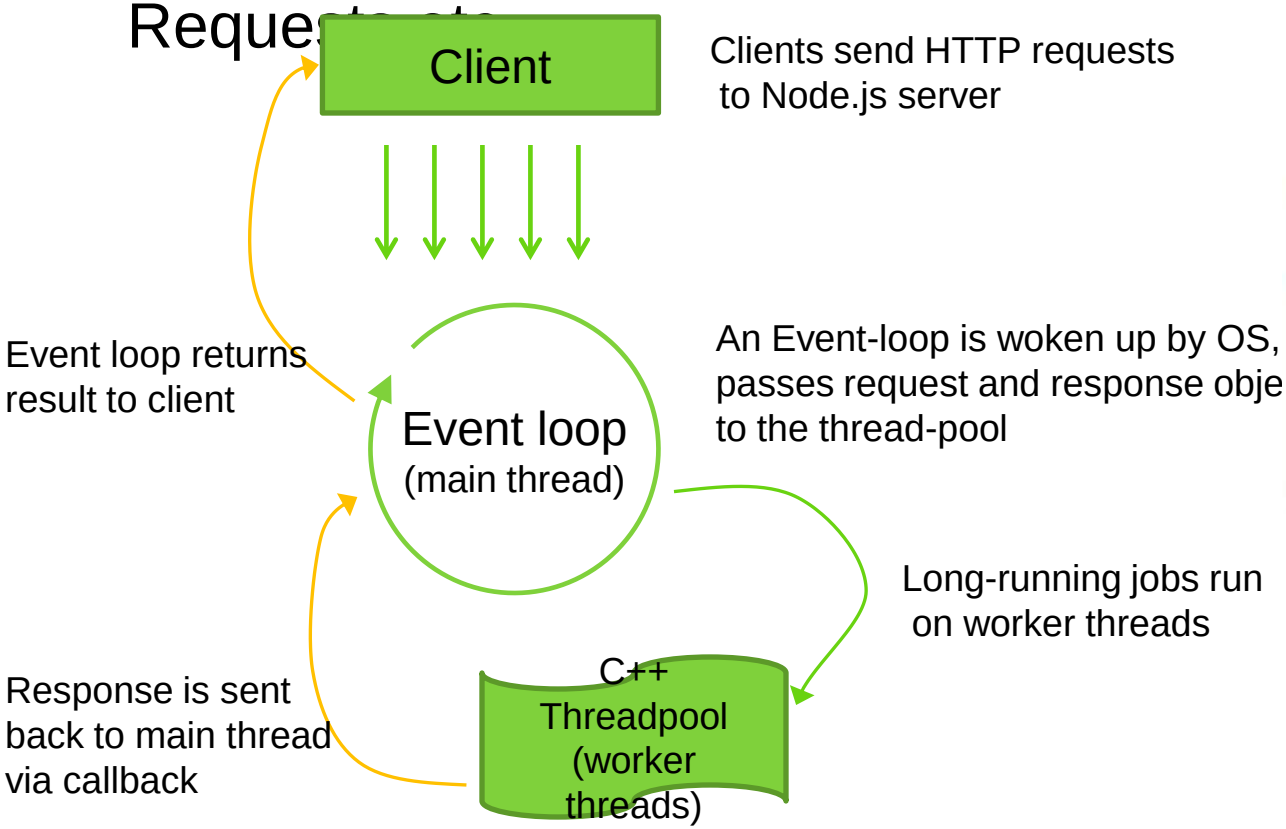
```
'node YOUR_FILE.js'
```

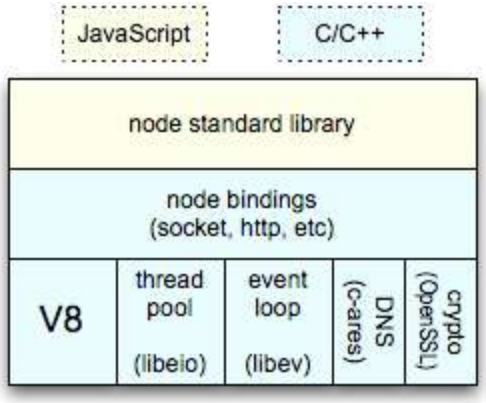
- Here is a simple example, which prints '*hello world*'

```
var sys = require("sys");
setTimeout(function(){
sys.puts("world");},3000);
sys.puts("hello");
//it prints 'hello' first and waits for 3 seconds and then
  prints 'world'
```

Some Theory: Event-loops

- Event-loops are the core of event-driven programming, almost all the UI programs use event-loops to track the user event, for example: Clicks, Ajax Reque





Some Theory: Non-Blocking I/O

- Traditional I/O

```
var result = db.query("select x from table_Y");
doSomethingWith(result); //wait for result!
doSomethingWithOutResult(); //execution is blocked!
```

- Non-traditional, Non-blocking I/O

```
db.query("select x from table_Y",function (result){
  doSomethingWith(result); //wait for result!
});
doSomethingWithOutResult(); //executes without any
  delay!
```

What can you do with Node.js ?

- You can create an **HTTP server** and print '*hello world*' on the browser in just 4 lines of JavaScript. (Example included)
- You can create a **TCP server** similar to HTTP server, in just 4 lines of JavaScript. (Example included)
- You can create a **DNS server**.
- You can create a **Static File Server**.
- You can create a **Web Chat Application** like GTalk in the browser.
- Node.js can also be used for creating online games, collaboration tools or anything which sends updates to the user in real-time.

Example -2 &3 (HTTP Server & TCP Server)

- Following code creates an **HTTP** Server and prints '*Hello World*' on the browser:

```
var http = require('http');
http.createServer(function (req, res) {
res.writeHead(200, {'Content-Type': 'text/plain'});
res.end('Hello World\n'); }).listen(5000, "127.0.0.1");
```

- Here is an example of a simple **TCP** server which listens on port 6000 and echoes whatever you send it:

```
var net = require('net');
net.createServer(function (socket) {
socket.write("Echo server\r\n");
socket.pipe(socket); }).listen(6000, "127.0.0.1");
```

Node.js Ecosystem

- Node.js heavily relies on **modules**, in previous examples **require** keyword loaded the http & net modules.
- Creating a module is easy, just put your JavaScript code in a separate js file and include it in your code by using keyword require, like:

```
var modulex = require('./modulex');
```

- Libraries in Node.js are called packages and they can be installed by typing

```
npm install "package_name"; //package should be
available in npm registry @ nmpjs.org
```

- **NPM** (Node Package Manager) comes bundled with Node.js installation.

Example-4: Lets connect to a DB (MongoDB)

- Install mongojs using npm, a mongoDB driver for Node.js

```
npm install mongojs
```

- Code to retrieve all the documents from a collection:

```
var db = require("mongojs")
.connect("localhost:27017/test", ['test']);
db.test.find({}, function(err, posts) {
  if( err || !posts) console.log("No posts found");
  else posts.forEach( function(post) {
    console.log(post);
  });
});
```

When to use Node.js?

- Node.js is good for creating streaming based real-time services, web chat applications, static file servers etc.
- If you need high level concurrency and not worried about CPU-cycles.
- If you are great at writing JavaScript code because then you can use the same language at both the places: server-side and client-side.
- More can be found at: http://stackoverflow.com/questions/5062614/how-to-decide-when-to-use-nodejs

Example-5: Twitter Streaming

- Install nTwitter module using npm:

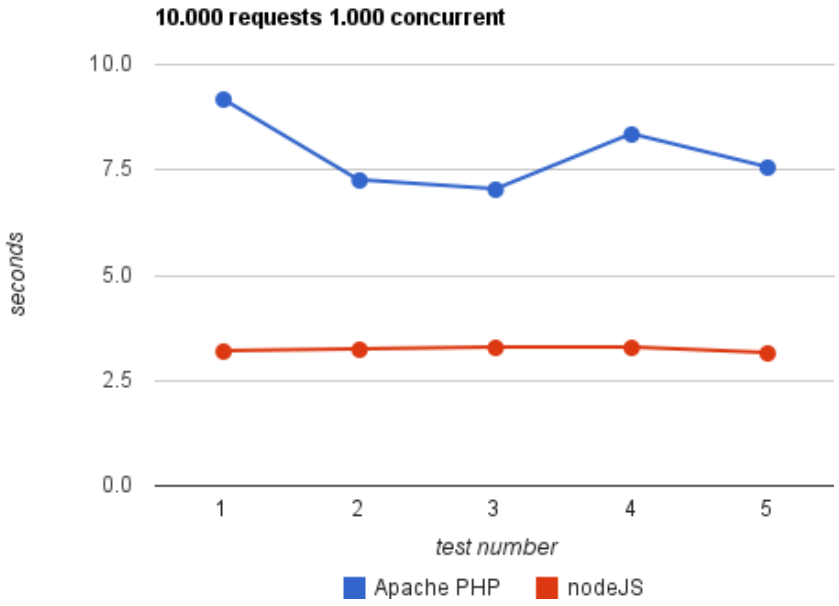
```
Npm install ntwitter
```

- Code:

```
var twitter = require('ntwitter');
var twit = new twitter({
      consumer_key: ‘c_key’,
      consumer_secret: ‘c_secret’,
      access_token_key: ‘token_key’,
      access_token_secret: ‘token_secret’});
twit.stream('statuses/sample', function(stream) {
    stream.on('data', function (data) {
      console.log(data); });
});
```

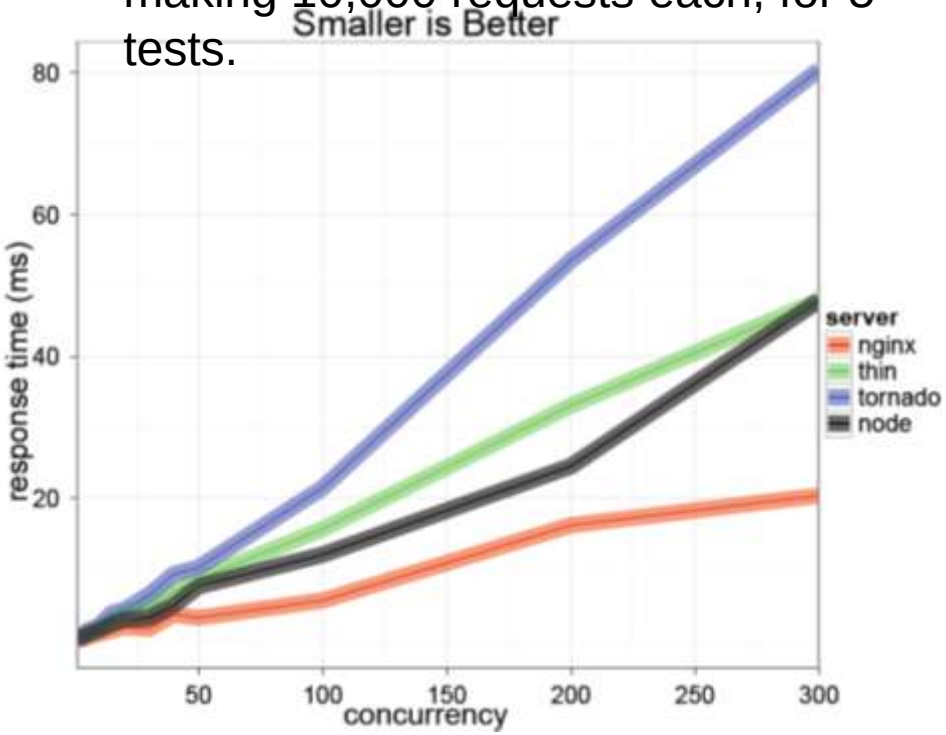

THE
END
IS
NEAR

Some Node.js benchmarks



Taken from:
http://code.google.com/p/node-js-vs-apache-php-benchmark/wiki/Tests
A benchmark between Apache+PHP and node.js, shows the response time for 1000 concurrent connections making 10,000 requests each, for 5 tests.

Taken from:
http://nodejs.org/jsconf2010.pdf
The benchmark shows the response time in milli-secs for 4 evented servers.



When to not use Node.js

- When you are doing heavy and CPU intensive calculations on server side, because event-loops are CPU hungry.
- Node.js API is still in beta, it keeps on changing a lot from one revision to another and there is a very little backward compatibility. Most of the packages are also unstable. Therefore is not yet production ready.
- Node.js is a no match for enterprise level application frameworks like
  Spring(java), Django(python), Symfony(php) etc.
  Applications written on such platforms are meant to be highly user interactive and involve complex business logic.
- Read further on disadvantages of Node.js on Quora:
  http://www.quora.com/What-are-the-disadvantages-of-

Appendix-1: Who is using Node.js in production?

- **Yahoo!** : iPad App **Livestand** uses Yahoo! Manhattan framework which is based on Node.js.
- **LinkedIn** : LinkedIn uses a combination of Node.js and MongoDB for its mobile platform. iOS and Android apps are based on it.
- **eBay** : Uses Node.js along with ql.io to help application developers in improving eBay's end user experience.
- **Dow Jones** : The WSJ Social front-end is written completely in Node.js, using Express.js, and many other modules.
- Complete list can be found at: https://github.com/joyent/node/wiki/Projects,-Applications,-and-Companies-Using-Node

Appendix-2: Resource to get started

- Watch this video at Youtube: http://www.youtube.com/watch?v=jo_B4LTHi3I
- Read the free O'reilly Book *'Up and Running with Node.js'* @ http://ofps.oreilly.com/titles/9781449398583/
- Visit www.nodejs.org for Info/News about Node.js
- Watch Node.js tutorials @ http://nodetuts.com/
- For Info on MongoDB: http://www.mongodb.org/display/DOCS/Home
- For anything else **Google!**

Appendix-3: Some Good Modules

- **Express** – to make things simpler e.g. syntax, DB connections.
- **Jade** – HTML template system
- **Socket.IO** – to create real-time apps
- **Nodemon** – to monitor Node.js and push change automatically
- **CoffeeScript** – for easier JavaScript development
- Find out more about some widely used Node.js modules at: http://blog.nodejitsu.com/top-node-module-creators


THANKS
A
TON