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.
- Latest version is **16.4**
- 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

- 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

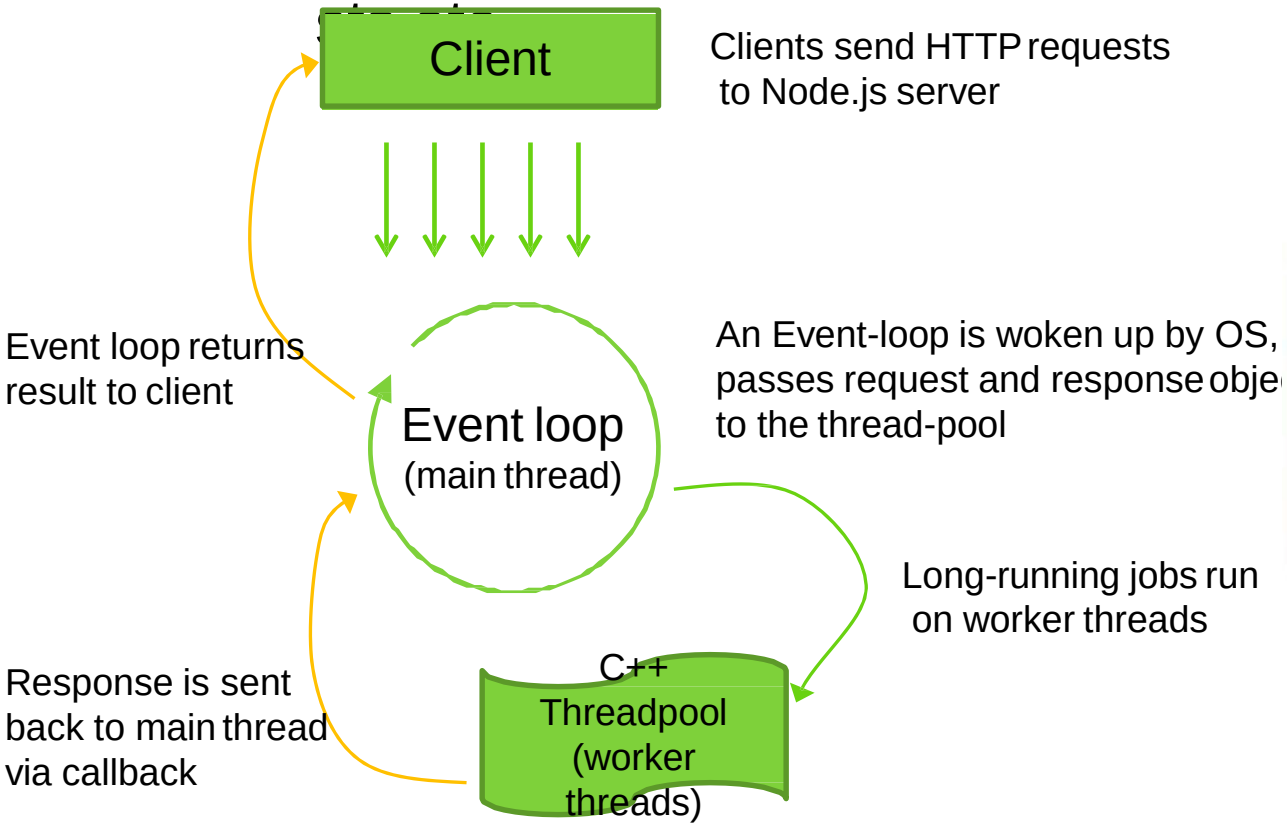
- Install/build Node.js.
  - (Yes! Windows installer is available!)
- Open your favorite editor and start typing JavaScript.
- When you are done, open cmd/terminal and type this:

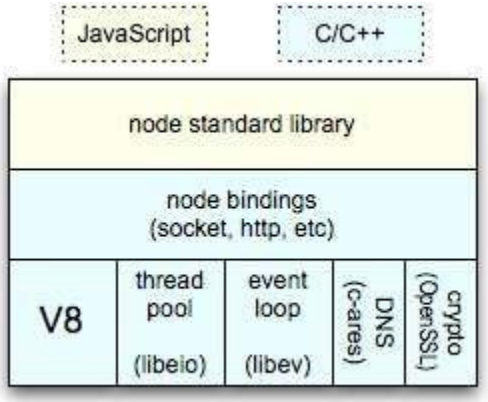
  ‘node YOUR_FILE.js’
- Here is a simple example, which prints ‘*hello world*’

```
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’
```

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





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.

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.

npm – Node Package Manager

- npm – Node Package Manager npm is the package manager for javascript and more.
- https://www.npmjs.com/

- npm install moduleName -option

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.

WHEN TO NOT USE NODE.JS

- When you are doing heavy and CPU intensive calculations on server side, because event-loops are CPU hungry.
- Still, Node's relational database support tools are not up to the expected level when compared to other languages. This makes Node an undesirable for use cases with relational databases

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.

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