# Node.js

“*JavaScript has certain characteristics that make it very different than other dynamic languages, namely that it has no concept of threads. Its model of concurrency is completely based around events.*” - Ryan Dahl

# Event in the DOM

Dom Element Events

→ Click

→ Hover

→ MouseOver

→ …

```
$(”element").on("click", function(){ ... });
```

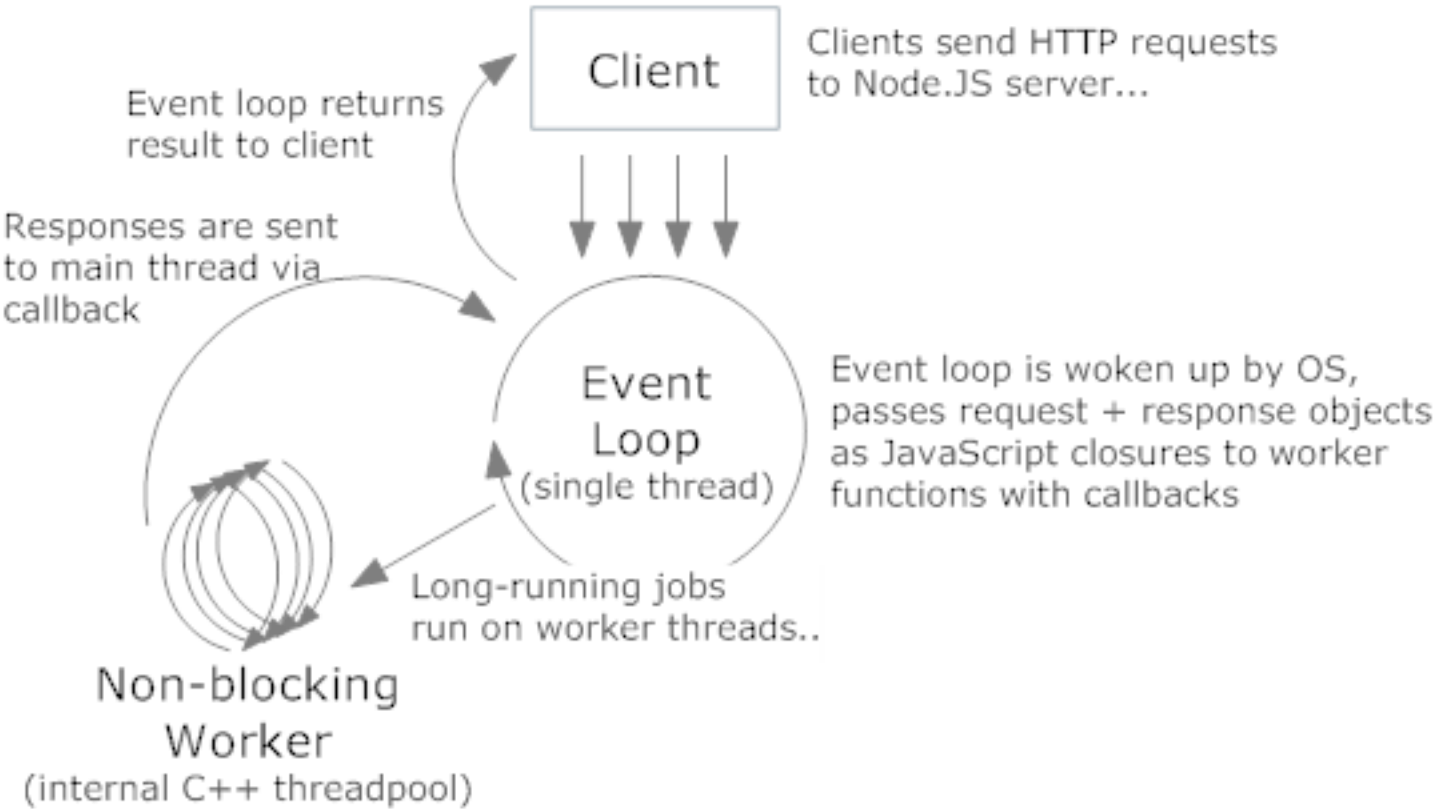
# Event Listener

```
var http = require('http');

var server =
http.createServer(function(req, res) {
  res.writeHead(200);
  res.end('Hello World');
});
server.listen(8080);
```

# Event Loop

Node.JS Processing Model



Client
Event Loop (single thread)
Non-blocking Worker (internal C++ threadpool)

Clients send HTTP requests to Node.JS server…

Event loop is woken up by OS, passes request + response objects as JavaScript closures to worker functions with callbacks

Long-running jobs run on worker threads..

Responses are sent to main thread via callback

Event loop returns result to client

# Event Loop

Is it same as multithreading?

# Module

```
var http = require('http');

var server =
http.createServer(function(req, res) {
  res.writeHead(200);
  res.end('Hello World');
});
server.listen(8080);
```

# Module

```
var http = require('http'); → http.js
var fs= require(fs'); → fs.js
```

How does ‘require’ return the libraries?

How the libraries are structured?

# Module

→ Use require to use other modules in your code

→ Use exports in your module to expose it publicly

```
var other = require('other_module');
module.exports = function() {
    console.log(other.doSomething());
```

# Module

How can you turn this code into a module?

```
var http = require('http');
var server =
http.createServer(function(req, res) {
  res.writeHead(200);
  res.end('Hello World');
});
server.listen(8080);
```

# Module

```
var http = require('http');
function start() {
    var server = http.createServer(function(req, res)
    {
       res.writeHead(200);
       res.end('Hello World');
    });
    server.listen(8080);
}
exports.start = start;
```

# Module

Using the module

```
var server = require("./server");
server.start();
```

# Module

→ Ordinary Javascript files following CommonJS module spec

→ Run on their own scope so that they do not conflict with other modules

→ Node facilitate module interoperability using some globals

# Module

Module in file,

```
var var= require(‘./fileName'); → same directory
var var= require(‘./fileName'); → parent directory
var var= require(‘/dir/fileName'); → absolute directory
```

Module as a package,

should located in

/Home/mashiyat/my_app/node_module

# Node Package Manager (npm)

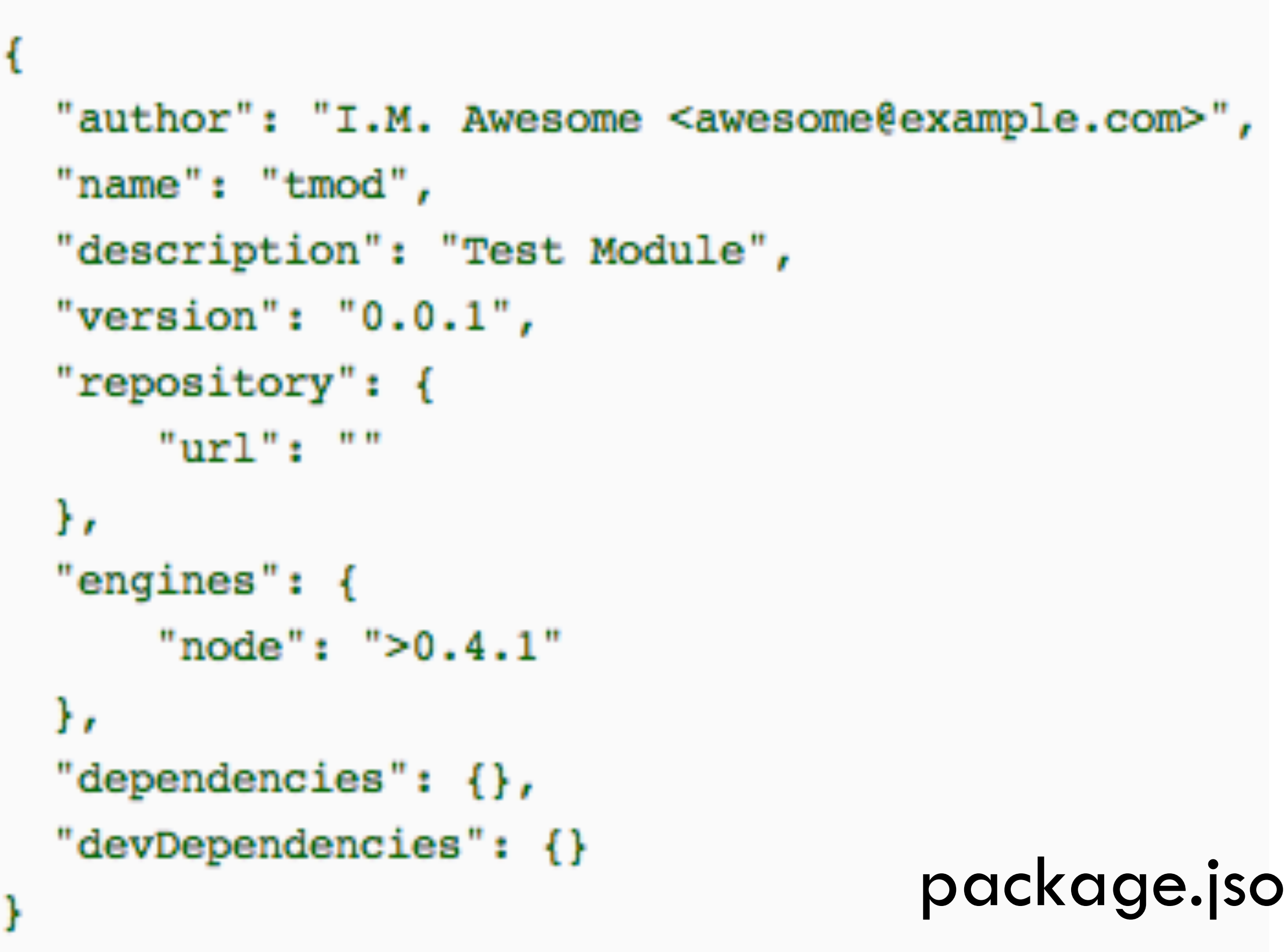
→ npm is the official package manager for Node.js http://npmjs.org

→ Search for node module by,

```
$ npm search packageName
```

→ If you know the module name you can install by, `$ npm install moduleName`

# Node Package Manager (npm)



```
{
  "author": "I.M. Awesome <awesome@example.com>",
  "name": "tmod",
  "description": "Test Module",
  "version": "0.0.1",
  "repository": {
      "url": ""
  },
  "engines": {
      "node": ">0.4.1"
  },
  "dependencies": {},
  "devDependencies": {}
}
```

package.json

# Node Version Manager (NVM)

Node is constantly evolving!

get **nvm** to manage your versions!