## Getting started with Node JS servers

1. Hello World

1. Install Node.JS from [here](https://nodejs.org/en/download/) and follow the install instructions specific to your O/S.
2. Upon installation, open a command window and type the following:

node –v

It should report back the version correctly if the installation was successful.

At the time of writing mine reported v0.12.7

Node should be globally available (e.g. added to Windows system path)

1. Creating Hello World: Open a command window and create a new file, app.js. (In windows, notepad app.js) Add the following content:

console.log(‘Hello World’);

Save the file and execute it on the command line with:

Node app.js

Console should show:

Hello World

2. Simplest node server

1. In a new command window, create a file. Server.js.
2. In the file place the following code:

**var** http = require("http");

http.createServer(**function**(request, response) {

response.writeHead(200, {"Content-Type": "text/plain"});

response.write("Hello from the simple node server”);

response.end();

}).listen(8888);

1. Save the file and start the simple server with:

Node server.js

1. Open a browser and navigate to http://localhost:8888/
2. Should show ‘Hello from the simple node server’.
3. Use CRTL C to stop the server.

3. Second simplest node server (modularized)

The purpose of this exercise is to modularize our application into a server (server.js) and our main file, index.js. The intent is to launch a single file, index.js, which in turn will “require” the server module and cause the launch.

1. In a new command window, create a modify server.js with the following code:

**var** http = require("http");

**function** start() {

**function** onRequest(request, response) {

console.log("Request received.");

response.writeHead(200, {

"Content-Type" : "text/plain"

});

response.write("Hello from the simple node server");

response.end();

}

http.createServer(onRequest).listen(8888);

console.log("Server has started.");

}

exports.start = start;

1. Save the server.js and create index.js with the following content:

**var** server = require("./server");

server.start();

1. Start the new server with

Node index.js

1. Open a browser and navigate to <http://localhost:8888/>

Why two request each time?

Further reading: <https://nodejs.org/docs/v0.4.2/api/modules.html>

Investigate “blocking” and “non-blocking” operations and how they apply to nodeJS.

Why is it important to use only non-blocking operations in node js code?

Why are callbacks important?

Look [here](http://openmymind.net/2012/2/3/Node-Require-and-Exports/) from some information on Node.js “require” and “exports”

4. Third simplest node server (w/router)

The purpose of this exercise is to launch a node server able to fetch resources from a public directory of your choosing. Additionally, and with the use of a ‘router’ module, we can set the mime type of the resource in the response. Lastly this server will support a favicon image.

All the code for this server has been placed in the ‘Third\_Simplest\_Sever’ folder. Launch it with:

Node server.js

Then open a browser window and navigate to http://localhost:8888/index.html

Points of interest in the code:

1. Modularity
2. Handles the ‘/’ path.
3. Allows port and public folder to be set easily.
4. Uses the ‘fs’ module to read and server static file names.
5. Uses a simple prototype to determine the file extension.
6. Uses the node global object, ‘\_\_dirname’.
7. Could further be modularized by exporting a start method in server.js (See Second Simplest node Server).

The URL module is great for handling incoming requests. The image below demonstrates how to extract the various parts of a request URL:

