

Node Exercises - Bravissimo

Steps

Installing Node.js

Install Node - Clone Node GitHub Repository & Run 'nvm.sh'

<http://nodejs.org/download/>

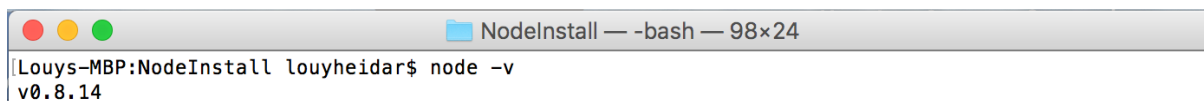
- Installers available for Windows & Mac OS X
- Binaries available for Windows, Mac, Linux and SunOS
 - Also available via many Linux package managers
- Source code also available

Manage multiple versions with 'nvm' (<https://github.com/creationix/nvm>)

- `git clone git://github.com/creationix/nvm.git ~/nvm`
- `. ~/nvm/nvm.sh`
- `nvm install 0.8.14`
- `nvm use 0.6.19`
- `nvm alias default 0.8.14`

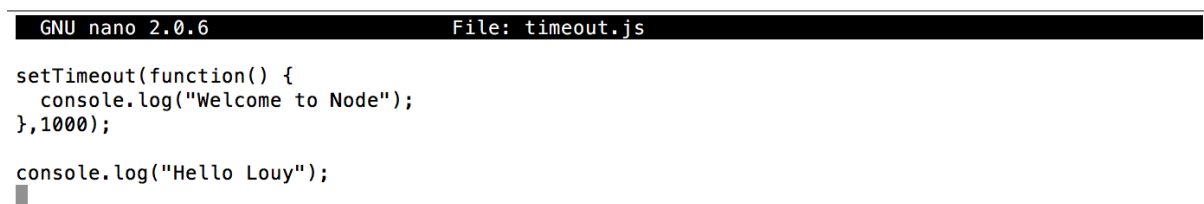
Check version

`node -v`

A terminal window titled 'NodeInstall — -bash — 98x24' showing the command 'node -v' being executed. The output is 'v0.8.14'. The terminal window has a title bar with red, yellow, and green window control buttons on the left.

```
Louys-MBP:NodeInstall louyheidar$ node -v
v0.8.14
```

Timeout Node Program

A code editor window titled 'GNU nano 2.0.6 File: timeout.js' showing the contents of a file named 'timeout.js'. The code contains two console.log statements: one for 'Welcome to Node' and another for 'Hello Louy'. The first statement is wrapped in a setTimeout function with a 1000ms delay. The cursor is at the end of the second line of code.

```
GNU nano 2.0.6 File: timeout.js

setTimeout(function() {
  console.log("Welcome to Node");
},1000);

console.log("Hello Louy");

```

[Read 5 lines]

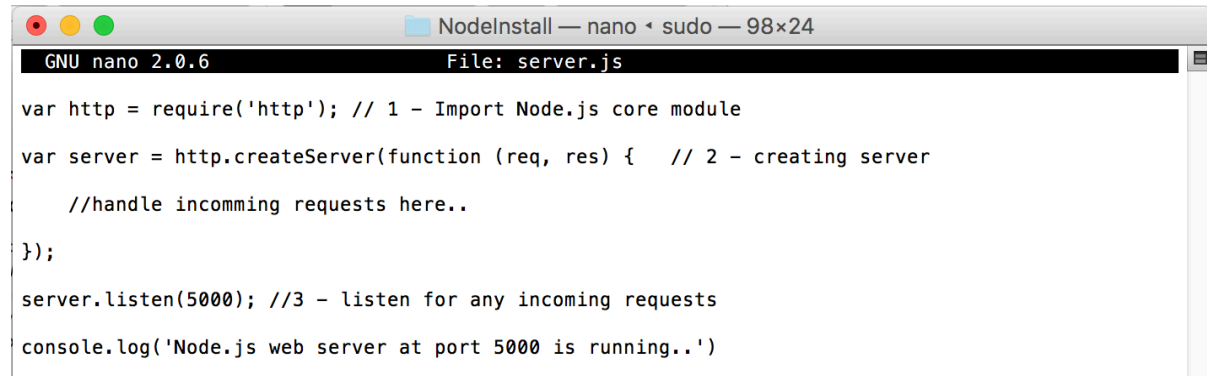
Executing the Timeout Program (timeout.js)

`node timeout.js`

Creating Node.js Web Server:

Node.js makes it easy to create a simple web server that processes incoming requests asynchronously.

Example below is a simple Node.js web server contained in 'server.js' file.



```
GNU nano 2.0.6 File: server.js

var http = require('http'); // 1 - Import Node.js core module

var server = http.createServer(function (req, res) { // 2 - creating server
    //handle incoming requests here..
});

server.listen(5000); //3 - listen for any incoming requests

console.log('Node.js web server at port 5000 is running..')
```

Above example, http module is imported using require() function. http module is a core module of Node.js, so no need to install it using NPM.

Next step is to call createServer() method of http and specify callback function with request and response parameter.

Finally, call listen() method of server object which was returned from createServer() method with port number. This starts listening to incoming requests on port 5000.

Running above web server by writing node server.js terminal window will display message as shown below.

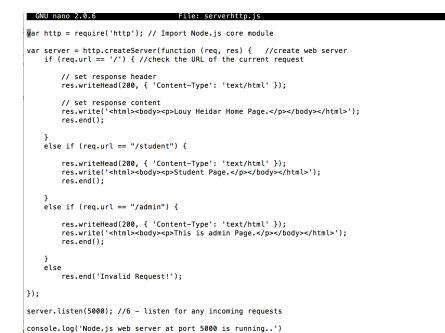
```
Node.js web server at port 5000 is running..
```

Handle HTTP Request:

http.createServer() method includes request and response parameters which is supplied by Node.js.

The request object can be used to get information about the current HTTP request e.g., url, request header, and data. The response object can be used to send a response for a current HTTP request.

Example below demonstrates handling HTTP request and response in Node.js



```
GNU nano 2.0.6 File: serverhttp.js

var http = require('http'); // Import Node.js core module
var server = http.createServer(function (req, res) { //create web server
    if (req.url === '/') { //check the URL of the current request
        // set response header
        res.writeHead(200, { 'Content-Type': 'text/html' });
        // set response content
        res.write('<html><body><p>Lousy Heider Home Page.</p></body></html>');
        res.end();
    }
    else if (req.url === "/student") {
        res.writeHead(200, { 'Content-Type': 'text/html' });
        res.write('<html><body><p>Student Page.</p></body></html>');
        res.end();
    }
    else if (req.url === "/admin") {
        res.writeHead(200, { 'Content-Type': 'text/html' });
        res.write('<html><body><p>This is admin Page.</p></body></html>');
        res.end();
    }
    else {
        res.end('Invalid Request!');
    }
});

server.listen(5000); //6 - listen for any incoming requests
console.log('Node.js web server at port 5000 is running..')
```

```
GNU nano 2.0.6 File: serverhttp.js
var http = require('http'); // Import Node.js core module

var server = http.createServer(function (req, res) { //create web server
  if (req.url == '/') { //check the URL of the current request

    // set response header
    res.writeHead(200, { 'Content-Type': 'text/html' });

    // set response content
    res.write('<html><body><p>Louy Heidar Home Page.</p></body></html>');
    res.end();

  }
  else if (req.url == "/student") {

    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.write('<html><body><p>Student Page.</p></body></html>');
    res.end();

  }
  else if (req.url == "/admin") {

    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.write('<html><body><p>This is admin Page.</p></body></html>');
    res.end();

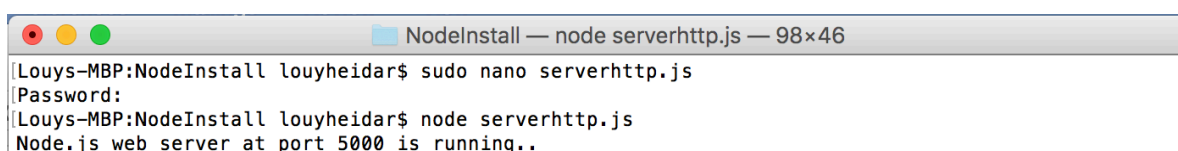
  }
  else
    res.end('Invalid Request!');
});

server.listen(5000); //6 - listen for any incoming requests
console.log('Node.js web server at port 5000 is running..')
```

Above example, req.url is used to check url of the current request and based on that it sends response. To send a response, you first set response header using writeHead() method and then writes a string as a response body using write() method.

Finally, Node.js web server sends the response using end() method.

Running the webserver is demonstrated



```
NodelnInstall — node serverhttp.js — 98x46
Louys-MBP:NodelnInstall louyheidar$ sudo nano serverhttp.js
Password:
Louys-MBP:NodelnInstall louyheidar$ node serverhttp.js
Node.js web server at port 5000 is running..
```

<http://localhost:5000>

