Node JS

An  is a powerful [cross-platform](https://en.wikipedia.org/wiki/Cross-platform)[JavaScript](https://en.wikipedia.org/wiki/JavaScript) [run-time environment](https://en.wikipedia.org/wiki/Runtime_system) for executing JavaScript code [server-side](https://en.wikipedia.org/wiki/Server-side). It enables JavaScript to be used for [server-side scripting](https://en.wikipedia.org/wiki/Server-side_scripting), and runs scripts server-side to produce [dynamic web page](https://en.wikipedia.org/wiki/Dynamic_web_page) content *before* the page is sent to the user's web browser. It used to develop web applications like single-page applications, streaming sites, and other web applications. Node.js is open source, free and used by a lot of web developers.

Node.js is written in JavaScript, and can be run within Node.js runtime on OS X, Microsoft Windows and Linux. It also provides a rich library of various JavaScript modules which simplifies the development of web applications.

Installing Node.js on Windows

1. Download the Node.js installer for Windows from the Node.js website.
2. Click and Run the installer.
3. Accept the license agreements, followed by a bunch of times and accept the default installation settings.

Testing Node.js

Make sure Node and NPM is installed:

1. To check if Node is installed, open Windows Command Prompt
2. Type ‘node –v’
3. A version of the installed node.js should appear (i.e. v0.10.35)
4. For NPM, type ‘npm -v’ in your terminal.
5. Again, an NPM’s version should appear
6. To test if node.js works, create a javascript file

Name it index.js, and type/add some code ‘console.log(‘Node is installed successfully!’)’

1. Run your code. Open command prompt and change your file directory to where the file has been saved. Type ‘node index.js’ and your Node should start running.
2. A “Node is installed successfully” should be the output.

Creating simple Node.js web server:

1. Create a file named ‘index.js’
2. Open your file and add necessary codes for the server to work.

var http = require(‘http’),

The require(‘http’) will load http module and store returned HTTP instance into a variable.

http.createServer( function(request, response){

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

response.end(‘Hello World !’);

});

server.listen(81);

console.log(‘Server is listening in port 81…..’);

The createServer() method will create and HTTP server. It takes callback functions as an argument and this function will be executed each time the server receives a new request. The argument will be the response and request object. The request object contains information regarding clients request, such as URL and HTTP headers. While the response object will be used to return requested data back to the client.

Inside the createServer method, begin your callback by calling the response.writeHead() method. The method sends a collection of response headers back to the client. The status code, for example, will be used to indicate the result of request. 404 for requested page not found, 200 for success and others. The Content-Type, indicated within response.writeHead(), means that a particular header defines a MIME type of response, in our code the MIME type is “text/html”.

After the HTML page has been written, the response.end() method is called. The method will tell the server that the response headers and body have been sent and the request is complete.

The call to listen() will cause the server to bind to a port 81 and listen for incoming connections.

1. Execute your index.js to start your server. There should be an output that a server is running.

Node index.js

1. Open your web browser and type <http://localhost:81/>

A Hello World plain text should appear.

Node.js modules

Node.js allows the creating of Web servers and networking tools using JavaScripts and a collection of muddles that handles various functionalities.

Including Modules:

To include a module, use the require() function with the name of the module:

var http = require(‘http’);

Node.js Built-in Modules:

assert Provides a set of assertion tests

buffer To handle binary data

child\_process To run a child process

cluster To split a single Node process into multiple processes

crypto To handle OpenSSL cryptographic functions

dgram Provides implementation of UDP datagram sockets

dns To do DNS lookups and name resolution functions

domain Deprecated. To handle unhandled errors

events To handle events

fs To handle the file system

http To make Node.js act as an HTTP server

https To make Node.js act as an HTTPS server.

net To create servers and clients

os Provides information about the operation system

path To handle file paths

punycode Deprecated. A character encoding scheme

querystring To handle URL query strings

readline To handle readable streams one line at the time

stream To handle streaming data

string\_decoder To decode buffer objects into strings

tls To implement TLS and SSL protocols

tty Provides classes used by a text terminal

url To parse URL strings

util To access utility functions

vm To compile JavaScript code in a virtual machine

zlib To compress or decompress files

MEAN.js (MongoDB, ExpressJS, AngularJS, and Node.js)

MEAN.JS is a full-stack JavaScript solution that helps you build fast, robust, and maintainable production web applications using MongoDB, Express, AngularJS, and Node.js. Because all components of the MEAN stack support programs written in JavaScript, MEAN applications can be written in one language for both [server-side](https://en.wikipedia.org/wiki/Server-side) and [client-side](https://en.wikipedia.org/wiki/Client-side) execution environments.

The components of the MEAN stack are as follows:

* [**M**ongoDB](https://en.wikipedia.org/wiki/MongoDB), a [NoSQL](https://en.wikipedia.org/wiki/NoSQL) database
* [**E**xpress.js](https://en.wikipedia.org/wiki/Express.js), a [web application framework](https://en.wikipedia.org/wiki/Web_application_framework) that runs on Node.js
* [**A**ngular.js](https://en.wikipedia.org/wiki/Angular.js) or [**A**ngular](https://en.wikipedia.org/wiki/Angular_(application_platform)), JavaScript [MVC](https://en.wikipedia.org/wiki/Model-view-controller) frameworks that run in browser JavaScript engines
* [**N**ode.js](https://en.wikipedia.org/wiki/Node.js), an execution environment for [event-driven](https://en.wikipedia.org/wiki/Event-driven_architecture) server-side and networking applications