**An Introduction To NodeJS -** [Ryan Schleck](https://ryan-m-schleck.medium.com/?source=post_page-----aeddbc7c6a91--------------------------------) Jun 5, 2022

**Preface**

If you are in to web development or have been browsing the job boards for software engineering positions lately you’ve most likely come across mention of the term ‘Node’ or ‘NodeJS’. This blog post will seek to cover the history of this technology, a bit on how it works, and what it is used for today.

**What Is NodeJS?**

In the late 2000s with the popularity of Javascript surging many browsers competed to develop ways to execute it faster. In 2008 Google Chrome created an extremely fast [Javascript execution engine](https://en.wikipedia.org/wiki/JavaScript_engine" \t "_blank) called V8 and is considered the first modern example of this technology. A year later in 2009 a software engineer by the name of Ryan Dahl created an open source project built off googles V8 javascript engine called NodeJS. The goal of this project was to extend the functionality of Javascript beyond that of the browser. Another year later in 2010 a package manager was created to allow programmers to share and publish source code for NodeJS called [npm](https://en.wikipedia.org/wiki/Npm_(software)) which allowed its usage to skyrocket. NodeJS provides a Javascript runtime environment that can be run locally or on a web server to compile Javascript into optimized machine code. To install NodeJS on your own computer I recommend using a node version managing software called [nvm](https://github.com/nvm-sh/nvm" \l "intro" \t "_blank).

Install it by opening a terminal and entering either the following curl:

curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash

or by using a the following wget command

wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash

After this is complete you can then use nvm to install NodeJS simply by entering the following into your terminal:

nvm install node

After this is complete you should check your version of NodeJS in a new terminal by entering

node -v

Now that we know the general history of NodeJS and how to install it using nvm, lets look at some examples of how it is used.

**NodeJS As Local Build Tools**

As previously stated one of the original goals in creating NodeJS was to extend Javascripts functionality beyond that of the browser. With the installation of NodeJS you can run Javascript locally on your computer simply by opening a terminal and entering **node**like the following:

Text

Description automatically generated

The greater than symbol confirms you are in the NodeJS CLI. Within this you can run Javascript within your terminal such as adding two integers or logging a message as seen in this picture.

In order to exit this environment simply press CTRL+C twice. It is important to note that each session you enter this environment is completely new and all previously defined variables or functions will be lost. You can also use NodeJS to interact with your file system. Lets do so by opening a text editor (I use VS code) and doing a little exercise where we use NodeJS to create a text file. First navigate to a directory you’d like to work in and create a file called **app.js**. Within it we will use two utilities provided by NodeJS to generate this text file. First we will create a variable set to [require](https://nodejs.org/api/modules.html#requireid) and pass it an argument of ‘fs’ which is a module in NodeJS that allows it to interact with our file system. We will then use this to call [writeFileSync](https://nodejs.org/api/fs.html" \l "fswritefilesyncfile-data-options" \t "_blank) and pass it strings representing the file we’d like to creates name, and the content of the file.

Text

Description automatically generated

If you save this and open a terminal within the same directory you can use node to execute this file by entering **node** followed by the name of the file (extensions optional)

node app

this will create **hello.txt** within the same directory. If you open this new file you should see the text *Hello Node.js World!*Go ahead and try it out!

**NodeJS For Server Side Code**

The more popular usage for NodeJS is to create and manage a web server. One of the advantages NodeJS offers in this area is that it processes everything in an asynchronous non-blocking fashion. It does this with a single [thread](https://en.wikipedia.org/wiki/Thread_(computing)) that continuously checks the call stack to see which functions need to be run and which have finished and need to be returned. This process is known as the ‘event loop’ good resource on the NodeJS event loop can be found within this article [here](https://nodejs.dev/learn/the-nodejs-event-loop). This allows many processes to be firing at the same time and while handling all of this logic at once is a tad heavier on hardware, it allows better scalability and a quick response.

Now that we have a general idea of what’s going on in the background lets set up a single page web server with NodeJS. Navigate to a folder you would like to work in and lets again create a file called **app.js** to open up within your text editor. Within app.js we will once again set a variable (this time called **http**) to the module returned by using require. This time however we will pass require the string of ‘http’ to signify what we will be working with the [http module](https://nodejs.dev/learn/the-nodejs-http-module) in NodeJS. We will then set another variable called **server** and set it to our http variable calling the **[createServer](https://nodejs.dev/learn/the-nodejs-http-module" \l "httpcreateserver" \t "_blank)** method provided by the http module in NodeJS. We will pass this createServer method an anonymous function which handles what we do with our request and response. Finally we will use this server variable to call the [**listen**](https://nodejs.org/api/net.html#serverlisten) method on the port we would like to host our server on. The whole file should look like the following:

Text

Description automatically generated

On line 11 if you were to uncomment **process.exit()** this would close the server once reached. Otherwise the server will remain active and continue listening on the specified port.

If you again navigate to the folder where app.js is and open a terminal you can run this file just as before by entering:

node app

This will begin our server. If you then opened up your browser and navigated to **localhost:3000** you will be greeted by a header reading “Hello NodeJS World!” with the title of “My NodeJS Page”. If you were to then navigate back to the terminal you used to run your server you will see a log containing a whole lot of information pertaining to your request. You can exit this process by pressing CTRL+C on your keyboard. Congratulations! If you followed along you just hosted your first NodeJS server!

**Conclusion**

Of course NodeJS is not only limited to the simple task of serving a static web page like what we just did. It can also validate input, connect to a database, return json, and much more. These are topics I will most likely cover in future blog posts, but in the mean time feel free to utilize [the documentation](https://nodejs.org/en/docs/) to review this functionality for yourself.