“Hello World!” app with Node.js and Express

Do you use Node…? You will.

Adnan Rahić · Follow

6 min read · Jan 9, 2017

This article is aimed for beginner developers and anyone interested in getting up and running with Node.js. Before diving into this article, you should be confident enough with JavaScript to know the basic concepts of the language. Technical terms regarding Node will be explained and linked below.

Hello World!

What is Node.js?

Node is an asynchronous event driven JavaScript runtime built upon Chrome's V8 JavaScript engine. It's designed to build scalable network applications.

That being the raw definition, let me clarify. Node.js enables you to write server side JavaScript. You may now be wondering, how? As you know, JavaScript is a language

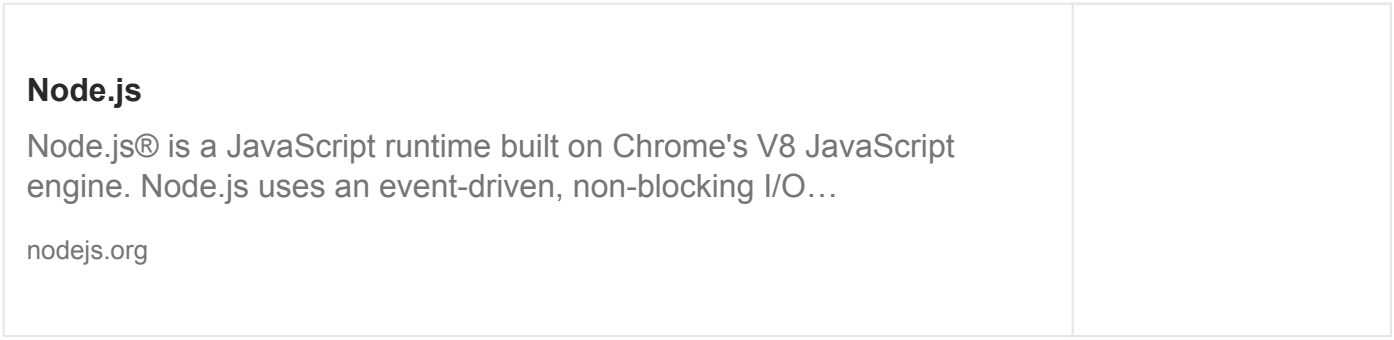
which runs in a browser. The browser's engine takes JavaScript code and compiles it into commands. The creator of Node.js took Chrome's engine and built a runtime for it to work on a server. Don't get confused with the word runtime. It's an environment where the language can get interpreted. So what do we have now? A way to write JavaScript on the back end.

Regarding the definition, you might be wondering what the term asynchronous even means in the current context. JavaScript is single threaded, meaning there is only one thread of execution. So you don't want events to interrupt the main thread of execution. This is what asynchronous means, handling events without interrupting the main thread. Node is based on this *non-blocking* execution, making it one of the fastest tools for building web applications today. In the following "Hello World" example, many connections can be handled concurrently. Upon each connection the callback is fired, but if there is no work to be done Node will remain asleep.

There are 6 simple steps in this example, bear with me.

. . .

1. **Install Node.js for your platform (MacOS, Windows or Linux)**



The first step is to get yourself an instance of the JavaScript runtime up and running on your local machine. Just smash **nodejs.org** in your browsers address bar, or click the link above, and you should be good to go. The home screen should give you what you want right away. As I am running Ubuntu on my machine, the respective version of Node.js for my operating system is listed. Go ahead, download and install it. This will give you the tools needed to run a server on your local machine.

. . .

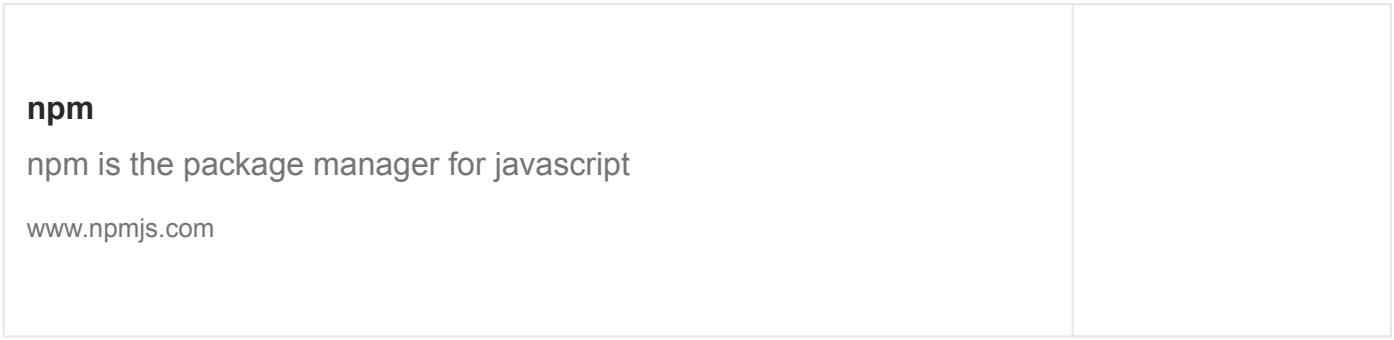
2. Open a command prompt and type:

```
mkdir myapp
cd myapp
```

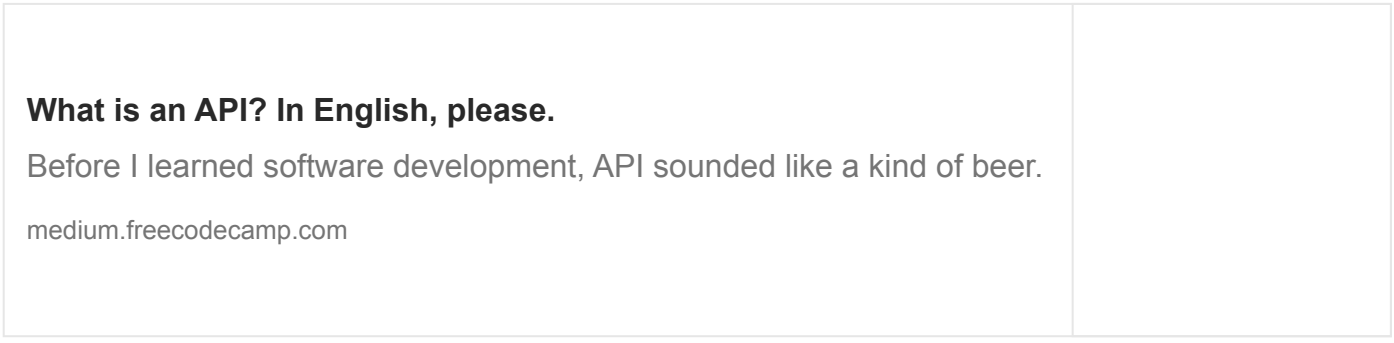
These commands are universal for whatever OS you'll be running. The former will create a new directory inside the directory you are currently in, *mkdir = "make directory"*. The latter will change into this newly created directory, *cd = "change directory"*. Hard-core windows users can calm down, this will work for you guys too, as it is equivalent to creating a new folder within your file system… only more fancy.

. . .

3. Initialize your project and link it to npm



Now the real fun starts. After creating your directory, very innovatively named *myapp*, you will need to initialize a project and link it to npm.
Np-what? Okay, calm down. Npm is short for *node package manager*. This is where all node packages live. Packages can be viewed as bundles of code, like modules, which carry out a specific function. This functionality is what we as developers are utilizing. We use the application program interface, the API, provided for us by these modules. What is an API you ask?



The modules in turn act like black boxes with buttons and levers we can push and pull to get the desired end result.

Running this command initializes your project:

```
npm init
```

This creates a *package.json* file in your *myapp* folder. The file contains references for all npm packages you have downloaded to your project. The command will prompt you to enter a number of things.

You can enter your way through all of them **EXCEPT** this one:

```
entry point: (index.js)
```

You will want to change this to:

```
app.js
```

. . .

4. Install Express in the *myapp* directory

Express - Node.js web application framework

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and…

expressjs.com

While still in the *myapp* directory run:

```
npm install express --save
```

The *install* command will go ahead and find the package you wish to install, and install it to your project. You will now be able to see a *node_modules* folder get created in the root of your project. This is a crucial step, as you will be able to *require* any of the recently installed files in your own application files. The addition of —save will save the package to your dependencies list, located in the *package.json*, in your *myapp* directory.

Yeah, I know what you're thinking. What exactly is this Express thing? Some mail delivery service, rivaling FedEx, I presume (please ignore these lame jokes). No, to burst your bubble, Express is a…

> "Fast, unopinionated, minimalist web framework for Node.js" — Taken from Express.js' official website

It gives you a set of robust and easy to use tools to get your web application up and running. Express has become so popular, it now is the de facto standard, in the vast majority of Node.js applications today. I strongly encourage the use of Express.

. . .

5. Start your text editor of choice and create a file named *app.js*.
Write the following:

```
var express = require('express');
var app = express();

app.get('/', function (req, res) {
  res.send('Hello World!');
});

app.listen(3000, function () {
  console.log('Example app listening on port 3000!');
});
```

Here is where you will need to use the package which was recently installed. The first line declares a variable which will contain the module called express, grabbing it from

the *node_modules* folder. The module is actually a function. Assigning the function call to another variable gives you access to a predefined set of tools which will in a great deal make your life much easier. You could view the variable *app* as an object, whose methods you are utilizing to build the actual program.

The listen method starts a server and listens on port 3000 for connections.
It responds with “Hello World!” for get requests to the root URL (/). For every other path, it will respond with a *404 Not Found*.

6. Run the app
Type the command:

```
node app.js
```

After running the command, load http://localhost:3000/ in a browser to see the output. You should also see “Example app listening on port 3000!” get logged to the command line.

Wow, so much output!

That’s it, you’re done. You have successfully created your first Node app. Don’t stop here, keep exploring the wonderful world of Node.js, as it has much more to offer.

. . .

Your finished app should have a folder structure somewhat resembling this.

Folder structure — green for folders — blue for files

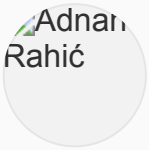
. . .

That’s it for me today. If you liked this article and if it helped you in any way, feel free to follow me, more tutorials like this one will be coming soon. If you believe this article will be of big help to someone, feel free to share.

Happy coding :)

If you liked this, click the ☐ below so other people will see this here on Medium.

Nodejs JavaScript Freecodecamp Coding Programming



Follow

Written by Adnan Rahić

4.97K Followers

Senior Developer Advocate at Tracetest.io part of Kubeshop.io. Failed startup founder, book/course author, and ex-freeCodeCamp local leader.

More from Adnan Rahić

Adnan Rahić in Kubeshop.io

Top 8 Open-Source Observability & Testing Tools

The DevOps mindset and the “shift-left” mentality impact how you work as a back-end engineer.

8 min read · Feb 23

30 1

Adnan Rahić in HackerNoon.com

A crash course on Serverless with Node.js

Regardless of your developer background, it’s inevitable you’ve heard the term Serverless in the past year. The word has been buzzing…

13 min read · Oct 4, 2017

10.9K 44

Adnan Rahić in We've moved to freeCodeCamp.org/news

Securing Node.js RESTful APIs with JSON Web Tokens

Have you ever wondered how authentication works? What's behind all the complexity and abstractions. Actually, nothing special. It's a way…

14 min read · Sep 4, 2017

Adnan Rahić in Sourcerer Blog

A Kubernetes quick start for people who know just enough about Docker to get by

What if I told you this is quite literally the Kubernetes guide you have been looking for?

20 min read · Mar 2, 2018

Melih Yumak in JavaScript in Plain English

Nodejs Developer Roadmap 2023

Explore nodejs developer roadmap for 2023. A step-by-step guide to how to become nodejs developer, increase knowledge as nodejs developer

· 7 min read · Jan 30

Elson Correia in Before Semicolon

How to Set up a TypeScript + NodeJs Server (2023)

With new releases and tools, setting up a node server has become super simple and until NodeJs ships with typescript built-in, adding…

· 7 min read · Feb 18

Lists

Stories to Help You Grow as a Software Developer

19 stories · 70 saves

Leadership

30 stories · 29 saves

How to Run More Meaningful 1:1 Meetings

11 stories · 35 saves

Stories to Help You Level-Up at Work

19 stories · 53 saves

Tom Nagle

Building Event-driven Microservices with Node.js & Kafka

Event-driven microservices are a design pattern for building scalable and resilient software systems. Instead of traditional monolithic…

· 2 min read · Dec 21, 2022

Lukas Wimhofer in Level Up Coding

API development with nodejs, express and typescript from scratch — Basic API

This is the third part of the series “API development with nodejs, express and typescript from scratch” and it is all about starting to…

· 9 min read · Feb 8