



JS

Setup Node.js

1

Node.js is a JavaScript runtime environment. This environment will include everything you need to start running JavaScript programs on your local machine!

2

Install Node.js

You can install Node.js by visiting nodejs.org.

3

Download and run the installer for your operating system. Once the installation is complete, you'll be able to run Node within a terminal window:

```
node
> 5 * 5
25
> const a = 3;
undefined
> a * 10
30
> 
```

By simply typing `node` and pressing enter you'll be able to run JavaScript commands!

This is called the Node.js REPL (Read-Eval-Print-Loop). You may exit it at any time by running `.exit`. Learn more about the [Node.js REPL here](#).

Opening a terminal window varies between operating systems. If you have not used the terminal window before, a quick Google search should help you quickly find the answer!

...

Download a Text Editor

In order to write code on our local machine, we need a text editor. There are many options available when it comes to text editors, varying from highly customizable environments with full testing suites to the free notepad application you find on every laptop!

Overall, it comes down to your preference. For a good lightweight editor, we recommend either [Atom](#) or [Sublime Text](#). These editors will give you everything you need to easily get coding and many developers still use these editors well into their career.

Run a Script

Now that we have a text editor, let's write a script and try running it!

Create a new file in your text editor and call it `index.js`

Add the following code to that file:

```
1 // this is my name, feel free to use yours :)
2 const myName = "Dan";
3
4 const message = `Hello, ${myName}!`;
5
6 console.log(message);
```

Save this file somewhere on your file system. Now in the terminal, let's navigate to where you saved the file.

Navigating the terminal is a skill you will master as you work with local development. The easiest way to navigate to a file is to copy and paste the path location of that file based on your file system directory. You can also manually type the path using forward-slashes ("/") to separate nested levels in your file system.

Once you're there, you can run the file by typing `node index` :

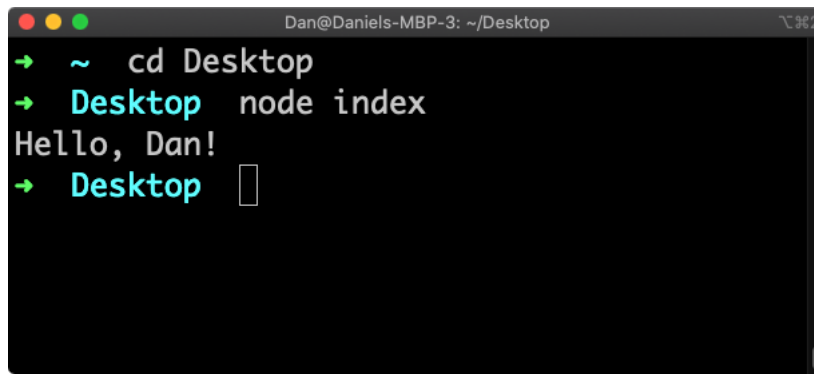


JS

1

2

3



```
Dan@Daniels-MBP-3: ~/Desktop
→ ~ cd Desktop
→ Desktop node index
Hello, Dan!
→ Desktop
```

Look it ran our program! How exciting!

You'll notice the command `cd` was used here. This is a common command line prompt in various Operating Systems. It means "change directory" and it allows us to navigate in the terminal to another location on our file system.

Command Line Arguments

Instead of writing `myName` inside of the file, we can pass it from the command line using `process.argv` !

Let's try editing the `index.js` file we wrote above.

```
1 | const message = `Hello, ${process.argv}!`;
2 |
3 | console.log(message);
```

Now we can pass in an argument to our `index.js` file by typing `node index Dan` .



```
Dan@Daniels-MBP-3: ~/Desktop
→ Desktop node index Dan
Hello, /usr/local/bin/node,/Users/Dan/Desktop/index,Dan!
→ Desktop
```

Whoops! What happened?

It turns out that `process.argv` is an **array of all argument values**. This includes the path to `node` as well as the script.

If we want to access the first argument we pass in after the script variable, we'll need to access it at index `2` :

```
1 | const message = `Hello, ${process.argv[2]}!`;
2 |
3 | console.log(message);
```

Now if we run this:



```
Dan@Daniels-MBP-3: ~/Desktop
→ Desktop node index Dan
Hello, Dan!
→ Desktop
```

Wrap Up



Great work on that one!

We went over how to installing Node.js, text editors, and running scripts in the terminal. We even learned how to pass arguments to our scripts! That's a great start to begin our networking chapter.

JS

Mark Complete

1

2

3

...