Running JavaScript Code

JavaScript is **the**language of the *Internet*! Whenever you browse your favorite website (google, facebook, twitter, appacademy.io), your web browser is executing JavaScript code. There are two main environments we use to run JavaScript: the first is the browser (Chrome, Firefox, etc.) and the second is Node. Writing code for the browser, (aka front end engineering), requires a lot more than just understanding JavaScript, so we'll come back to that topic later in the course. For now, we will concentrate on running JavaScript on our computers using Node.

So what is Nodeexactly? Node.jsis a very powerful runtime environment built on Google Chrome's JavaScript V8 Engine. It is used to develop I/O intensive applications like video streaming sites, robots, and other general purpose applications. For our purposes the most advantageous feature of Node is that it provides a way for us to run JavaScript outside of the browser.

Now that you have Node installed on your local computer it's time to run some JavaScript! Running your own code on your own computer is a rite of passage for all developers. We know you are up to the challenge!

By the end of this reading you should be able to:

- Use the Node REPL to test out simple expressions and functions
- Use VSCode to create a folder and a `.js`file within that folder
- Use Node to run a `.js`file

Node REPL vs. JavaScript File

Before we begin running code we wanted to make a clear distinction. Using Node there are two ways that we can run JavaScript code:

1. using the **Node REPL**
2. using Node to run a **.js file**

Both the Node REPL and using a JavaScript file are common ways to execute JavaScript code, but they are useful in different scenarios:

Node REPL(Read, Evaluate, Print, Loop) is used for testing quick ideas. The Node REPL is useful when playing around with any curiosities you have because you can see how an expression is evaluated quickly. Any code that you type into the Node REPL will be lost when you exit the REPL. If you've ever used a program that let you write a line of code and execute it immediately, without a separate command, then you've used a REPL.

JS Filesare used for saving code for the long term. If you create a `.js` file and save it then all the code within can be referenced and used later. When you work on problem sets, projects, and anything else you want to save, you should always save your code to a `.js`file!

Using the Node REPL

To use the **Node REPL**, simply open up your command line (Terminal) and enter the command `node`. In the examples below we use the `$` to show that we are in the command line (in our case Terminal).

```
~ $ node
Welcome to Node.js
Type ".help" for more information.
>
```

Notice that as soon as we enter the `node`command, we get a welcome message and we see our Terminal icon change to look like this: `>`. This `>` icon means that we are inside the Node REPL, so we can type any valid JavaScript lines and see what they evaluate to:

```
~ $ node
Welcome to Node.js
Type ".help" for more information.
> 1 + 1
2
> let message = "Hello" + "world"
undefined
> message
'Helloworld'
```

We can also define functions in the Node REPL though you'll find writing them in that environment is not super fun due to the Node REPL not being optimized for that kind of coding.

Here is an example of defining and invoking a function using `node`:

```
~ $ node
Welcome to Node.js
Type ".help" for more information.

> function sayHello () {
... console.log("hello!");
... }
undefined
> sayHello();
hello!
```

If you want to exit the Node REPL, and head back to our plain old command line enter the command `.exit`in the REPL. Doing this will get rid of the `>`icon, which means we are no longer in the REPL. When we are back inside our command line we can enter the normal commands (i.e `cd`, `ls`, `pwd`):

```
$ node
> 1 + 1
2
> "How do I get out of here" + "!?!?"
'How do I get out of here!?!?'
> .exit
~ $
```

Using JavaScript Files

The first thing you'll need in order to run a JavaScript file is to create a file that will contain the code you will be running. A new file is like a blank canvas - just awaiting the chance to be made into art.

If you don't currently have a dedicated coding folder start off by creating a new folder somewhere accessible, like your `Desktop`folder. Then you can open that folder using VS Code. From there you can simply create a "New File". In order to create a JavaScript file, make sure that you change the file name to one that ends in `.js`, for example `myFile.js`.

Now take a moment to enter some code into your new `.js` file like the following:

```
// AppAcademyWork/myFile.js
console.log("hello world!");
```

Don't forget to save the file with your new code!

Now to run a JS file you need to first go into the folder that contains that file by using `cd` in your command line. Feel free to use `ls`to list your folders and see where you have to go. Once you are inside of the correct folder, run `node <fileName>`, for example `node myFile.js`. When you enter these commands, be aware of the capitalization. **File names are case sensitive**!

```
~ $ ls
Downloads
Desktop
Music
Videos

~ $ cd Desktop
~ Desktop $ ls
AppAcademyWork

~ Desktop $ cd AppAcademyWork

~ AppAcademyWork $ ls
myFile.js

~ AppAcademyWork $ node myFile.js
Hello world
```

That is how you run JavaScript on your local computer! You create and save a file, navigate to that file in your terminal, then run the file using the `node`command followed by the filename (`node <fileName>`).

What you learned

- How to use the Node REPL to test out simple expressions and functions
- How to use VSCode to create a folder and a `.js`file within that folder
- How to use Node to run a `.js`file