

part3: JS string interpolation requires backticks (` `) not single-quotes (' ').

In this assignment, you will get some practice with server-side scripting using Node.js. As discussed, to use Node.js, you first need to [install it on your machine](#). Once you've setup Node.js, you will be able to run your scripts through the command line interface.

Download the files for the lab and place them inside your webpages directory under a new directory called lab8.

For parts 1 and 2, you should open them in your browser, since Node.js still doesn't support every ES6 feature. Parts 3 and 4 can be run using the Node.js command line.

Part 1 (10 pts)

Visit part1.html in your browser and open the console to see the output. Next, open up part1.js and take a look through the code. In the comments that follow, answer what each console.log statement prints and give your best explanation as to why that is.

Part 2 (40 pts)

Here, you will be getting some practice with ES6 classes. Your task is to rewrite your Binary Search Tree from [Lab 4](#) using ES6 class declaration syntax. Everything else should still work the same way as it did in Lab 4.

Create a file called part2.js and do the following:

1. You will need two classes, one for the individual nodes, and one for the BST.
2. You will need three functions, insert, search, and inorder, all of which behave the same way as they did in Lab 4.

Lastly, to take advantage of ES6 modules, export both your Node and BinarySearchTree classes so that they can be imported in the test file, part2.html, which will test the BST the same way it was tested in Lab 4.

Part 3 (30 pts)

In this part, you will write a server side script that returns HTML containing a multiplication table with all of the products from 1 to MAX, where MAX is a constant defined at the top of the script. Your final table should look something like this:

1 * 1 = 1	1 * 2 = 2	1 * 3 = 3	1 * 4 = 4	1 * 5 = 5	1 * 6 = 6
2 * 1 = 2	2 * 2 = 4	2 * 3 = 6	2 * 4 = 8	2 * 5 = 10	2 * 6 = 12
3 * 1 = 3	3 * 2 = 6	3 * 3 = 9	3 * 4 = 12	3 * 5 = 15	3 * 6 = 18
4 * 1 = 4	4 * 2 = 8	4 * 3 = 12	4 * 4 = 16	4 * 5 = 20	4 * 6 = 24
5 * 1 = 5	5 * 2 = 10	5 * 3 = 15	5 * 4 = 20	5 * 5 = 25	5 * 6 = 30
6 * 1 = 6	6 * 2 = 12	6 * 3 = 18	6 * 4 = 24	6 * 5 = 30	6 * 6 = 36

Each row should be a `<tr>` element and each cell inside each row should be a `<td>` element. Your multiplication table script should be scalable, that is, if I changed the constant MAX to 10, for instance, it should produce a multiplication table from 1 to 10 without changing any of the script code.

Note: You can dynamically write HTML by passing it as a string to the `res.write()` method

Part 4 (20 pts)

In this part, you will practice using the `fs` module. First, create a simple HTML file, `part4.html`, with any content you like (it can even just say "Hello World!"). Now, that we have the file we're going to read, modify the script, `part4.js`, to read that file and return it in the response.