**Reflective Report**

**Phase 1 and 2**

Nouraldin Hassan

Contents

[Evidence of Learning 3](#_Toc179647756)

[Program Architecture 7](#_Toc179647757)

[Program Demonstration via Screen Shots 9](#_Toc179647758)

[Unit Testing Demonstration via Screen Shots 10](#_Toc179647759)

[Source Code Commenting Example 10](#_Toc179647760)

# What Was Learned About my Programming Language

The programming language that I have learned is Server-Side Javascript utilized with Node.JS. Node.JS itself is a runtime environment that allows software developers to run JavaScript code outside of a web browser. It is used primarily for Server-Side Applications. The following topics have been learned alongside the programming language:

* C.R.U.D. Functions for CSV files.
* The use of imports/requirements from modules and other programming code files.
* The use of readline for an interactive Command Line Interface.
* How to run Node.JS’ own unit testing service.
* How to utilize callbacks

The strengths that I have found in Node.JS and Server-Side Javascript is that it handles asynchronous activities very well, to where I/O operations are nearly instantaneous for simple command line operations, as well as being able to use it for both frontend and backend development, and having a large ecosystem of libraries and modules for as many purposes as required, all open source too [1] [2].  
  
The weaknesses I have found however is that there is something called Callback Hell, where multiple nested callbacks result in very indented code that makes it hard to maintain and comprehend, and Node.JS being single-threaded meaning that CPU-Intensive tasks limit it’s capability and therefore affect it’s performance [3] [4].

# The Best Resources for Me to Learn Are

The NodeJS documentation.

General resource types are listed in the table below:

|  |  |  |  |
| --- | --- | --- | --- |
| Resource | Time Consumption | Usefulness | Rank |
| Books | High | Medium | Medium |
| Documentation Sites | Low | High | High |
| Online Videos | Medium | High | High |
| Training Sites | Low | High | Medium |
| Other Websites | Medium | Medium | Medium |

The most effective resource I found was the Web Documentation sites, due to the information displayed at a practical level and shows quick and simple detail on how a given function, method, or parameter works.

# WBS, Project Management Software, Reflection of Time Estimation

The timing was miscalculated to where it was very difficult to find out what has been over-estimated and what has been under-estimated, due to the time-providing restriction where I could not use the dates and times that I pick to have it span the start date and the due date (i.e. 8:00 AM cannot be changed to 11:59 PM for the same date for starting and ending).  
That has at least taught me to plan dates ahead of time and format them based better on actual estimated or intended completion, rather than on exact due dates of completion in reference to Brightspace assignments, so I can at least submit the assignments utilizing the charts before the due date ends and can have some room for improvement via quick fixes and resubmitting when it is needed.

# Discussion Board Post Archive

**Discussion Forum 01 Post:**

In the past week, I have learned how to use Node.JS and it’s own test runner for use with assert and test cases, and have applied the tests to the program code as part of the assignments. I had also learned that:

* Node.JS allows server-side JavaScript to be a thing in the first place.
* It can read and parse CSV files with use of the “fs” and “csv-parser” libraries/modules.
* I learned about how .pipe and .on methods work in the use case of “fs” and “csv-parser”

I found that the most interesting part of Node.JS and server-side JavaScript was its use of parsing the data from CSV files, due to the involvements of working with real data given and allowing me to simply see the data from having it fed into the program rather than me opening the data, and I can select or decide upon what specific data or value can appear in the program, which can save some time in more practical cases.

Website resources worked best for me, specifically the node.js documentation, because it covers almost every method there is that is from node.js itself. The examples were presented clearly, and they also helped me understand more about how they work and what they can be used for.  
Online videos about Node.JS also work, but not as well compared to the documentations. They do explain other examples outside of documentation about how to use what method for which program goal you have in mind, or simply for you to follow along.  
Going through online forums for specific issues takes very long due to having to also go through non-relevant or incorrect information provided.  
The documentation mixed with some online videos took the least time, since the videos typically show the screen and allows the user to follow along what is provided in said video, and the documentation has concise information that is easy to digest.  
Resources: <https://nodejs.org/docs/latest/api/>, <https://www.youtube.com/watch?v=ENrzD9HAZK4>, <https://www.freecodecamp.org/news/the-definitive-node-js-handbook-6912378afc6e/>.  
  
The Work Breakdown Structure that I use is fair yet could use more specific information, so I’ll add more info if needed.  
  
The Gantt chart that I have uses due dates that are intended to be close by to the due date, but then I realized that I can just space the times out further for the progress of the course, so I will do that next time.

**Discussion Forum 02 Post:**

In the past two weeks, I have better learned how to use Node.JS and its own test runner for use with assert and test cases, and have applied the tests to the program code as part of the assignments. I had also learned that:

* It is possible to import variables from other JavaScript files, rather than just methods/functions.
* I can utilize readline for making the program more visually efficient.
* I can save the record data from the program memory to a new csv file.

I have improved upon my knowledge in:

* How to organize the program structure.
* Using dedicated program code/scripts in other files to call upon.
* The practices in assertion for testing.

I found that the most interesting part of Node.JS and server-side JavaScript in this case was its use of multiple files in different folders to call upon via imports/requirements, due to its practical use with real project management, and I can have a better view at how the code works and therefore not get confused as to what function does what without having to go through much of the code.

Website resources continued to work best for me, specifically the node.js documentation, because it covers almost every method there is that is from node.js itself. The examples were presented clearly, and they also helped me understand more about how they work and what they can be used for. If I needed to gain insight on a module not covered in node.js, the documentation on npm or the module’s own documentation and info can be of service.  
Online videos about Node.JS still also work, but not as well compared to the documentations. They do at least explain other examples outside of documentation about how to use what method for which program goal you have in mind, or simply for you to follow along.  
Going through online forums for specific issues still takes very long due to having to also go through non-relevant or incorrect information provided.  
The documentation mixed with some online videos took the least time, since the videos typically show the screen and allows the user to follow along what is provided in said video, and the documentation has concise information that is easy to digest.  
Resources: <https://nodejs.org/docs/latest/api/>, <https://www.youtube.com/watch?v=ENrzD9HAZK4>, <https://www.freecodecamp.org/news/the-definitive-node-js-handbook-6912378afc6e/>.  
  
The Work Breakdown Structure that I use is once again fair, yet could use more detail, so I’ll add to that if needed. The process I have for creating one is, in my opinion, good enough to use; I can of course improve on the process if it is called for.  
The Gantt chart that I have uses due dates that are intended to be close by to the due date, but that resulted in that makes it neither overestimated nor underestimated. I kept getting denied to use the intended dates to match as closely to what I want as much as possible, so I will try to figure that out next time.

# References

|  |  |
| --- | --- |
| [1] | Node.JS, "JavaScript Asynchronous Programming and Callbacks," *Node.JS*, 03 July 2024. [Online]. Available: https://nodejs.org/en/learn/asynchronous-work/javascript-asynchronous-programming-and-callbacks. [Accessed 16 October 2024]. |
| [2] | E. Miquelito, "10 Powerful Node.js Libraries Every Developer Should Know About," *DEV Community*, 27 April 2023. [Online]. Available: https://dev.to/emiquelito/10-powerful-nodejs-libraries-every-developer-should-know-about-2c5g. [Accessed 16 October 2024]. |
| [3] | GeeksForGeeks, "Understanding Callbacks and Callback Hell in JavaScript," *GeeksForGeeks*, 13 August 2024. [Online]. Available: https://www.geeksforgeeks.org/what-to-understand-callback-and-callback-hell-in-javascript/. [Accessed 16 October 2024]. |
| [4] | S. Ghosh, "Node.js Architecture: The Single-Threaded Event Loop," *Sudipta.dev*, 21 April 2024. [Online]. Available: https://sudipta.dev/why-is-node-js-single-threaded/. [Accessed 16 October 2024]. |