WEB322 Assignment 1 (Ver 1.1)

# Submission Deadline:

Tuesday, January 21 @ 11:59pm

# Assessment Weight:

5% of your final course Grade (including week 1 and week 2 in-class assignments, 0.5% each)

# Objective:

In this assignment, students will create a Node.js application that uses the built-in "[fs](https://webprogrammingtoolsandframeworks.sdds.ca/Introduction/hello-world#fs)" and "[readline](https://webprogrammingtoolsandframeworks.sdds.ca/Introduction/hello-world" \l "readline)" modules to interact with files and directories. This program will read data from a file or directory specified by the user, analyze the content, and generate a report in the console.

## Step 1: Getting Started (Tools, Files & Directories)

To begin this assignment, make sure you have installed both [Visual Studio Code](https://code.visualstudio.com/) and [Node](https://nodejs.org/) (ie: you should be able to run programs written in JavaScript from the Integrated Terminal in Visual Studio Code using the command "node *filename.js*")

* Once you have the above tools installed, create a folder somewhere on your system to store the assignment.
* Download the **a1-files-2251.zip** file which contain "planets" directory, the "milky-way.txt" file and the "solar.txt" file. Unzip the file and place the "planets" directory and the two files within your newly-created assignment folder
* Open Your folder in Visual Studio Code to begin your assignment
* Create an "a1.js" file (this will be the file that your assignment is written in)
* Finally, your assignment folder should contain the files:

(Assignment Folder)  
 planets  
 - Earth.txt  
 - Jupiter.txt  
 - Mars.txt

- Mercury.txt  
 - Neptune.txt  
 - Saturn.txt  
 - Uranus.txt

- Venus.txt  
 a1.js  
 milky-way.txt  
 solar.txt

## Step 2: Processing the Current Directory

With our files / folders in place, we can begin editing a1.js. The first thing we must do is to read the files and folder(s) in the CURRENT (i.e. assignment folder) and list them in terminal. This can be done using the built-in "[fs](https://webprogrammingtoolsandframeworks.sdds.ca/Introduction/hello-world#fs)" module as mentioned in the notes.

If the user run the a1.js using Node .js, we must process to read the current ('.') directory to get the files and folders in the current directory and show a menu in terminal so the user can work with the menu in next step. The following is what we should show in terminal:

**A screenshot of a computer code

Description automatically generated**

**NOTE:** If the directory cannot be read, output the error to the console using "console.log(err.message);"

## Step 3: User Input

The second thing we must do is determine whether the user wishes to analyze a file or directory. This can be done using the "[readline](https://webprogrammingtoolsandframeworks.sdds.ca/Introduction/hello-world" \l "readline)" module as mentioned in the notes. The prompts for the user should be the following (sample user responses in **red**):  
  
**NOTE:** For now, we will simply allow the user to enter a file or directory NAME from the menu created in the last step (one case at a time):

* Enter a file or directory name from the list above: **solar.txt**
  + TODO: Process file ***solar.txt***
* Enter a file or directory name from the list above: **planets**
  + TODO: Process directory ***planets***
* Enter a file or directory name from the list above: **abc**
  + Invalid Input – TODO: show meaningful error message

## Step 4: Processing a File in the Current Directory

If the user enter a file name (e.g. "solar.txt" or “milky-way.txt”), then we must process the file name that the user entered ("solar.txt" from the example above) to generate the following report.  
  
**NOTE:** If the file cannot be read, output the error to the console using "console.log(err.message);"

Assuming that the following report was generated from the provided “solar.txt” file, the user should see the below information in the console (instead of the "TODO" output created in Step 3):

A white background with black text

Description automatically generated

**HINTS:** To determine a path is file of directory, we use:

* let stats = fs.statSync( "example.txt" );
* stats.isFile() or stats.isDirectory(); // return a Boolean value

To get the contents of the file as a string without any newline characters, the following code may be used:

* .toString().replace(/\s+/g, ' ');

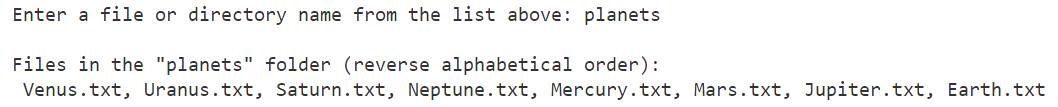
Similarly, to get an array of *words* from the file contents (string), the below code can be used:

* .replace(/[^\w\s\']/g, "").split(' ');

## Step 5: Processing a File in a Sub-Directory

If the user enters a directory name (e.g. "planets" for directory ***planets***), then we must show all files ~~and folders~~ in this sub directory (similar to Step 2) and show them in one line and separated with comma (,).

The sample output in terminal:



## Step 6: Using process exit event

Using Node.js process ‘exit’ event (handler) to output the current directory in operating system to terminal when the Node.js process is about to exit, i.e., show the following (as example) in terminal when the program stops after analyzing a file, showing files in a directory, or showing an error message:

**A screenshot of a computer program

Description automatically generated**

## Note: Hard Coding is Not Allowed

Hard coding will result in penalties or a zero-mark for the assignment. So, make sure that there is not hard coded code in your assignment 1. For example, if we change the file and directory names in the project (as shown below), the a1.js must keep working:

(Assignment Folder)   
 innerPlanets  
 - Earth.txt   
 - Mars.txt

- Mercury.txt

- Venus.txt  
 outerPlanets  
 - Jupiter.txt

- Neptune.txt  
 - Saturn.txt  
 - Uranus.txt

a1.js  
 space.txt

## Assignment Submission:

* Add the following declaration at the top of your a1.js file  
    
  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  \* WEB322 – Assignment 1  
  \* I declare that this assignment is my own work in accordance with Seneca Academic Policy.   
  \* No part of this assignment has been copied manually or electronically from any other source  
  \* (including web sites) or distributed to other students.  
  \*   
  \* Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Student ID: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  \*  
  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
* Compress (.zip) your assignment project folder (this is the folder that you opened in Visual Studio Code) with all files in it and submit the .zip file to My.Seneca under **Assignments** -> **Assignment** 1

## Important Note:

* **NO LATE SUBMISSIONS** for assignments. Late assignment submissions will not be accepted and will receive a **grade of zero (0)**.
* Submitted assignments must run locally, ie: start up errors causing the assignment/app to fail on startup will result in a **grade of zero (0)** for the assignment.