Assignment Node.js

Repo: <https://github.com/globant-ui/NodeJsWorkshopPune16>

Every participant should fork this repo and commit the assignment code in his/her own forked repo. On the last date of the assignment, a pull request should be raised as a final submission of the assignment.

## Day 1

### Objective:

This lab session involves **File IO**, Handling of **JSON Strings**, and **XML Documents** in Node.js.

A flat file contains student records in JSON format. Write a program to parse and process these student records.

What You Learn?

* Basic Javascript and Node.js Syntax.
* JSON Basics.
* XML Basics.
* Concept of Callbacks in Node.js.
* How to perform File IO Operations in Node.js.
* How to process JSON strings in Node.js.
* How to generate XML documents in Node.js.
* How to create package.json file, how to add dependencies in it.
* How to create your own module
* How to use third party Node modules

Requirements: Part 1:

* Assume, there is a JSON file (**source.json**) that contains student records along with *scores* of each student such as:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | {  "students":  [  {  "id": 123,  "fName": "John",  "lName": "Doe",  "score": 234  },  {  "id": 124,  "fName": "Jane",  "lName": "Doe",  "score": 543  },  {  "id": 125,  "fName": "Jackie",  "lName": "Doe",  "score": 453  }  ]  } |

* Write a NodeJS program which can read this source.json file. It will then extract the *first name, last name,* and *score* for every student from this JSON file.
* The program should write each record to a output file (**destination.txt**) in the following format:

|  |  |
| --- | --- |
| 1  2  3  4 | First Name | Last Name | Score  123 | John | Doe | 234  124 | Jane | Doe | 543  125 | Jackie | Doe | 453 |

* We extend the Part 1 to  **sort** all the students based on their score (**descending** sort).

* The program should write the sorted output to the destination.txt file as follows:

|  |  |
| --- | --- |
| 1  2  3  4 | Id | First Name | Last Name | Score  124 | Jane | Doe | 543  125 | Jackie | Doe | 453  123 | John | Doe | 234 |

Requirements: Part 2:

* We extend the Part 2 to generate an XML output in the destination.xml file as follows:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | <students>  <student id="124">     <name>Jane Doe</name>     <score>543</score>  </student>  <student id="125">     <name>Jackie Doe</name>     <score>453</score>  </student>  <student id="123">     <name>John Doe</name>     <score>234</score>  </student>  </students> |

* Note the sequence of records in the generated XML should be in the **descending order of the score**.
* Make sure you use a DOM generation library and **do not** construct the output XML by simple string concatenations. The intent here is to learn to use a DOM generation library.

Package Your Work

* Write a README.txt for your project and commit that as well.

* **Code Quality:**
  + Did you have a coding guideline document that you followed?
  + Is your code neatly formatted and indented?
  + Is your code refactored into a few methods that a clear interface / responsibility each?
  + Do all your methods have a method-level comment at the top?
  + Does your source file have a class-level comment at the top?
  + Did you get someone else to review your code during development?

* **User Experience:**
  + Does your program print the source file and destination file names to assist the user?
  + Are validation errors or any data-format errors neatly shown to the users?
  + Did you run a spell check on all your text, messages and errors shown to the user?

* **Validations and Defensive Programming:**
  + Does your code check if the source file actually exists?
  + Did you check if the source file has a valid JSON? How does your program behave if the source file does not have a valid JSON?
  + Did you check if the source file has the expected “keys” within the JSON? How does your program behave if any expected key is missing?
  + Does your code check if the destination file already exists? What do you do if the destination already exists?
  + If you are overwriting the destination file, are you showing a warning message to the user accordingly?
  + Does your code check if the destination directory is writeable?
  + What does your program do if the destination file cannot be created / written? Does it elegantly fail with an error?

* **Testing:**
  + Find the test cases that will validate your program
  + Does your test report identify which test cases pass and which ones fail?

Test Scenarios

How does your program behave if…

* It is unable to find / read the source file?
* If the source file does not have a JSON string in it?
* If the source file contains JSON, but is not in the expected structure?
* If certain keys are missing in the student records (say, score is missing)?
* If you are unable to write to the destination file due to lack of write permissions?
* If the source file has a very, very, large number of records. Say a million records? Does your program still function as expected?
* If your source file has non-english characters?