

**19F CST8276\_010 Advanced Database Topics**

**Vi Thi Phuong Pham**

**Exercise 03**

Table of Contents

[1. Evidence of Learning 3](#_Toc20743107)

[2. Research 5](#_Toc20743108)

[3. Program Demonstration via Screen Shot 6](#_Toc20743109)

[4. Source code 10](#_Toc20743110)

[References 13](#_Toc20743111)

# Evidence of Learning

1. **Variable**

Create variables with const and let

*//regex expression for (,)*

const regex = /,(?=(?:[^"]\*"[^"]\*")\*[^"]\**$*)/;

*//print out my name*

console.log("Name: Vi Thi Phuong Pham - 040886894");

*//path to the csv file*

let path = "canadianCheeseDirectory.csv";

let max = 5;

let counter = 0;

From lines 11 to 19 in exercise 3.

1. **Loops**

I am using a simple For loop and an enhanced For loop in this example

*//loop through totalrows to get data with selected columns*

*//put it into rows. So now rows is perfect data*

*for* (let i = 0; i <= max; i++) {

      let row = [];

*for* (const col of columnIndex) {

        row.push(totalRows[i][col]);

      }

      rows.push(row);

    }

From lines 67 to 73 in exercise 3.

1. **File input-output**

In this exercise, I use fs, readline modules along with createReadStream and createInterface method to read the file line by line.

*//import fs, readline*

const fs = require("fs");

const readline = require("readline");

let readFile = fs.createReadStream(path);

let rl = readline.createInterface({ input: readFile });

From lines 7, 8 and 54, 55

1. **Simple Data structure**

I use multiple arrays in the exercise like array totalRows (line 44), rows (line 47), header (line 49), data (line 52)

*for* (let i = 0; i <= max; i++) {

      let row = [];

*for* (const col of columnIndex) {

        row.push(totalRows[i][col]);

      }

      rows.push(row);

    }

*//put the first array of rows into header*

    header = rows[0];

*//put the leftover data into data array*

    data = rows.slice(1);

This code above indicate how I use arrays to handle data. From line 67 to 78

1. **Methods**

Throughout the exercise, I have used different built in methods including (map, for each, push, split, slice…)

*//put the leftover data into data array*

    data = rows.slice(1);

*//convert rows into objects*

    const cheeseRecordList = data.map((rowData, rowIndex) => {

*//create object obj*

      const cheeseRecord = {};

*//for each header, put data into object*

      header.forEach((headerName, colIndex) => {

        cheeseRecord[headerName] = rowData[colIndex];

      });

*//print out the object*

*// console.log(cheeseRecord);*

*return* cheeseRecord;

    });

    console.log(cheeseRecordList);

  }

I use map for array data in line 81, slice for data in line 78, forEach for array header in line 85.

# Research

1. **Memory management**

Nodejs memory is managed by V8. V8 is a JavaScript engine at first made for Google Chrome, yet it can likewise be utilized as an independent. V8 compiles JavaScript down to local code and executes it. During execution, it deals with the distribution and freeing of memory as required. Like Java Virtual Machine, V8 uses a similar scheme called Resident Set and it divides the memory into segments:

* Code - the real code being executed
* Stack - contains all value types with pointers point to the objects on the heaps
* Heap- a memory segment committed to holding reference types. [1]

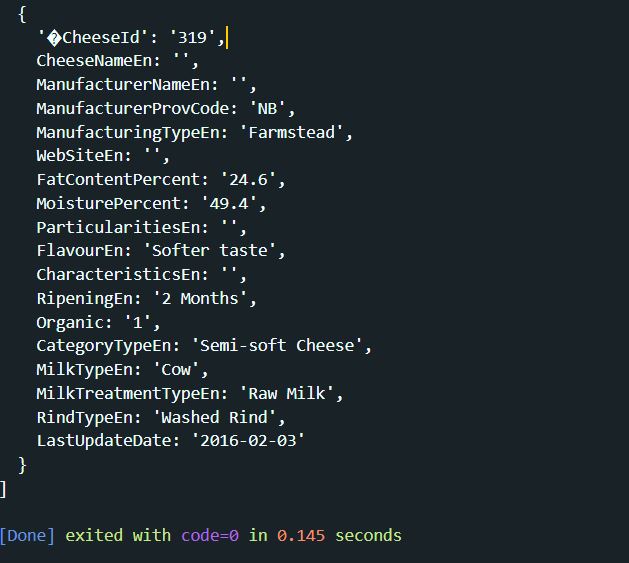
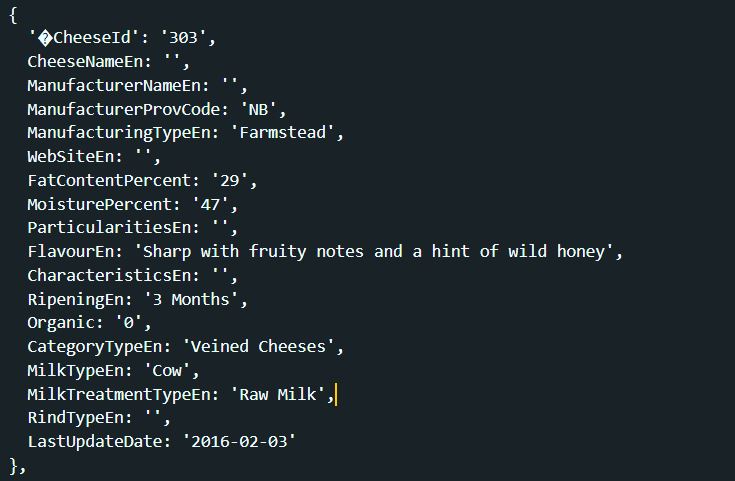
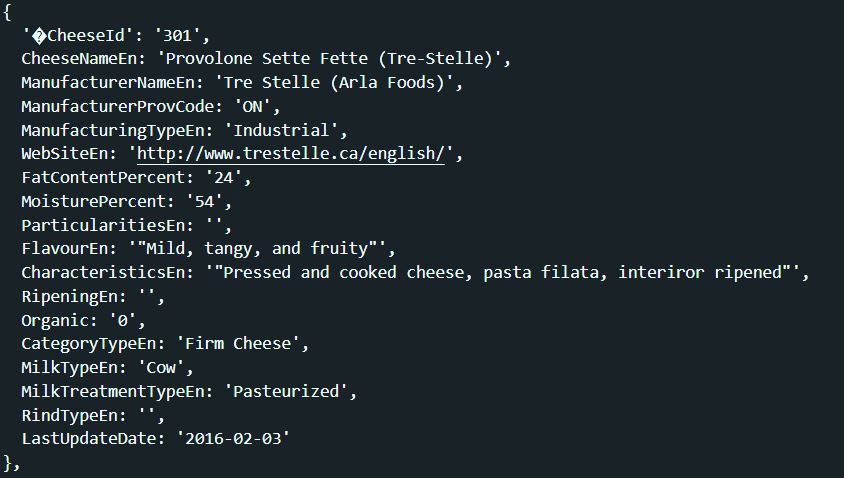
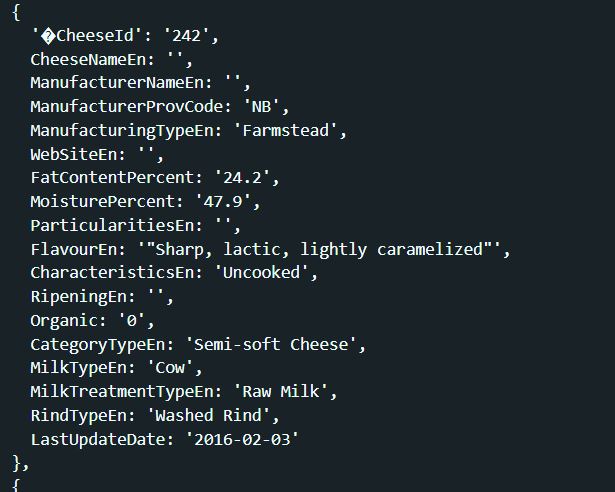
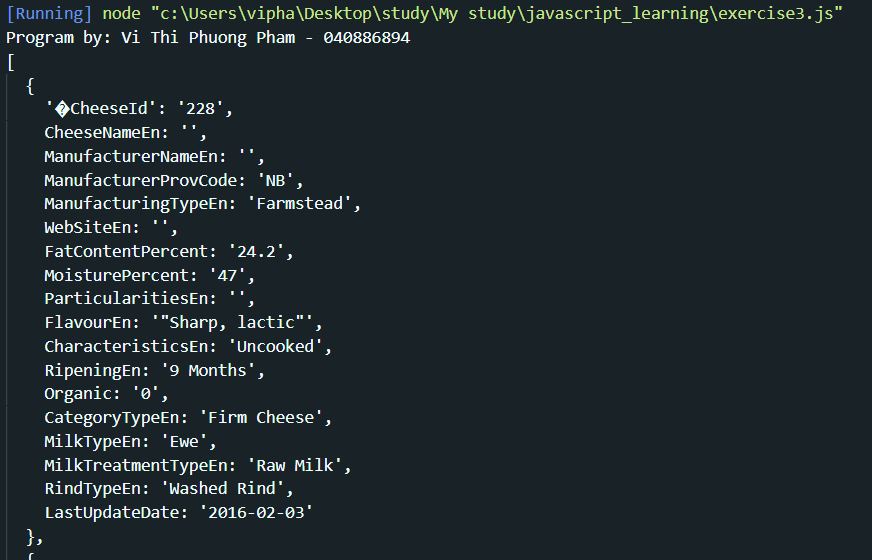
V8 also uses a notable component called garbage collection for memory leak problems. The hypothesis behind the garbage collection is very straightforward: If a memory portion isn't referenced from anyplace, we can accept that it isn't utilized and, hence, can be freed [1].

Nodejs JavaScript is dynamically typed since I don’t need to declare the type of the variable, the type will be determined at runtime.

1. **Unit tesing**

UnitJs is the Unit testing for JavaScript. It provides multiple libraries, functions for you to write your unit tests in the way you want. It has some similar methods like JUnit such as isEqualTo(), assert(), equal(), notEqual(), etc. In the path of Nodejs JavaScript learning, I will use Unitjs.com as a main resource page for unit testing. [2].

# Program Demonstration via Screen Shot



# Source code

/\*\*

\* @fileOverview Using file-IO to read and collect 5 records from a CSV file. Create 5 object in memory with that 5 records and print it out.

\* @author Vi Thi Phuong Pham

\*/

//import fs, readline

const fs = require("fs");

const readline = require("readline");

//regex expression for (,)

const regex = /,(?=(?:[^"]\*"[^"]\*")\*[^"]\*$)/;

//print out my name

console.log("Program by: Vi Thi Phuong Pham - 040886894");

//path to the csv file

let path = "canadianCheeseDirectory.csv";

let max = 5;

let counter = 0;

//create an array of selected column index

let columnIndex = [

0,

1,

3,

5,

6,

8,

10,

11,

12,

14,

16,

18,

20,

21,

23,

25,

27,

29

];

//create an array of rows

let totalRows = [];

//create an array of rows with selected columns

let rows = [];

//create header(column name)

let header = [];

//create an array of data without header

let data = [];

let readFile = fs.createReadStream(path);

let rl = readline.createInterface({ input: readFile });

rl.on("line", d => {

//put the line into an array

totalRows.push(d.split(regex));

if (counter++ >= max) {

//stop reading

rl.close();

rl.removeAllListeners();

//loop through totalrows to get data with selected columns

//put it into rows. So now rows is perfect data

for (let i = 0; i <= max; i++) {

let row = [];

for (const col of columnIndex) {

row.push(totalRows[i][col]);

}

rows.push(row);

}

//put the first array of rows into header

header = rows[0];

//put the leftover data into data array

data = rows.slice(1);

//convert rows into objects

const cheeseRecordList = data.map((rowData, rowIndex) => {

//create object obj

const cheeseRecord = {};

//for each header, put data into object

header.forEach((headerName, colIndex) => {

cheeseRecord[headerName] = rowData[colIndex];

});

//print out the object

// console.log(cheeseRecord);

return cheeseRecord;

});

console.log(cheeseRecordList);

}

});

# References

|  |  |
| --- | --- |
| [1] | D. Khan, "Understanding Garbage Collection and Hunting Memory Leaks in Node.js," CodeShip, 03 05 2017. [Online]. Available: https://blog.codeship.com/understanding-garbage-collection-in-node-js/. [Accessed 30 09 2019]. |
| [2] | "UnitJs introduction," UnitJs, [Online]. Available: https://unitjs.com/guide/introduction.html. [Accessed 30 09 2019]. |