COMP1231
Web Programming

Assignment #2

Due Date: Friday, April 9th, 11:59 pm

Weight: 15% of Final Grade

Table of Contents

COMP1231 Assignment 2

Assignment Type: Individual Assignment
Due Date: Friday April 9th, 2021 @ 11:59pm (Late Submission @ **10%** per day)

1. **Description**

   In this assignment, you are to develop a JavaScript file that contains the implementation of **10 functions**. Each function is represented as one step, and marks will be awarded on the completion of each step (each function).

   Each function is independent and solves a unique problem, as such, treat and implement each function in isolation of the others, that's is, you should only focus on one problem at a time.

   There are two versions of this assignment (Version A and Version B). The last two digits of your student id, determine which set of functions (set of 10 steps) you must implement.

   PLEASE NOTE: Input validation is NOT required for any function parameter – You may assume that all input parameters provided to your functions, are all valid.

2. **Objective**
   - Write decision-making statements and control structures to solve problems
   - Apply programming logic to solve basic to intermediate problems
   - Testing and debugging

3. **Learning Outcomes:** After conducting this assignment students will / will be able to:

   1. Acquainted with implementing JavaScript functions
   2. Follow instructions, and to implement various requirements

4. **Instructions:**

   1. You are to create ONE JavaScript file which must be named **<*STUDENT_ID*>-functions.js**, where ***STUDENT_ID*** represents your student number.

   2. The implementation for your **10** JavaScript functions must be completed within your **<*STUDENT_ID*>-functions.js** file.

      You are **NOT** permitted to use any external files or libraries. Using any external libraries will result in a final grade of ZERO for your assignment submission.

   3. Your JavaScript functions file **must** be configured for strict mode ("**use strict**")

   4. Please note the function names used in this document are used to demo the examples and you must apply the best practices by naming your functions something meaningful

5. **What to Implement, Version A or Version B?**

   Version A - Last two Digits of Student #: 00 to 50
   If the last two digits of your Student # are within the range 00 to 50 (inclusive), you are required to do version A of this assignment (**Function 1A – Function 10A**)

   Version B - Last two Digits of Student #: 51 to 99
   If the last two digits of your Student # are within the range 51 to 99 (inclusive), you are required to do version A of this assignment (**Function 1B – Function 10B**)

Function 1A: Unique Array

```
function _one(array)
```

Create a JavaScript function that meets the following requirements:

- Please give your function a descriptive name
  - (_one is not acceptable, and is only displayed here for illustration purposes)
- Receives an array of integers as an argument
- The function removes all duplicates (if they exist) from the array and returns it to the caller.
- Assume the input array parameter will have at least one element in it.

Examples:

```
_one([33])              ➔ [33]
_one([33, 33, 1, 4])    ➔ [1, 4]
_one([33, 33, 1, 4, 1]) ➔ [4]
```

Function 2A: Fail / Pass Averages

```
function _two(array)
```

Create a JavaScript function that meets the following requirements:

- Please give your function a descriptive name
  - (_two is not acceptable, and is only displayed here for illustration purposes)
- Receives an array of numbers as a parameter, with each integer being a mark out of 100.
- The function should return an array consisting of all the averages less than 50, followed by the average of all the mark more than 50.
  - both averages should be rounded to the nearest whole number.
  - If there are no failing marks (< 50), then the average fail mark should be set to -1
  - If there are no passing marks (< 50), then the average pass mark should be set to -1

Examples:

```
_two([63, 65, 33]))       ➔ [33, 64]
_two([63, 62, 100, 100])) ➔ [-1, 81]
_two([33, 42, 20, 10]))   ➔ [26, -1]
```

Function 3A: Check Date

```
function _three(string)
```

Create a JavaScript function that meets the following requirements:

- Please give your function a descriptive name
  - (_three is not acceptable, and is only displayed here for illustration purposes)
- Receives a string as a parameter.
- The function returns a Boolean indicating if the parameter string passed, is a valid date
  - True indicates a valid date was provided
  - False indicates that a valid date was not provided.
- A valid date string, will have only the full month name (ex "January" not "Jan") and a day number
  - A year will not be provided and thus your logic need not be concerned in that regard.
- The string parameter can contain any number of spaces, but the month name must always start at the first non-space character from the beginning of the string.
- The day number part of the date string to be tested could contain alphabetic characters and thus making it invalid.
- You may assume February only has 28 days in it.

Examples:

```
_three("January     21"))        ➔ true
_three("Auust 3"))               ➔ false
_three(" June   15B "))          ➔ false
_three("February 0"))            ➔ false
_three(" December 3K1"))         ➔ false
_three("February 29"))           ➔ false
_three("    February    28 "))   ➔ true
```

Function 4A: How many Days Between

```
function _four(str_date1, str_date2)
```

Create a JavaScript function that meets the following requirements:

- Please give your function a descriptive name
  - (_four is not acceptable, and is only displayed here for illustration purposes)
- Create a function that takes two valid dates as arguments
- The function returns the number of days between str_date1 and str_date2

Examples:

```
_four("June 14, 2021", "June 20, 2021")         ➔ 6
_four("December 29, 2021", "January 1, 2022") ➔ 3
_four("July 20, 2021", "July 30, 2021")         ➔ 10
```

Function 5A: Swap Characters

```
function _five(string, c1, c2)
```

Create a JavaScript function that meets the following requirements:

- Please give your function a descriptive name
  - (_five is not acceptable, and is only displayed here for illustration purposes)
- That take three arguments, a string, character 1 and character 2
- The function replaces all instances of c1 with c2, and vice versa
- The function returns the updated string

Examples:

```
_five( "aabbccc", "a", "b")                    ➔ "bbaaccc"
_five("random w#rds writt&n h&r&", "#", "&") ➔ "random w&rds writt#n h#r#"
_five("128 895 556 788 999", "8", "9")         ➔ "129 985 556 799 888"
```

Function 6A: Move Capital Letters

```
function _six(string)
```

Create a JavaScript function that meets the following requirements:

- Please give your function a descriptive name
  - (_six is not acceptable, and is only displayed here for illustration purposes)
- That takes in a string parameter, of mixed casing (mix of upper and lowercase letters)
- The function moves all capital letters to the front of a word.
- The uppercase letters moved to the front, maintain their original relative order
- The lowercase letters moved to the back front, maintain their original relative order

Examples:

```
_six("hApPy")      ➔ "APhpy"
_six("moveMENT")   ➔ "MENTmove"
_six("shOrtCAKE") ➔ "OCAKEshrt"
```

Function 7A: Leading Zeros

```
function _seven(string)
```

Create a JavaScript function that meets the following requirements:

- Please give your function a descriptive name
  - (_seven is not acceptable, and is only displayed here for illustration purposes)
- That takes in a string number as an argument
- The function returns the number removing any trailing and/or leading zeros.
- Trailing zeros are the zeros after a decimal point which don't affect the value of the number
- Leading zeros are the zeros before a number which don't affect the value of the number

Examples:

```
_seven("230.000") ➔ "230"
_seven("00402")   ➔ "402"
_seven("03.1400") ➔ "3.14"
_seven("30")      ➔ "30"
```

Function 8A: Sort Me

```
function _eight(arr[], str)
```

Create a JavaScript function that meets the following requirements:

- Please give your function a descriptive name
  - (_eight is not acceptable, and is only displayed here for illustration purposes)
- That takes in an array of integers, and a string ("Asc", "Des", "None")
- The function returns the array of number according to the following rules
  - "**Asc**" returns a sorted array in ascending order
  - "**Des**" returns a sorted array in descending order
  - "**None**" returns an array without any modification

Examples:

```
_eight([4, 3, 2, 1], "Asc" )  ➔ [1, 2, 3, 4]
_eight([7, 8, 11, 66], "Des") ➔ [66, 11, 8, 7]
_eight([1, 2, 3, 4], "None")  ➔ [1, 2, 3, 4]
```

Function 9A: Repeating Characters

```
function _nine(str)
```

Create a JavaScript function that meets the following requirements:

- Please give your function a descriptive name
  - (_nine is not acceptable, and is only displayed here for illustration purposes)
- That takes in a string
- The function returns the first character that repeats
- If there is no character that repeats, return -1
- The function should be case sensitive (case sensitive "I" not equal to "i")

Examples:

```
_nine("legolas") ➔ "l"
_nine("Gandalf") ➔ "a"
_nine("Balrog")  ➔ "-1"
_nine("Isildur") ➔ "-1"
```

Function 10A: Capitalize first Letter of Each Word

```
function _ten(string)
```

Create a JavaScript function that meets the following requirements:

- Please give your function a descriptive name
  - (_ten is not acceptable, and is only displayed here for illustration purposes)
- That takes in a string as an argument
- The function converts first character of each word to uppercase
- The function returns the newly formatted string

Examples:

```
_ten("This is a title")        ➔ "This Is A Title"
_ten("capitalize every word") ➔ "Capitalize Every Word"
_ten("I Like Pizza")           ➔ "I Like Pizza"
_ten("PIZZA PIZZA PIZZA")      ➔ "PIZZA PIZZA PIZZA"
```

Function 1B: Common array elements

```
function _one(array1[], array2[])
```

Create a JavaScript function that meets the following requirements:

- Please give your function a descriptive name
  - (_one is not acceptable, and is only displayed here for illustration purposes)
- That takes two sorted arrays of numbers (both in ascending order) as arguments
- The function returns a single array of numbers, that contains only the numbers that are common in both arrays

Examples:

```
_one([-1, 3, 4, 6, 7, 9], [1, 3])              ➔ [3]
_one([1, 3, 4, 6, 7, 9], [1, 2, 3, 4, 7, 10]) ➔ [1, 3, 4, 7]
_one([1, 2, 2, 2, 3, 4, 5], [1, 2, 4, 5])      ➔ [1, 2, 4, 5]
_one([1, 2, 3, 4, 5], [10, 12, 13, 15])        ➔ []
```

Function 2B: Temperature Converter

```
function _two(array[], string)
```

Create a JavaScript function that meets the following requirements:

- Please give your function a descriptive name
  - (_two is not acceptable, and is only displayed here for illustration purposes)
- That takes two parameters
  - The first parameter is an array with both a temperature type (string) and temperature included within it
  - The second parameter is a temperature type (string)
- The function returns the temperature (in the array) converted to the desired temperature type (second parameter)
- The temperature types passed are one of are:
  - Celsius, Fahrenheit, and Kelvin

Examples:

```
_two(["Fahrenheit", 3] , "Kelvin")    ➔ 257.0
_two(["Celsius", 10] , "Fahrenheit") ➔ 50.0
_two(["Celsius", 20] , "Kelvin")      ➔ 293.2
_two(["Celsius", 25] , "Celsius")     ➔ 25
```

Function 3B: Sum missing numbers

```
function _three(array[])
```

Create a JavaScript function that meets the following requirements:

- Please give your function a descriptive name
  - (_three is not acceptable, and is only displayed here for illustration purposes)
- That takes an array of positive integers
- The function returns the sum of the numbers missing from the given array.
- The range of numbers to consider when searching for the missing number in the array is, the sequence of consecutive numbers between the min and max numbers inclusive to the argument array.

Examples:

```
sumMissingNumbers([4, 3, 8, 1, 2])           ➔ 18    // 5 + 6 + 7 = 18
sumMissingNumbers([17, 16, 15, 10, 11, 12]) ➔ 27    // 13 + 14 = 27
sumMissingNumbers([1, 2, 3, 4, 5])           ➔ 0     // No Missing Numbers
```

Function 4B: Even Index, Odd Index

```
function _four(array[])
```

Create a JavaScript function that meets the following requirements:

- Please give your function a descriptive name
  - (_four is not acceptable, and is only displayed here for illustration purposes)
- The function takes an array of numbers as an argument
- Returns a Boolean based on the following:
  - returns **true** if every even index in the array, contains an even number and every odd index in the array contains an odd number.
  - returns **false** if otherwise.

Examples:

```
_four([2, 7, 4, 9, 6, 1, 6, 3]) ➔ true
_four([2, 7, 9, 1, 6, 1, 6, 3]) ➔ false
_four([2, 7, 8, 8, 6, 1, 6, 3]) ➔ false
```

Function 5B: Remove Duplicates

```
function _five(array[])
```

Create a JavaScript function that meets the following requirements:

- Please give your function a descriptive name
  - (_five is not acceptable, and is only displayed here for illustration purposes)
- That takes an array of items (numbers or strings) as argument
- The function removes all duplicate items in the array
- The function returns a new array in the same sequential order as the original source array (minus the duplicates)

Examples:

```
_five([1, 0, 1, 0])                → [1, 0]
_five(["The", "big", "cat"])       → ["The", "big", "cat"]
_five(["John", "Taylor", "John"]) → ["John", "Taylor"]
```

Function 6B: Censor words

```
function _six(string)
```

Create a JavaScript function that meets the following requirements:

- Please give your function a descriptive name
  - (_six is not acceptable, and is only displayed here for illustration purposes)
- That takes a string (sentence) as an argument
- The function censors' words over four characters with "*"
  - Do not censor words with exactly four characters
  - If all words have four characters or less, return the original string
  - The amount of "*" is the same as the length of the word being censored

Examples:

```
_six("The code is forty")       → "The code is *****"
_six("Two plus three is five") → "Two plus ***** is five"
_six("aaaa aaaaa 1234 12345")  → "aaaa ***** 1234 *****"
```

Function 7B: Reverse the order of words with 5 Letters

```
function _seven(string)
```

Create a JavaScript function that meets the following requirements:

- Please give your function a descriptive name
  - (_seven is not acceptable, and is only displayed here for illustration purposes)
- That takes a string of one or more words as an argument
- The function returns the same string, but with all five or more letter words reversed

Examples:

```
_seven("Reverse")                      ➔ "esreveR"
_seven("This is a typical sentence.") ➔ "This is a lacipyt .ecnetnes"
_seven("The dog is big.")              ➔ "The dog is big."
```

Function 8B: Unique Character String

```
function _eight(arr[])
```

Create a JavaScript function that meets the following requirements:

- Please give your function a descriptive name
  - (_eight is not acceptable, and is only displayed here for illustration purposes)
- The function takes an array of strings as an argument
- The function returns a single array, containing only the strings from the original input array, that have only unique characters (have no repeating characters) within them.
- You can assume case-sensitivity in our logic, that is, **'a' != 'A'** and thus are both distinct from one another.

Examples:

```
_eight(["abb", "abc", "abcdb", "aea", "bbb"])  ➔ ["abc"]
_eight(["88", "999", "989", "9988", "9898"])   ➔ []
_eight(["ABCDE", "DDEB", "BED", "CCA", "BAC"]) ➔ ["ABCDE", "BED", "BAC"]
```

Function 9B: Find Unique Numbers in Array

```
function _nine(array[])
```

Create a JavaScript function that meets the following requirements:

- Please give your function a descriptive name
  - (_nine is not acceptable, and is only displayed here for illustration purposes)
- The function takes an array of integers as an argument
- The function locates unique numbers from the input array, and returns them as an array (array of unique values) back to the caller.
- The order of the resultant array is not a concern.

Examples:

```
_nine([2, 2, 2, 1, 3, 4, 4, 4])                    ➔ [1, 3]
_nine([2])                                         ➔ [2]
_nine([])                                          ➔ []
_nine([7, 13, 3, 14, 6, 5, 4, 6, 4, 13, 5, 3, 18]) ➔ [7, 14, 18]
```

Function 10B: Fix Spacing

```
function _ten(string)
```

Create a JavaScript function that meets the following requirements:

- Please give your function a descriptive name
  - (_ten is not acceptable, and is only displayed here for illustration purposes)
- The function takes a string (sentence) as an argument
- The string argument may or may not have added additional back spaces added to it.
- The function should return the corrected sentence by removing these added extra spaces.
  - The correct and returned sentence, should have all words separated by only one space, and there should be no spaces at the beginning or end of the sentence.

Examples:

```
_ten("The film   starts       at      midnight. ")
➔ "The film starts at midnight."

_ten("The     waves were crashing  on the     shore.   ")
➔ "The waves were crashing on the shore."

_ten(" Always look on    the bright   side of  life.")
➔ "Always look on the bright side of life."
```

Submission Procedure and Rules

Please ensure to remove all instances of the following from your final submission solution:

- console.log
- document.write()
- innerHTML()
- alert()
- irrelevant comments or any commented code
- calls to your functions

Do NOT use the following built-in Functions

You are not permitted to use any JavaScript built-in functions for this project. Below is a sample list some built-in functions that should not be used:

String built-in functions	Array built-in functions
endsWith()	concat()
includes()	copyWithin()
indexOf()	every()
lastIndexOf()	fill()
localeCompare()	filter()
match()	find()
repeat()	findIndex()
replace()	forEach()
search()	indexOf()
slice()	isArray()
split()	join()
startsWith()	lastIndexOf()
substr()	map()
substring()	pop()
toLocaleLowerCase()	push()
toLocaleUpperCase()	reduce()
toLowerCase()	reduceRight()
toString()	reverse()
toUpperCase()	shift()
trim()	slice()
valueOf()	some()
trimLeft()	sort()
trimRight()	splice()
unshift()	toString()
valueOf()	

Submission Procedure

1. In order to complete this assignment, you will need to create the following file.

    I. **<*STUDENT_ID*>-functions.js**
    As mentioned in the assignment instructions, this is where your functions will be implemented. A template of this file has been provided to you.

2. Late submissions are graded at a **-10%** penalty per day

We are going to use Moss (https://theory.stanford.edu/~aiken/moss/) to detect similarities in code among all students. There will be zero tolerance for plagiarism. Please be aware.

All extra code (unrelated to the assignment requirement) or commented code, must be removed from your files.

There should be no calls to your functions. Our tester will call your functions based on the function names listed in main.html file.

Marking Rubric

40% - Your overall code and functionality
40% - Logic
20% - Best practices

- Variables/function name can be short, yet descriptive/meaningful, camelCasing applied
- Use let/const over keyword var when possible
- Avoid globals
- Stick to a strict coding style
- Comment as much as needed but not more
- Clean and easy to read code (indentation, space before and after any operator)

```
For example:

if(userName=="sam"){allow=true;

console.log("something"):

}

//is more readable when written as:

if(userName == "sam"){

    allow = true;

    console.log("something"):

}
```

GOOD LUCK!