Assignment
1

COMP1231 Web Programming

Summer 2018

Due Date: Tuesday, July 31st, 2018 @ 11:59:59

Introduction

It's time to put all your JavaScript skills to the test to build an app combining everything you've learned about JavaScript so far. In this assignment, you are required to build 10 functions.

Goals and Outcomes

- Reverse engineering the built-in functions and built user define functions
- Write decision-making statements and control structures to solve problems
- Apply programming logic to solve basic to intermediate problems
- Testing and debugging

Description

In this assignment, you are going to create a JavaScript file containing 10 functions. Each function is worth 10 marks. The functions you will be creating might be very similar to a JavaScript built-in function. Therefore, you are not allowed to use most of the built-in functions. The list of functions that are not allowed to be used is provided at the end of this document.

Each function needs to be considered as an individual problem.

There are two versions of the assignment. The last two digits of your student id will determine the set of functions you need to implement. No mark will be given to those who compete the wrong version.

What to implement?

- Create a JavaScript file named **STUDENT_ID-functions.js** (i.e 101010332-functions.js), that will contain all your functions definition

- An HTML file named main.html that would reference your JavaScript file, in order to call and test each function.

- You are not allowed to use any other external file or library. Using an extra file or library result in zero marks for your assignment.

If the last two digits of your Student ID ends with: 00 to 49

1. Write a function to return a string that contains random numbers where the length must be equal to the value of the parameter. The second parameter contains an array of numbers to omit.

   E.G. function(5, [1,7]) returns “20243” (1 or 7 is not present)

2. Write a function that makes the following calculation:

   param1 + param2 - param3 + param4 – param5…and keeps with this pattern. If any non-numerical character is encountered, ¼ of its value is computed as the numerical value. The result should truncate precision.

   E.G. function(11,a,2,30,Z,3) returns 35

   11 + (97/4) - 2 + 30 - (122/4) + 3

3. Write a function that accepts 3 arrays and returns a string of duplicate elements in ascending order. Return all alphabetic characters in lowercase.

   E.G. function ([1,2,3,d], [5, 3, 0, a], [A,d,9] returns “3,d”

4. Write a function that accepts an array and a number. The function adds the number value to each element of the array and returns a string representation of the new series of characters separated by comma. If the sum of the ASCII value of the element + number is the same type (odd or even) as the original ASCII value of the element, increment the value otherwise don't change it.

   E.G.

   function([2,5,7,C, Z], 5) returns “ 2, 5, 7, C, Z “

   - 2 is even, 2+5 is odd. Even and odd are not the same type so no change
   -5 is odd, 5+5 is even. Odd and even are not the same type so no change
   -7 is odd, 7+5 is even. Odd and even are not the same type so no change
   -C = 67, which is odd, 67+5 is even. Odd and even are not the same type so no change
   -Z = 90, which is even, 90+5 is odd. Even and odd are not the same type so no change

   function([2,5,7,C, Z], 2) returns “4, 5, 7, E, \ “

5. Write a function that accepts an array. The function adds each element's value to the value of the character before it to generate a number. It returns a string of numbers separated by comma. If the value is a digit, use its numerical value. Otherwise, use the character's ASCII value.

   E.G. function([z, C, 2,5,7]) returns "129,189,69,7,12 "

6. Write a function that accepts two arguments (array and a number). The function would iterate through the array to find numbers that are divisible by the number passed as second argument. The function returns a string containing the index positions of divisible elements separated by comma. Return string should not end with comma. Make sure the second argument is a Number greater than zero. Otherwise, the function should not return anything.:

   E.G. function([3, 2, 0, 5, 9, 24, 888, 10], 2); returns: 1, 5, 6, 7

   E.G. function([0, , k, 5, 19, 240, 99], 5); returns: 3, 5

7. Write a function that accepts three arguments (an array, delimiter, and a number). The function would join content of the array, inserting the delimiter after every nth elements of the array and returns the result as string:

   E.G. function([0, 2, "Hello", 7, "TO", "---"], "&", 2); returns: 02&Hello7&TO---&

   function([0, 2, "Hello", 7, "TO", "---"], "+", 3); returns: 02Hello+7TO---+
   function([0, 2, "Hello", 7, "TO", "---"], "*", 4); returns: 02Hello7*TO---

   - if number of elements is 5 [1,2,3,4,5] and delimiter is & and n is 2 return string must be: 12&34&5
   - if number of elements is 6 [1,2,3,4,5,6] and delimiter is & and n is 2 return string must be: 12&34&56&

8. Write a function that accepts two arguments, an object and an array. The array contains property names that may exist in the object. Search object for the properties and if found, concatenate values of those properties, separated by comma and return as string.
   E.G. function({name: "Al", age: 22, setName: function(a){this.name = a} },

   ['name', 'age']);

   returns: Al,22

   E.G. function({name: "Tom", age: 18}, ['name', 'phone', "address"]);

   returns: Tom

9. Write a function that would accept an string and checks if any character is repeated more than once. The function would return the repeated chars and the number of times they are repeated. The return value is a string, all pairs separated by comma.

   E.G. function(“2abc**zac22”); returns: 2=3,a=2,c=2,*=2,

10. Write a function that converts the alphabetical characters of first half of each argument to uppercase and returns the result as one string. Number/s and special chars are ignored. If the length of string is not even, append an extra “-“ (hyphen) to end of it to make it even.

    E.G. function(“We&are*well”, “2, That is gr8.”, 33, [‘abcd’, 3, ‘4k2go’]);

    returns: WE&ARE*well-2, THAT is gr8.-33,ABcd34K2go-

If the last two digits of your Student ID ends with: 50 to 99

1. Write a function to return a string that contains random lowercase alphabetic characters where the length must be equal to the value of the parameter. The second parameter contains an array of letters to omit.

   E.G. function(8,[t,b]) returns “eicdiofd”

2. Write a function that accepts a string. The function takes the last digit of the ASCII code and returns a string of digits sorted in ascending order

   E.G. function("zDbc") returns “2,8,8,9”

   (z => 122, D=>68, b=>98, c=>99)

3. Write a function that makes the following calculation:

   param1 * param2 / param3 * param4 / param5…and keeps with this pattern. If any non-numerical character is encountered, ¼ of its value is computed as the numerical value. The result should truncate precision.

   E.G. function(11,a,2,30,Z,3) returns 393

   11 * (97/4) / 2 * 30 / (122/4) * 3

4. Write a function that accepts 3 arrays and returns a string of the elements sorted in ascending value, with duplicates removed. Return all alphabetic characters in lowercase.

   E.G. function([1,2,3, d], [5, 3, 0, a], [a,d,9] returns "0,1,2,3,5,9,a,d"

5. Write a function that accepts an array. The function subtracts value of the character next to it to generate a new character. It returns a string of new characters with a string length of one less than the parameter. If the character is a digit, use its numerical value. Otherwise, use its ASCII value.

   E.G. function([z, C, 2,5,7]) returns "7, A, - ,   . "

6. Write a function that excepts three arguments. First argument will be the string that needs to be manipulated. Second argument would be a character that needs to be selected and moved to the left/right. And the third argument is a number that indicates how many positions we need to move the selected char in the specified direction. If the number is positive we will move to the right and if it is negative, move to the left. You will only need to search and select the first occurrence of the char. Empty spaces should be used as space filler when the number is greater than length of the string. The function would return the modified string as in example below:

   E.G. function("This is #2test", "#", 5); returns: This is 2test#
   E.G. function("This is #2test", "#", -12); returns: #   This is 2test

7. Write a function that accepts an string. Checks if any character is repeated more than 3 times consecutively. The function would remove the excessive repeats of that char's instance and returns the result as a new string.

   E.G. function("45555aaabbbbb78KKKKL"); returns: 4555aaabbb78KKKL

8. Write a function that accepts two string as arguments. Searches the first string for each letter passed in second string. If the characters are found, reverse the letter's case (uppercase/lowercase). The function would return the updated string.

   E.G. function("4%AbCd -jd", "ajd"); returns: 4%abCD –JD

   E.G. function("4%abcD -d", "ajd"); returns: 4%Abcd –D

9. Write a function that would get rid of all none alphanumeric characters from the string and return the cleaned string:

   E.G. function("4%Aa**/;22Cd-"); returns: 4Aa22Cd

   E.G. function("@#~4%*(){}|{{["); returns: 4

10. Write a function that accepts two arguments (string and a number). The function will check each letter/digit of the string. If it is a Number, divisible by the number that is passed to the function, it would be moved to the beginning of the string. The function returns the modified string:

    E.G. function(“435a7892L0L”, 2); returns: 48235a79L0L

    function(“0134 69*”, 3);   returns: 369014 *

Do not use the following functions:

You are not allowed to use any of the built-in functions listed below. Using any of the following function will result in zero marks for your assignment.

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()`
`trimLeft()`	`some()`
`trimRight()`	`sort()`
valueOf()	`splice()`
	`toString()`
	`unshift()`
	`valueOf()`

Submission deadline and rules

- Due date: Tuesday July 31 2018 @ 11:59:59PM. It must be posted on your GBlearn account
- 10% penalty per day
- Assignment must be submitted in the following path: public_html/comp1231/assignment/
  - In the directory above, you should have the files:
    - main.html
    - STUDENT_ID-functions.js (i.e 101010332-functions.js)

Assignment tester will be released a week before due date. You must use assignment tester before submitting your assignment. I will go over this (using the tester) in one of our lab/lecture or you can ask Mobi-Help tutors.