Lab 6: String Stuff

This lab gives you the chance to code JavaScript that manipulates strings. Worth mentioning is that the exercises are a bit silly but they do give you a chance to use common JavaScript string functions.

There are three exercises to this lab:

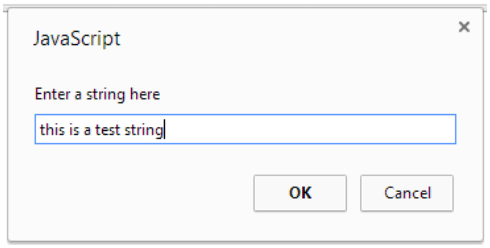
Code JavaScript that *extracts the unique characters from a string*
Code JavaScript that *lists all substrings of a string*
Code JavaScript that *sorts characters in a string using ol' mr. Bubble*

These exercises are easier to code when using *arrays* but we didn't get to that part of the class yet.

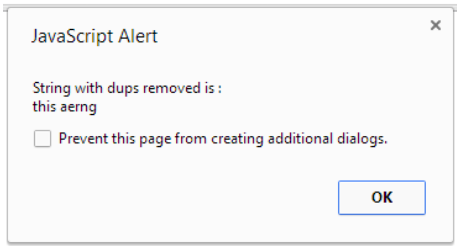
Let's take a look at each of these problems:

Extract Unique Characters

Here's what the program execution should resemble:



After you enter a string, you'll see:

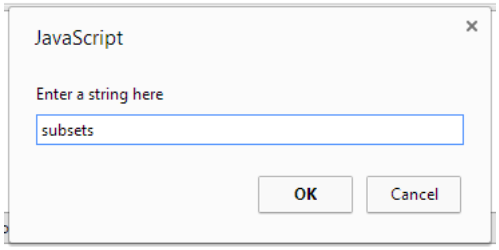


Use the file *removedup.html* to get started. This file has the alerts/prompts coded and some hints.

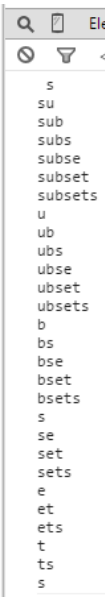
You'll have to code *at least three lines* to finish this exercise.

List Substrings of a String

Here's what the program execution should resemble:



After you enter a string, you'll see:

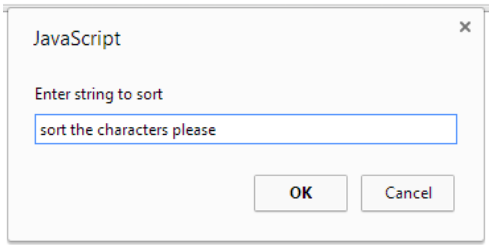


Use the file *substr.html* to get started. The hints should guide you on your way to success.

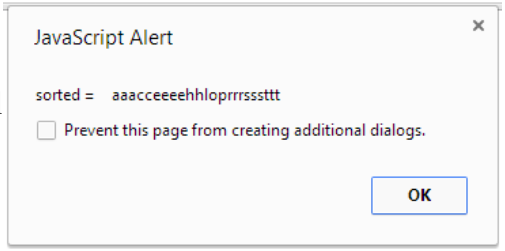
You'll need *at least three lines* to finish this exercise.

Bubble Sort String Characters

This exercise is a bit tougher; requiring more lines of code. Here's the outputs:



After you enter a string, you'll see:



As an aside, the *bubble sort is terrible* but makes for a good coding exercise.

Anyway, here's the algorithm expressed in pseudocode:

```
procedure bubbleSort( A : list of sortable items )
   n = length(A)
   repeat
     swapped = false
     for i = 1 to n-1 inclusive do
       /* if this pair is out of order */
       if A[i-1] > A[i] then
         /* swap them and remember something changed */
         swap( A[i-1], A[i] )
         swapped = true
       end if
     end for
   until not swapped
end procedure
```

The bubble sort (and just about all sorts) is usually done on *arrays* but here, you'll use characters in a string.

The *list of sortable items* will be the string you enter.

The *array references* shown in the pseudocode will be *string functions that do the same thing.* For example, the reference A[i] will 'translate in string-speak' to the ***ith character in the string.***

The ***swap*** function referenced in the pseudocode exchanges the characters in the ith and (i+1)th positions. You'll need to think of a way to do this with strings. Allow me to offer a suggestion:

Given a string ABCDEFG and the goal of swapping ther characters in, say, positions 2 and 3 may be simpler if we view the string as follows:

The original string – String up to POS 2 followed by character at POS 2 followed by character at POS 3 followed by the rest of the string (AB + C + D + EFG). Recall the first character (A) is at POS 0.

The string with ***swapped characters*** **–** String up to POS 2 followed by char at ***POS 3*** followed by char at ***POS 2*** followed by rest of string (AB + ***D*** + ***C*** + EFG).

Oh – one more point worth mentioning; the pseudocode statement:

```
repeat
  // Buncha stuff
until not swapped
```

May be mapped to a JavaScript statement that *loops with test of condition done at the* ***bottom of the loop***.

Start with the file *bubblestring.html*. No hints in the file; just HTML for the prompts/alerts. The pseudocode is all the help you get on this one.