Lab 7: Counting Words

This lab gives you the chance to code JavaScript that manipulates strings, objects and arrays.

In short, your program uses a *predefined* string containing about 230 words. From this string, you'll create tables showing the:

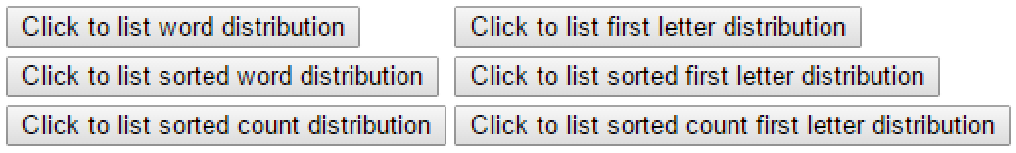
Count of each word
Count of each word *sorted by word order*
Count of each word *sorted by count order*
Count of each word *by its first letter*
Count of each word *by its first letter sorted by that letter*
Count of each word *by its first letter sorted by letter count*

You'll have a file *countwords.html* that has some prewritten code with mucho comments. You also have a file *wordsandtablefunction.js* that contains the string containing the 230+ words you'll use to construct the table and a function that creates the HTML table.

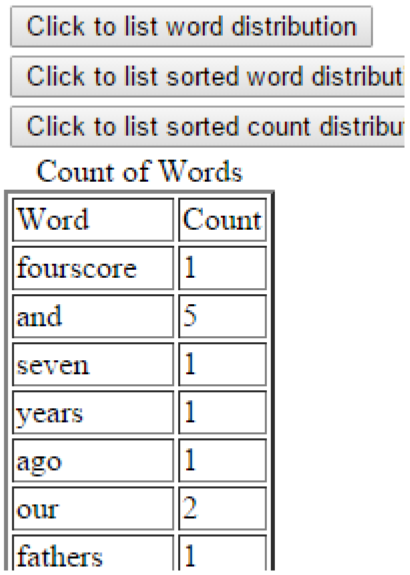
The solutions are included in your Lab 9 folder; you may want to run it and examine the outputs. We'll look at SOME of the outputs then examine your starter files.

Program Outputs

When you run the program, you'll see:


Click to list word distribution
Click to list sorted word distribution
Click to list sorted count distribution
Click to list first letter distribution
Click to list sorted first letter distribution
Click to list sorted count first letter distribution

Click on 'list word' and you'll see this table under the buttons:

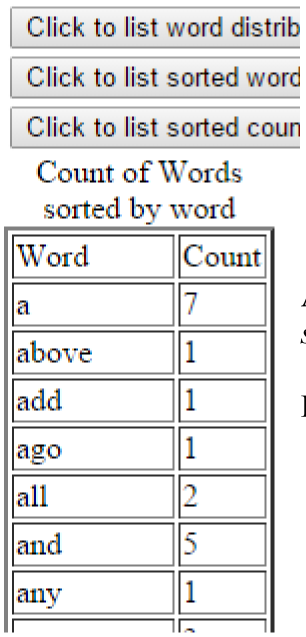

Click to list word distribution
Click to list sorted word distribut
Click to list sorted count distribu

Count of Words

Word	Count
fourscore	1
and	5
seven	1
years	1
ago	1
our	2
fathers	1

You're not seeing the whole table here (about 70 rows)

Click on 'sorted word' and you'll see:


Click to list word distrib
Click to list sorted word
Click to list sorted coun

Count of Words
sorted by word

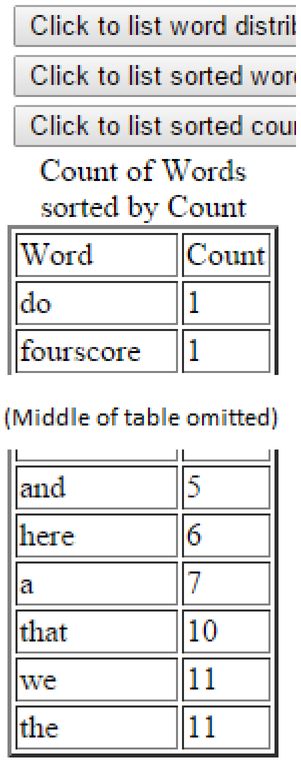
Word	Count
a	7
above	1
add	1
ago	1
all	2
and	5
any	1

Notice the previous table is gone, replaced by this table.

Also note the heading (caption, really) has changed to tell you the items are *sorted.*

Finally, note the items are *really sorted by word*.

Click on 'sorted count' and you'll see:



Count of Words
sorted by Count

Word	Count
do	1
fourscore	1

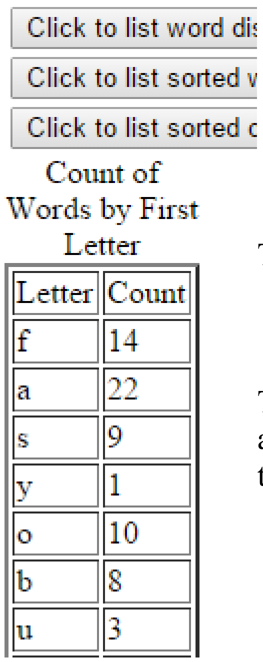
(Middle of table omitted)

Word	Count
and	5
here	6
a	7
that	10
we	11
the	11

The middle of the table is omitted so you can see the table is sorted by *word counts* not just a table of words and ones.

We'll just show one more:

Click on 'first letter distribution'; you'll see:



Count of Words by First Letter

Letter	Count
f	14
a	22
s	9
y	1
o	10
b	8
u	3

The *headings and caption* are changed to show what we're doing.

The 'sorted first letter' and the 'sorted count first letter' are similar to what we've already seen; no need to see these tables. One difference worth noting is that ALL the letter tables have no more than 26 rows (There's no entries for 'q', 'x', 'z')

Let's take a look at the files provided and talk about their contents.

The Provided Files

You have *two* files provided; *countwords.html* and *wordsandtablefunction.js*.

The HTML file contains the HTML that displays the buttons and the *onclick handlers* that run your JavaScript code when you click the buttons. This file also contains *skeletomn JavaScript* and *lots of comments* to help you get this lab done.

The JS file contains the *string containing the words* and the *function that creates the tables showing the counts*.

We'll look at the JS file first:

wordsandtablefunction.js

The JS file contains a variable *words* that is a long string:

```
// This variable contains a list of words that is used to list the word distribution
var words = "Fourscore and seven years ago our fathers brought forth upon this continent a new nation " +
            "conceived in liberty and dedicated to the proposition that all men are created equal " +
            "Now we are engaged in a great civil war testing whether that nation or any nation so conceived " +
// LOTS OMITTED!!!!
            "by the people for the people shall not perish from the earth" ;
```

Your HTML file references this variable; the variable is **global** (known everywhere in your JavaScript)

The other 'piece' of the JS file is a *function* named *showWords* that creates the HTML table displayed upon button clicks. The function accepts three arguments:

> The table headers as an *array of three strings*
> The *keys* used to provide content for the first column and to *access data used in the second column*
> The *object containing the data* that provides content for the second column

Your JavaScript in the HTML file will *call this function, passing the arguments used to display the table*.

The showWords() function generates a string *that contains the HTML and data* for the table. The last line of the function uses our friend *document.getElementById( )* to access the part of the HTML page to be updated with the table display.

You'll have to code *one and a half* statements in this function. We'll talk about that soon.

countwords.html

Here is where you'll spend most of your time. This file contains the *precoded HTML* for the buttons/event handlers and the *stubs/comments for your JavaScript*.

Here's the HTML; remember, this is already provided for you but we'll discuss its workings anyway:

```
<body onload="collectWordDistribution( );">
<table>
<tr>
  <td><input type="button" onclick='showWordDistribution( )'
                           value="Click to list word distribution" /></td>
  <td><input type="button" onclick='showFirstLetterDistribution( )'
                           value="Click to list first letter distribution" /></td>
</tr>
<tr>
  <td><input type="button" onclick='showSortWordDistribution( )'
                           value="Click to list sorted word distribution" /></td>
  <td><input type="button" onclick='showSortLetterDistribution( )'
                           value="Click to list sorted first letter distribution" /></td>
</tr>
<tr>
  <td><input type="button" onclick='showSortCountDistribution( )'
                           value="Click to list sorted count distribution" /></td>
  <td><input type="button" onclick='showSortCountLetterDistribution( )'
                           value="Click to list sorted count first letter distribution" /></td>
</tr>
</table>
<!-- Here's where the table will go  -->
<div id="words"></div>
</body>
```

First, notice we call a function ***collectWordDistribution( )*** when the *page loads*. This is an example of a *JavaScript event handler* that responds to a *system event*, not a *user event*. Put simply, once the HTML page loads (renders the HTML, parses out the JavaScript) this function is called. As the comment in the page states we do not NEED to use the *onload* handler but it's an example of how we can take action based on a system event.

Let's take a look at selected lines of code in your HTML file.

These lines:

```
    var wordCounts         = {} ;          // Holds counts by word
    var wordCountsByLetter = {} ;          // Holds counts of first letter of words
```

are your *objects* that hold the information created by *collectWordDistribution( )*. IOW, *collectWordDistribution* accesses the *word* global variable coded in the JS file, counts words and the words by first letter and saves this info in the two variables cited above. BTW, these two variables *are global.*

The data in these variables are *indexed by word (wordCounts) or by the first letter of the word (wordCountsByLetter)*. The structure (or will be, after *collectWordDistribution( )* completes):

```
    wordCounts["are"]       = # times 'are' appears in the string
    wordCountsByLetter["g"] = # times a word beginning with 'g' appears in the string
```

Of course, "are" and "g can be *any word in the string* and *any first letter* respectively.

IMPORTANT: JavaScript usually uses ***dot notation*** to access *object property values*. In this lab, you'll be *passing arguments for your property values* which JavaScript doesn't understand with dot notation. Ergo, you'll use ***bracket notation as shown above.***

Let's look at the onload( ) function *collectWordDistribution( )* and see how your function will save these counts.

The *collectWordDistribution( )* function uses a *local function named saveCounts( )*. This local function will be called from *collectWordDistribution( )* ***twice***: once to save the count for a word and once to save the count for the first letter of the word. The *saveCounts( )* function accepts two arguments: the *key* into the object holding the counts (word or letter) and the *object that holds the counts* (wordCounts or wordCountsByLetter). The *logic* you'll code can be done with *two lines* or a single *if/else statement*. Comments in the HTML file describe the logic.

Next in *collectWordDistribution( )* you want to *create an array of blank-delimited words* from the string variable *words*. You should convert the words to *lowercase* (don't want 'Are' and 'are' to be considered separate words for this exercise!) and use a string function to compose the array. The assignment to variable *arrayOfWords* is already coded; you need to code the appropriate JavaScript functions to perform the conversion.

Finally in *collectWordDistribution( )* you need to examine each word in *arrayOfWords* and call the local function *saveCounts( )* TWICE: first, pass the *word and the object containing the word counts;* second, pass the *first letter and teh object containing the counts for first letters*.

Recall that *collectWordDistribution( )* executes *after your page loads*. Ergo, when you see the buttons on your HTML page, your *wordCounts* and *wordCountsByLetter* objects should be 'filled' with data.

The Mouse Click Event Handlers

The other functions bolded/italicized/supersized respond to *button clicks.* Yeah, there's six functions and all this JavaScript seems daunting but the routines that create the tables (call the *showWords* function) are very similar. Note that ALL six functions have the *array of strings for the headers* **already coded**. Your job in *each* of these functions is to:

> *Extract* the required keys (word, letter, count, sorted, unsorted)
> *Pass* the extracted keys and object containing the counts to the function *showData( )*.
> Recall *showData( )* is coded in the JS file and *creates the table*

One other point worth mentioning: *Two* tables show the data *sorted by counts* (word and letter counts). Sorting on a ***data field*** is different from sorting on a ***key field***. We discussed the use of *sort functions* in JavaScript; the required sort functions needed for sorting data fields is already coded.