CSC230 Assignment – Week 5/6

Part I: Read and try the code in the following two chapters from "Eloquent JavaScript 3rd edition (2018)" https://eloquentjavascript.net/ No submission is required but part of future quiz may be based on these chapters:

Chapter 14: The Document Object Model
Chapter 15: Handling Events

Part II: Complete the following exercises from "Eloquent JavaScript 3rd edition (2018)" https://eloquentjavascript.net/

1.

FIZZBUZZ (page 38)

Write a program that uses `console.log` to print all the numbers from 1 to 100, with two exceptions. For numbers divisible by 3, print `"Fizz"` instead of the number, and for numbers divisible by 5 (and not 3), print `"Buzz"` instead.

When you have that working, modify your program to print `"FizzBuzz"` for numbers that are divisible by both 3 and 5 (and still print `"Fizz"` or `"Buzz"` for numbers divisible by only one of those).

(This is actually an interview question that has been claimed to weed out a significant percentage of programmer candidates. So if you solved it, your labor market value just went up.)

2.

BEAN COUNTING (page 56)

You can get the Nth character, or letter, from a string by writing `"string"[N]`. The returned value will be a string containing only one character (for example, `"b"`). The first character has position 0, which causes the last one to be found at position `string.length - 1`. In other words, a two-character string has length 2, and its characters have positions 0 and 1.

Write a function `countBs` that takes a string as its only argument and returns a number that indicates how many uppercase "B" characters there are in the string.

Next, write a function called `countChar` that behaves like `countBs`, except it takes a second argument that indicates the character that is to be counted (rather than counting only uppercase "B" characters). Rewrite `countBs` to make use of this new function.

3.

BUILD A TABLE (page 241)

An HTML table is built with the following tag structure:

```
<table>
  <tr>
    <th>name</th>
    <th>height</th>
    <th>place</th>
  </tr>
  <tr>
    <td>Kilimanjaro</td>
    <td>5895</td>
    <td>Tanzania</td>
  </tr>
</table>
```

For each *row*, the `<table>` tag contains a `<tr>` tag. Inside of these `<tr>` tags, we can put cell elements: either heading cells (`<th>`) or regular cells (`<td>`).

Given a data set of mountains, an array of objects with `name`, `height`, and `place` properties, generate the DOM structure for a table that enumerates the objects. It should have one column per key and one row per object, plus a header row with `<th>` elements at the top, listing the column names.

Write this so that the columns are automatically derived from the objects, by taking the property names of the first object in the data.

Add the resulting table to the element with an `id` attribute of `"mountains"` so that it becomes visible in the document.

Once you have this working, right-align cells that contain number values by setting their `style.textAlign` property to `"right"`.

You can use `document.createElement` to create new element nodes, `document.createTextNode` to create text nodes, and the `appendChild` method to put nodes into other nodes.

You'll want to loop over the key names once to fill in the top row and then again for each object in the array to construct the data rows. To get an array of key names from the first object, `Object.keys` will be useful.

To add the table to the correct parent node, you can use `document.getElementById` or `document.querySelector` to find the node with the proper `id` attribute.

Hints:

Build a table

You can use document.createElement to create new element nodes, document. createTextNode to create text nodes, and the appendChild method to put nodes into other nodes. You'll want to loop over the key names once to fill in the top row and then again for each object in the array to construct the data rows. To get an array of key names from the first object, Object.keys will be useful. To add the table to the correct parent node, you can use document.getElementById or document.querySelector to find the node with the proper id attribute.

```
<!doctype html>
<style>
 /* your style sheet is here for a better looking table, this is optional */
 /* … */
</style>

<body>
<script>
  var MOUNTAINS = [
    {name: "Mt. Rainer", height:14417, Place: Washington State},
    {name: "Mt. Adams", height:12281, Place: Washington State},
    {name: "Mt. Baker", height:10786, Place: Washington State},
  ];

 function buildTable(data) {
   var table = document.createElement("table");
   // your javascript code here to create a table
   // ….
   return table;
 }

 // invoke the buildTable function:
  document.body.appendChild(buildTable(MOUNTAINS));
</script>
</body>
```