Workshop 4

After working as a group to look at the debugger and the DOM Inspector in Chrome, attempt to build this program in class. There are some tricky steps that are likely to produce unexpected or incorrect results, if not errors, the first time that you attempt them. The goal here is to learn to examine the parts of your program to figure out where things have gone wrong.

Start from **Workshop4Starter.zip**, which can be downloaded from Week 4 Lectures. I have written the HTML file and the first three (correct) lines of the JavaScript file for you. You will write and debug the rest. You can test each step using console.log() and/or the DOM inspector (which will show you the dynamic elements that you've created that won't show up using "View Source")

1. Create a nested set of loops. The outer loop, **i**, should count from 0 to (but not including) **numRows**. The inner loop, **j**, should count from 0 to (but not including) **numColumns**.

2. Each time through the outer loop **i**, before inner loop **j** begins, use **document.createElement()** to create a new table row. Assign the newly created element to variable **theRow.** Then make **theRow** a child of **theTable** using **appendChild().** This step should create three table rows.

3. Within inner loop **j**, do the following:

    a. Create a new **td** element and assign it to the variable **theCell**. The idea of a "column" doesn't really exist in the DOM model for a table, so calling it a "cell" is a good reminder.

    b. Make **theCell** a child of **theRow.** At this point, if you test the script, you should have 3 cells per row for 3 rows, though they will be invisible since we haven't created any CSS styling. This is intentional, to make you use the DOM Inspector.

    c. Create a new **button** element and assign it to the variable **theButton.**

    d. Set the **id** attribute of **theButton** to the value of **j** followed by an underscore plus the value of **i**. This means the upper-left button will have an **id** of "0_0" and the lower-right button will have an **id** of "2_2".

    e. Make **theButton** a child of **theCell**

    f. Using **document.createTextNode()**, create a textNode that contains "Button" plus a Number (the capitalization of Number here is a hint!) that counts how many buttons have been created, starting from 0 and going to 8, reading across then down. There are two ways to do this step. One involves creating a separate "counter" variable, but I want you to try the other method, which is to figure out how to calculate a combination of **i, j,** and **numColumns** that will count from 0 to 8. This is one of the tricky steps that you may need to debug! Assign the textNode to variable **theText**.

The result should look like this screen shot:

g. Make **theText** a child of **theButton**.

h. Add an Event Listener to **theButton** that will, when the button is clicked, call a function called **changeMyName**. This is a straightforward Event Listener setup, since we don't need to pass any variables to the function.

4. After the end of your two loops, write a stub for the function **changeMyName**. This simply means writing the function with nothing inside it or with a simple console.log (like a "got here" message) to the console. The idea is to be able to test that everything works without worrying about what goes in this function. As long as the function exists, it will work.

5. Once you have everything else working, let's do the final step and actually change the buttons' names. This will require you to put together your knowledge of variable types, DOM text elements, and String manipulation in one place. This function's body (meaning the stuff inside the curly braces) will be three lines long unless you choose to break up the first step into smaller units.

    a. For the button that was just clicked, get the last character of the button's name, convert it to a number, and assign that number to variable **myNumber**. Some hints:

        i. When a function is called by an Event Listener, **this** refers to the object that received the event -- in this case, whichever Button was pressed.

        ii. Because we know there is nothing else inside of these Buttons, the cleanest way to get the Button's text is via the **innerText** property.

        iii. The **charAt()** method works on any String and returns the character at a specific position in the String, with 0 being the first character (just as with an Array the first element is index 0)

        iv. Just as with an Array, the last item of a String is always one less than its **length**, which is conveniently a built-in property for both.

6. You might want to test that you're getting the number that you expect (0 for the first button, 8 for the last button) before proceeding.

7. Add 1 to **myNumber**.

8. Set the **innerText** of your Button (again using **this** to refer to the Button) to "Button " plus **myNumber.**

9. If you did the last few steps correctly, clicking on any button should increase the button's number by 1. A common bug here is to discover that your buttons are doing something funny like adding a digit to the end of the number (so "1" becomes "11" instead of actually increasing). If things are working correctly, then any Button that reaches 10 will cycle back to 1.

10. DISCUSSION QUESTION: Why do the buttons cycle back to 1 after they reach 10? We did not write any code to say "Go back to 1", so they should go to 11, 12, 13, etc., right? There is a very good reason that this is happening, and it has to do with how we are extracting the number from our Button name. If you have extra time, you can try to solve this problem by manipulating the Button text in a different way, but the important thing is just to understand why this phenomenon is happening, since it does not look like an error, but it is not what we intended to have happen! Bugs like this are the subtle ones that you must watch closely when programming! The debugger is great, but human checking of the results is always necessary!