EECS 428 / ECE 578
DATA VISUALIZATION
Spring 2015

ASSIGNMENT 1

Due Date: *Thursday, March 26th, 2015, 17:00*
(10 Points)

Assignment Submission: *Turn in your assignment by the due date through LMS. Prepare and upload* ***one zip file*** *that should have* ***one subdirectory for each part of the assignment*** *(PartA & PartB). Name the zip file as <your first name>_<your last name>_ assignment1. See individual questions for what you should return.*

You can and encouraged to discuss HTML, CSS, SVG, JavaScript and D3 with each other. ***However, all work in questions (implementation) must be your own; you must neither copy from nor provide assistance to anybody else (including online resources). If you need guidance for any question, talk to the instructor or TA.***

PART A: *JavaScript Warm-Up*

Before starting solving the following exercises, you are encouraged to go through this basic tutorial from Mozilla Developer Network MDN. This network provides an elegant documentation for JavaScript. It will be very helpful for you as you get acquainted more with JavaScript in case you are new to the JavaScript.

QUESTION 1

Simple Sort. Write a function that takes an array of numbers, sorts them, and finally prints them to the console.

Name your function *simpleSort*. The signature of the function is:
simpleSort(array);

QUESTION 2

Simple Shift. Write a function that takes an array and shift its elements one position. The function should take two arguments: first parameter is the array and second parameter is the shift direction (left or right). Use 'R' for right and 'L' for left. The shift behavior is circular (i.e, after a right shift, the rightmost element becomes the leftmost or vice versa).

Name your function *simpleShift*. The signature of the function is:
simpleShift(array, direction);

QUESTION 3

DizVizz. Write a program that uses console.log to print all the numbers from 1 to 100, with these exceptions:

- For the numbers divisible by 3 (but, not divisible by 5), print "Diz" instead of the number
- For the numbers divisible by 5 (but, not divisible by 3), print "Vizz" instead
- For the numbers that are divisible by both 3 and 5, print "DizVizz"

Name your function *dizVizz*. The signature of the function is:
divVizz();

QUESTION 4

ChessBoard. Write a function that creates a string that represents an NxN grid, using newline characters ('\n') to separate lines. Each position of the grid is represented by either a space or a "#" character. The string should consist of alternating sequences of space and '#', effectively forming a chess board-like pattern.

Passing this string (for N=8) to console.log should show something like this:

```
  # # # #
 # # # #
  # # # #
 # # # #
  # # # #
 # # # #
  # # # #
 # # # #
```

Name your function *chessBoard*. The signature of the function is:
chessBoard(N);

QUESTION 5

Sorting Words. Write a function that takes a sentence as a parameter, sorts the sentence *words* alphabetically and outputs an array of objects in the following format:
[{word: a, Position: x}, {word: b, Position: y}, …]

Position is the index of the word in the sentence.

Example:

Input string: "Data Visualization Class"
Position: 0 1 2

Output:

[
{Word: “Class”,
Position : 2},

{Word: “Data”,
Position : 0},

{Word: “Visualization”,
Position : 1}
]

Name your function *sortSentence*. The signature of the function is:
sortSentence(sentence);

You may find these resources useful:
MDN Arrays

MDN Objects

IMPORTANT. Please return in PartA directory:

- A ***single*** JavaScript file that includes functions for all 5 questions. Name it *<your first name>_<your last name>_assignment1_partA*.js. The content of the file should look like:

```
function simpleSort(array){

// your code here

}

function simpleShift (array, direction){

// your code here

}

function dizVizz(){

// your code here

}

function chessBoard(N){

// your code here

}

function sortSentence(sentence){

// your code here

}
```

PART B: *Web-based calculator*

Calculator design and implementation. You are asked to build a simple web-based calculator, just like the one you use daily in this assignment. The calculator should include all basic arithmetic operations: *addition, subtraction, multiplication and division* (feel free to add other operations). Refer to the figure 1 for a sample layout. All input operands and operations are entered through calculator buttons so you do not need to add keyboard support. As you enter a number, it should show up on calculator screen. In this question, you will use HTML and CSS (and SVG if you prefer) to design the calculator and JavaScript will enable you to implement the functionality.

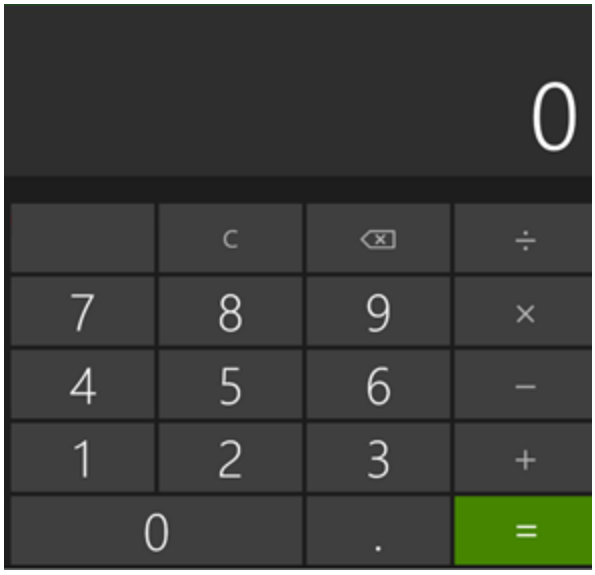

Figure 1. Basic Calculator

D3 Selection and Styling. After completing the design and implementation of the calculator, create a small rectangular palette of colors that displays under the calculator. Once a user clicks on a particular color, the background of the calculator should change accordingly. ***You should use D3 for this task.*** *Of course, the task could also be achieved without D3*. The default background color is gray as in the figure. The palette could consists of the rainbow colors as shown below (feel free to use different colors as long as you have a large selection of colors)

Figure 2. Rainbow Colors Palette

Bar Chart for Operands and Result. Finally, after each operation, say addition or multiplication, you are asked to visualize the operation in a *3-bars* bar chart as can be seen in the sample layout in Figure 3. Displays the operands in the first two bars, and the result of the

operation in the third. This should apply, of course, only to binary operations in case you decide to implement other operations. Make sure that the size of the area you use for showing bar-chart is fixed and the operands & result scale nicely within the limited area. Remember that the input can be very large or very small, so you have to account for that in your bar chart and scale accordingly. Do not forget to add labels along both axes (1st operand, 2nd operand, result). Although it may seems evident, it is a good practice. *Although this part can also be completed without using D3*, ***you are required to use D3.***

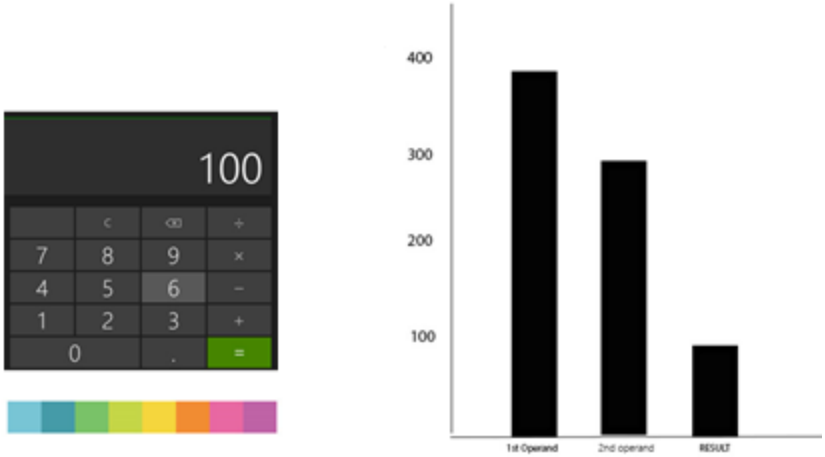

Figure 3. The Big Picture. This is how your final HMTL page should resemble

Notes:

- You are not obliged to produce the exact images shown in the illustrations above. In fact you are encouraged to use your creativity to create your own designs as long as it supports the intended functionality and the visually it is at least as appealing as the examples ☺
- You can write all of your code in a single HTML file, but it is highly discouraged. Try to separate your code files: HTML, JavaScript and CSS.

IMPORTANT. Please return in PartB directory:

- All your HTML, JavaScript and CSS files (you can have sub-directories under partB if you prefer). Your main HTML file should be named as ***index.html***. When we open *index.html* in our web browser (*Chrome* will be used for grading), your calculator and visuals should show up in the main page.