Exercise03: Javascript

Objectives:

To learn to use Javascript objects, functions, and closures – and put them to use!

Work with your group (or by yourself). Each group is to upload only one submission.

1. Warm Up: Try Some Examples

1. First, open blackboard, go to Course Contents, and then download exercise03.zip file into your workspace (U:\workspace or something like that!). Then, unzip.

   Play with each of the given examples (in examples directory). Open them using a text editor of your choice and modify parts of the html or js files to learn how the different instructions work. Please do the TODO segments for each example.

   NOTE01: One suggestion (to help you play with javascript) is to use online Javascript code tool like http://codepen.io/pen/ or https://jsbin.com. They are very useful for trying javascript examples as you can change the html or javascript directly on the website, and you can immediately see the results of your changes.

   NOTE02: You will need to also learn how to use the **available tools for JS debugging**.

   - Firefox has tools->WebDeveloper->Debugger,
   - **Chrome** has Tools->Developer Tools (ctrl-shift-I).

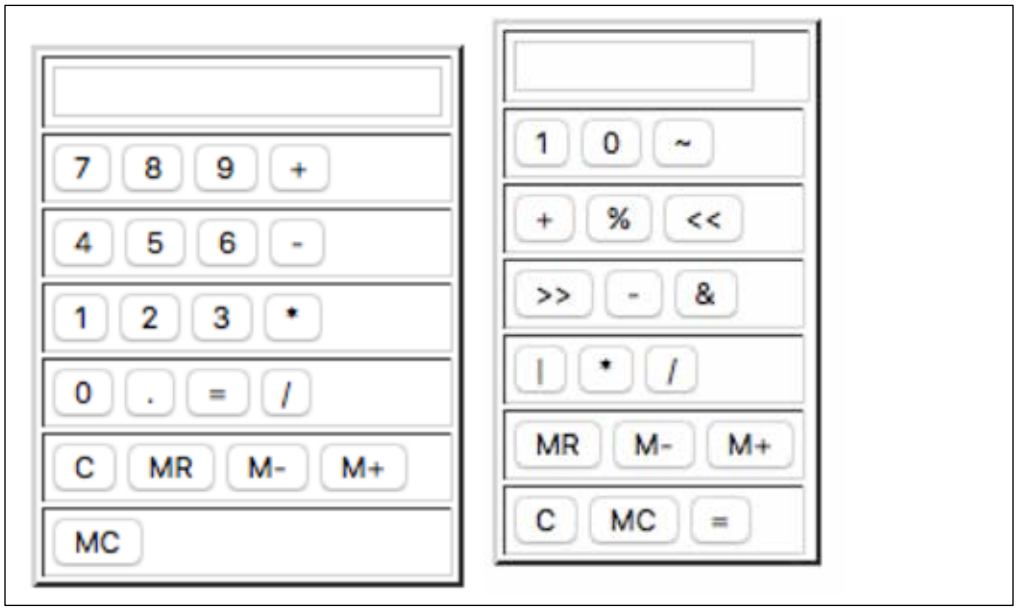
   LIST OF EXAMPLES AND THEIR PURPOSES

   01: shows js is loosely typed
   02: shows use of === and also how conditions are always evaluated to true or false.
   03: shows the two ways of declaring functions. Shows how variables store pointer to functions. Start using the browser developer tools to figure out why the FOURTH print is missing.
   04: shows function declaration hoisting. Also, shows correct and incorrect uses of function expressions.
   05: shows named function expressions.
   06: shows local and global variables in js and also order of execution of javascript
   07: shows functions are first class objects in javascript
   08: shows closures
   09: shows an example of use of closure to create hidden variables
   10: shows an example on what to watch out for in using closures
   11: shows example of code reuse using functions as first class objects
   12: shows how to create an object. Also, shows the use of the THIS object in javascript!
   13: shows object attributes can be accessed like arrays.

   IT IS REQUIRED THAT YOU TRY EACH EXAMPLE AND DO THE "TODO" SECTIONS.

2. Simple Calculator

A complete example of another program (Matching game) is provided in folder 04_SampleProgram. A starting template is provided in folder 03_ExerciseHelp. Your assignment is to use this template to create two simple calculator programs (one decimal and other binary) using **Objects, functions, and closures**. These calculators should look ***approximately*** like the below pictures.



Clarifications:

1. For binary calculator,

    a. You can assume that only positive binary numbers are represented. For example, 9 is represented as 1001. Negative 9 will be represented by '–' operator followed by 1001.
    b. ~ not (invert the binary value)
    c. % mod or remainder (divide the first value by the second, what is remaining)
    d. << one bit-shift left (insert zeroes in the vacated positions on the left)
    e. >> one bit-Shift right (insert zeroes in the vacated positions on the right)
    f. – minus
    g. & and
    h. | or (or two binary values together)
    i. * multiply
    j. / division
    k. Note that for some operations on binary calculator, it may be more convenient to convert the binary numbers to integers and then do the operation.

2. Look at the normal calculator to figure out the functionality of M+, M-, MR and MC.

3. Assume that the calculator does not need to calculate complex operations such as 5 + 5 * 5. Instead, expect users to press '=' operator after a basic operation. So, press 5 + 5 followed by =. At this point it should show 10. Then, press * and then 5 followed by =. At this point show 50. In other words, assume that we are only expecting "operand1 operator operand2" operation. However, we can use the results of the previous operation as the first operand for the next operation.

CHECKLIST

[ ] Your javascript files should be named “calculator.js" and "calculatorBinary.js".
[ ] Name your Objects based on their purpose. Do the same with your JavaScript functions.
[ ] MR (shows memory value on screen)
[ ] MC (clears memory value)
[ ] C (clears screen value)
[ ] = (shows results of an operation) (repeatedly pressing '=' repeats the last operation – this part is extra-credit worthy)
[ ] M+ (Whatever is on screen gets added to memory)
[ ] M- (Whatever is on screen gets subtracted from memory)
Make sure that your variables are not global (so that if someone includes some other js files with same names for variables, then your code still works ok). You do this by using closures.
After a few days, if you are unable to make progress - please contact TAs!

Submission

Make sure your solutions work on Chrome (which is what TAs will use to grade the assignment).

SUBMIT ONLY SIX FILES (DO NOT SUBMIT EXTRANEOUS FILES):

1) calculator.html and calculator.js
2) calculatorBinary.html and calculatorBinary.js
3) a README.txt file describing any specifics of how to run your program. It also should indicate any "extra-credit" worthy parts that were attempted.
4) a participation.txt (a simple txt file, which clarifies the specific participation of group members)
   s

Zip the above six files and submit this zip file on blackboard. Remember there is only one submission per group. Make sure to include all the files that are needed in order to run your program.