Web Lab
Project 0:
A Few JavaScript Exercises
Due: Thursday, September 12, 2013 by 10:30 am

Goal
The goal of this project is to give you practice with JavaScript by completing a few small exercises.

Files Provided
The following files are in the same .zip folder as this project description was in:

```
p0.js             javascript file that you will need to edit
p0.html           html file that loads the p0 javascript file for testing
```

Project Overview
For this project, modify p0.js by adding the functions below. Each function should behave as specified. You should also add a test function called runTests() which tests the functions that you wrote. The functions you need to define are below. You can 'run' your javascript file using p0.html. See the very last few lines of p0.js to see how you can easily call runTests once the p0.html page has loaded.
IMPORTANT: Make sure your function names are correct!

addHeader(content, level) (10 points)
Summary: Adds a <hN> tag (where N is specified) under the <body> tag of the html document (aka the 'document body'). HINT: you should use document.body.appendChild().

Example:
If before your html file is:

```
<html>
…
        <body></body>
</html>
```

e1("foo", 4) should add an h4 element with the text "foo", and your html should now be:

```
<html>
…
        <body><h4>foo</h4></body>
</html>
```

Parameters:
content: the text content of the header
level: the level of the header (can be from 1 to 5). If the value is not a valid number or the passed in parameter is too high, defaults to h3.

fibonacci(n) (10 points)

Summary: Generates a string containing the first N numbers of the Fibonacci sequence. See http://en.wikipedia.org/wiki/Fibonacci_sequence for more on the sequence. Each number in the string should be separated by commas. If the parameters passed are invalid (smaller than one or not a number), the function should return undefined.

Examples:

fibonacci(0)-> undefined
fibonacci (1)->"0"
fibonacci (2)->"0,1:
fibonacci (7)->"0,1,1,2,3,5,8"
fibonacci(-1)-> undefined
Fibonacci("invalid")-> undefined

Parameters:

n: the number of Fibonacci numbers to generate. If n is invalid, returns undefined.

Return: A comma separated string of the first n Fibonacci numbers. If n is invalid, returns undefined.

findNumbersInStr(strToParse) (10 points)

Summary: Takes in a string of numbers separated by spaces and returns an array containing only the numbers in the string AS NUMBERS (not strings). You may assume that only numbers are passed into the string.

Example:

findNumbersInStr("5 1 2 3 4 5 42")-> [5,1,2,3,4,5,42]

Parameters:

strToParse: The string to parse which contains one or more numbers.

Return: An array containing all numbers that are present in the string. If the input is not a string, returns undefined.

map(array, functionToApply) (10 points)

Summary: Applies a function to every element in an array and returns the resulting array.

Example:

```
var oneTo5 = [1,2,3,4,5];
map(oneTo5, function (x) { return x * 2 }); -> [2,4,6,8,10]
```

Parameters:

array: the input array.
functionToApply: the function to apply.

Return: The array that is a result of applying the function to every element in the input array. Returns undefined if parameters are invalid.

makeIterator(data) (5 points, extra credit)

Summary: Returns an object which contains an array, and which has an 'each' method that applies an input function to every element in its array. NOTE: this is not quite like the previous exercise. Your each function should *NOT* return an array that is the result of applying an input function to each element in the object's array. Instead, the each function should simply apply the input function to each element in the object's array. In other words, your returned object should be of the form:

{

array: [the array that is passed in],
each(functionToApply): [a function which applies the given function to the array your object contains]
}

Example:

```
var oneTo5 = [1,2,3,4,5];
var oneTo5_mod = makeIterator(oneTo5);
var twoTo10 = [];
oneTo5_mod.each(function (x) { twoTo10.push(2 * x) });
twoTo10 -> [2,4,6,8,10]
```

Parameters:

data: An Array of data that the object should store

Return: An object which stores the array, which contains a method called 'each' which takes a function as a parameter and applies it to each element in the data array (but does not modify the array).

runTests(10 points)

Summary: This function should test each of the above functions. In other words, you should verify that the above functions behave as specified by using output statements. You should also check for edge cases. For each above function, write at least three tests per function. You can either output results to the chrome debugger using `console.log(message)`, or output to the actual webpage (hint: you can use addHeader to output your results).

*** Make sure to **comment all of your code** and to **write good code** because you will be graded both for your comments and your coding style.

Grading

Your program will be run on my machine against test cases unknown to you. If your functions performs properly, are well structured, and well documented you will receive 10 points for each exercise.

You will be graded on the quality of your code and commenting. This will include factors such as modularity, sensible method and variable names, and overall clarity. You may find this reference useful: http://particletree.com/features/successful-strategies-for-commenting-code

Turning Your Program In

The program is due Thursday, September 5, 2013 at 10:30 am. You should turn in your assignment via Blackboard, attaching a zip file with the contents below. Please write a README.txt file which mentions any online sources you used to help with the project, as well as any notes about your programs (i.e. if you couldn't get a particular part of the project to work). Create a zip file that contains p0.js, p0.html, and the README.txt file. Then, name the file LASTNAME_FIRSTNAME_P#.zip. For example:

```
chang_kerry_p0.zip
```

You will lose 3 points if your submission is not in the correct format.

Late Policy

Each day (24 hrs) late 5% will be deducted from your assignment grade.