SENG 4640 - Lab 3 - Introduction to JavaScript

In this lab, you will implement some simple JavaScript functions.

In completing this lab, you will:

- Gain familiarity with JavaScript syntax and writing JavaScript functions
- Get experience working with JavaScript arrays and objects
- See how to execute JavaScript code using a web browser's console

Getting Started

Start by downloading the original HTML and JS files, "petstore.html" and "petstore.js" from Moodle and then saving them on your computer.

The petstore.js file contains the starter code for the functions that you will implement in this lab; their full descriptions are below. You should be able to open this file with a plain-text editor and see the code that we provide. You will also see that petstore.js includes prototype functions for creating the objects you will need in implementing and testing the functions that you will write. Please do not change these functions.

At the bottom of petstore.js is a simple "hello world" function that you can use to make sure your development environment is set up correctly.

For this lab, we recommend that you edit petstore.js using a plain-text editor and then use your web browser's JavaScript console to execute your code. If petstore.js is in the same folder/directory as petstore.html, and you open petstore.html, you should be able to access the functions in petstore.js from the browser's console.

We recommend you use Google Chrome for this lab. Open petstore.html in your browser, then open the console by CTRL + SHIFT + J.

Once you've opened the console, type "helloworld()" in the console (without the quotes) to run the helloworld() function in petstore.js. Your browser should print "hello world!" in the console. If so, you're ready to proceed!

Activity

The functions that you will implement in this lab are all related to the operation of a pet store (as you'll see later in your course project), where shoppers can purchase animals as pets.

In petstore.js, implement these functions as follows:

- **calculateFoodOrder**: This function should calculate the total amount of pet food that the store should order for the upcoming week. The *numAnimals* parameter represents the number of animals in the store, and *avgFood* represents the average amount of food (in kilograms) eaten by each animal each week. The function should return the total amount of pet food that should be ordered for the upcoming week, or -1 if *numAnimals* or *avgFood* is less than 0 or non-numeric.

- **mostPopularDays**: This function determines which day of the week had the most number of people visiting the pet store. If two or more days are tied for the highest amount of traffic, an array containing the days (in any order) should be returned. If the input is null or an empty array, the function should return null. The input is an array of Weekday objects, which are created using the prototype function defined toward the bottom of petstore.js. This function should return a string containing the name of the most popular day of the week if there is only one most popular day, and an array containing the names (as strings) of the most popular days if there are more than one that are most popular.

- **createAnimalObjects**: Given three arrays of equal length containing information about a list of animals – where *names[i]*, *types[i]*, and *breeds[i]* all relate to the same, single animal – this function should return an array of Animal objects constructed from the information provided in the arrays. The parameter *names* represents the array of the animals' names; *types* represents the array of the animals' types (e.g. "Dog", "Cat", "Bird"); and *breeds* represents the array of the animals' breeds. This function should return an array of Animal objects (which you can create using the prototype function at the bottom of petstore.js), each of which contains the animal's information, or an empty array if the arrays' lengths are unequal or zero, or if any array is null.

Your implementations are only expected to handle the "normal" operations of these functions, and any extra conditions listed above. You may think of or encounter other situations or inputs not described here, but you only need to consider the ones listed above for grading.
One more important note. Please do not change the names or lists of parameters for any of the functions, as these will be used during grading. Also, please do not change the function prototypes for the Weekday, Item, or Animal classes, as those are used during grading as well. Be sure that all JavaScript is in petstore.js and that you have not created any additional files.

Notes

- Be sure that you're able to run the "helloworld()" function in your browser's JavaScript console before trying to implement the other functions, so that you know you have a development environment in which to work.

- For *calculateFoodOrder*, use the JavaScript *Number* function for converting a variable to its numeric form, and consider the *isNaN* function for determining whether the conversion was successful.

- For *mostPopularDays*, remember that you can use the JavaScript console as a "REPL" to create variables – including objects – and then use them in invoking your functions. For instance, you can create a Weekday object using the JavaScript "new" keyword with the prototype function (review the lesson on functions if you don't recall how to do that), and then create an empty array and then use the array's "push" function to add objects to the array before calling *mostPopularDays*. You can then use a similar approach in implementing *createAnimalObjects* when it comes to creating new values and putting them into an array.

- The instructor is available during office hours. Meetings (physical or virtual) at other times can also be arranged, by appointment.

Marking Schema

Criterion	Weight	Description
JavaScript Lab		
calculateFoodOrder Function	30%	• Correctly calculates the total food order based on 'numAnimals' and 'avgFood'. • Handles invalid input ('numAnimals' or 'avgFood' less than 0 or non-numeric) by returning -1.
mostPopularDays Function	40%	• Correctly identifies the day(s) with the most visits based on the array of 'Weekday' objects. • Handles ties by returning an array of day names. • Returns 'null' for 'null' or empty array input.
createAnimalObjects Function	30%	• Correctly creates an array of 'Animal' objects from the provided 'names', 'types', and 'breeds' arrays. • Handles unequal array lengths, zero-length arrays, or 'null' arrays by returning an empty array.
Total	**100%**	

Good Luck!