Assignment #2

Due: 6/14 at 11:59 PM

Can work in groups or alone. Max group size is 4.

Goal:

To improve your understanding of JavaScript syntax, functions, and higher-order functions, and to better understand JavaScript and Array methods.

Point Breakdown:

15 pts - Assignment functionality

5 pts - Code documentation (proper commenting), code organization (no sloppy code, well-formatted and easy to read), following git feature branch workflow, creating pull requests when merging feature branches, small and frequent commits with appropriate commit messages, etc.

~~~~~~

Assignment:

The following are some of the most popular and used methods in functional languages:

- forEach
- Map
- Filter
- Some
- Every
- Reduce
- Includes
- indexOf
- Push
- lastIndexOf
- Object.keys()
- Object.values()
~~~~~~

For this assignment, you will recreate these methods using JavaScript functions. Make sure to carefully understand what each method is designed to do, and DO NOT use any of the respective native JS methods to implement your solutions. Also, keep in mind that these methods do not mutate the input array. **Strongly suggested: use Mozilla Developer Network (MDN) Web Docs to understand how each method works and what arguments they take.**

forEach()

Without using the native “Array.prototype.forEach” method of JavaScript, compose a function titled “myEach” that will take in an array of elements and execute any callback function on each of those elements.

map()

Without using the native “Array.prototype.map” method of JavaScript, compose a function titled “myMap” that will take in an array of elements and execute a callback function on each of those elements.

filter()

Without using the native “Array.prototype.filter” method of JavaScript, compose a function titled “myFilter” that will take in an array of elements and execute a callback function on each of those elements.

some() (aka any())

Without using the native “Array.prototype.some” method of JavaScript, compose a function titled “mySome” that will take in an array of elements and execute a callback function on each of those elements.

every()

Without using the native “Array.prototype.every” method of JavaScript, compose a function titled “myEvery” that will take in an array of elements and execute a callback function on each of those elements.

reduce()

Without using the native “Array.prototype.reduce” method of JavaScript, compose a function titled “myReduce” that will take in an array of elements and execute a callback function on each of those elements.

includes()

Without using the native “Array.prototype.includes” method of JavaScript, compose a function titled “myIncludes” that will take in an array of elements and indicate whether or not a target element is contained within the input array. This returns a boolean.

indexOf()

Without using the native “Array.prototype.indexOf” method of JavaScript, compose a function titled “myIndexOf” that will take in an array of elements and returns the index of the first encounter of a target element (if it is found) or -1 if that element does not exist within the input array.

push()

Without using the native “Array.prototype.push” method of JavaScript, compose a function titled “myPush” that will take in an array of elements as well as an elementToAdd and append that element to the end of the array.

lastIndexOf()

Without using the native “Array.prototype.lastIndexOf” method of JavaScript, compose a function titled “myLastIndexOf” that will take in an array of elements and returns the index of the last encounter of a target element (if it is found) or -1 if that element does not exist within the input array.

Object.keys()

Without using the native “Object.keys()” method of JavaScript, compose a function titled “grabKeys” that will take in an object and return all of the keys of the key:value pairs of that object.

Object.values()

Without using the native “Object.values()” method of JavaScript, compose a function titled “grabValues” that will take in an object and return all of the values of the key:value pairs of that object.