Assignment 2 - JavaScript Methods

DUE DATE

- **Due on Friday, 3/3, at 11:59 PM**
- **You can work in groups or individually. Maximum 4 students per group. Working in groups is strongly recommended and encouraged.**

HOW TO SUBMIT

- **Blackboard:** Submit (1) the link to your GitHub repository and (2) your group member names on Blackboard by 11:59 PM on the due day. Each student must submit the assignment individually on Blackboard—even when working in a group.
- **GitHub:** In the README section of GitHub repository, please list your group member names and GitHub usernames.

GRADING

This assignment is worth **10%** of your grade.

- **7%** - Assignment functionality
- **3%** - Git version control, continuing the use of feature branch workflow. Create a feature branch for each function being implemented.

GOAL

The goal is to improve students' understanding of JavaScript syntax, functions, and higher-order functions, in addition to a better understanding of JavaScript and Array methods.

ASSIGNMENT

Consider the following *native* JavaScript methods:

1. **map()** [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map]
2. **filter()** [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/Filter]
3. **some()** [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/Some]
4. **every()** [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/Every]
5. **reduce()** [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/Reduce]
6. **includes()** [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/includes]
7. **indexOf()** [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/indexOf]
8. **lastIndexOf()** [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/lastIndexOf]
9. **Object.keys()** [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/keys]
10. **Object.values()** [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/values]

In this assignment, you will re-create these methods using JavaScript functions (see list below). For example, you will create your own "myMap" method corresponding to the *native* "map" method.

Strongly suggested: Carefully understand what each *native* method is designed to do. Use *Mozilla Developer Network (MDN) Web Docs* links above to understand how each *native* JavaScript method works. Pay attention to what arguments they take as well as the return value.

Requirements and Functionalities

1. DO NOT use any of the *native* JavaScript methods to implement your solutions.
2. If applicable, you can re-use your own methods that you have created.
3. For each method, make a feature branch on GitHub.

Implement your solutions for the following JavaScript methods:

1. map()

Without using the *native* "Array.prototype.map" method of JavaScript, compose a function titled "**myMap**" that shall take in an array of elements and execute a callback function on each of those elements. **Syntax: myMap(callbackFn)**

2. filter()

Without using the *native* “Array.prototype.filter” method of JavaScript, compose a function titled "**myFilter**" that shall take in an array of elements and execute a callback function on each of those elements. **Syntax: myFilter(callbackFn)**

3. some()

Without using the *native* "Array.prototype.some" method of JavaScript, compose a function titled "**mySome**" that shall take in an array of elements and execute a callback function on each of those elements. **Syntax: mySome(callbackFn)**

4. every()

Without using the native “Array.prototype.every” method of JavaScript, compose a function titled "**myEvery**" that shall take in an array of elements and execute a callback function on each of those elements. **Syntax: myEvery(callbackFn)**

5. reduce()

Without using the *native* "Array.prototype.reduce" method of JavaScript, compose a function titled "**myReduce**" that shall take in an array of elements and execute a callback function on each of those elements. **Syntax: myReduce(callbackFn)**

6. includes()

Without using the *native* "Array.prototype.includes" method of JavaScript, compose a function titled "**myIncludes**" that shall take in an array of elements and indicate whether or not a target element is contained within the input array. This returns a boolean. **Syntax: myIncludes(searchElement)**

7. indexOf()

Without using the *native* "Array.prototype.indexOf" method of JavaScript, compose a function titled "**myIndexOf**" that shall take in an array of elements and return 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. **Syntax: myIndexOf(searchElement)**

8. lastIndexOf()

Without using the *native* "Array.prototype.lastIndexOf" method of JavaScript, compose a function titled "**myLastIndexOf**" that shall take in an array of elements and return 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. **Syntax: myLastIndexOf(searchElement)**

9. Object.keys()

Without using the *native* "Object.keys" method of JavaScript, compose a function titled " **Object.myKeys**" that shall take in an object and return all of the keys of the key:value pairs of that object. **Syntax: Object.myKeys(object)**

10. Object.values()

Without using the *native* "Object.values" method of JavaScript, compose a function titled " **Object.myValues**" that shall take in an object and return all of the values of the key:value pairs of that object. **Syntax: Object.myValues(object)**