Arrays

Intstructions:

Create a new github repository named “arrays-exercises”, clone it into your WSL, open it in VS Code and create a new file called “index.js” and start coding! Good luck!

Ex 1:

Write a function named *wordWithinArray(array,word)* using *Array.indexOf*. The function takes in both a word and an array of words as arguments and returns a *boolean* that returns true if that string is located inside of the array, or false if it does not.

Tests:

- *console.log(wordWithinArray(["apple", "banana", "caramel", "chocolate"], "apple")); //=> true*
- *console.log(wordWithinArray(["dog", "cat", "camel", "bird"], "camel")); //=> true*
- *console.log(wordWithinArray(["apple", "banana", "caramel", "chocolate"], "pineapple")); //=> false*
- *console.log(wordWithinArray(["dog", "cat", "camel", "bird"], "panther")); //=> false*

Ex 2:

Write a function *combineArray(array1, array2)* that takes in two arrays of numbers and *returns* the two arrays combined into a single array.

Hint: Use the *Array.concat* method but be aware that calling this method won't permanently change, also known as mutate, either array.

Ex 3:

Write a function *threeIncreasing(numbers)* that accepts an array of numbers as an argument. The function should return a *boolean* indicating whether or not the array contains three consecutive numbers in consecutive increasing order, like 7, 8, 9.

Tests:

- *console.log(threeIncreasing([3, 2, 11, 12, 13, 2, 4])); // true*
- *console.log(threeIncreasing([2, 7, 8, 9])); // true*
- *console.log(threeIncreasing([7, 2, 4, 5, 2, 1, 6])); // false*
- *console.log(threeIncreasing([1, 2, 4, 5, 2, 7, 8])); // false*

Ex 4:

Write a function *myIncludes(arr, target)* that accepts an array and a target value as args. The function should return a boolean indicating whether the target is found in the array. Solve this **without** *Array.includes* or *Array.indexOf.*

Tests:

- *console.log(myIncludes(['a', 'b', 'c', 'e'], 'c')); // true*
- *console.log(myIncludes(['a', 'b', 'c', 'e'], 'a')); // true*
- *console.log(myIncludes(['a', 'b', 'c', 'e'], 'z')); // false*
- *console.log(myIncludes([43, -7, 11, 13], 11)); // true*
- *console.log(myIncludes([43, -7, 11, 13], 1)); // false*

Ex 5:

Write a function *myIndexOf(arr, target)* that takes in an array and target value as args. The function should return the first index where the target is found in the array. If the target is not found, it should return -1. Solve this **without** using *Array.indexOf*.

Tests:

- *console.log(myIndexOf(['a', 'b', 'c', 'e'], 'c')); // 2*
- *console.log(myIndexOf(['a', 'b', 'c', 'e'], 'e')); // 3*
- *console.log(myIndexOf(['a', 'b', 'c', 'e'], 'z')); // -1*
- *console.log(myIndexOf([43, -7, 11, 13, 43], 43)); // 0*
- *console.log(myIndexOf([43, -7, 11, 13], 1)); // -1*

Ex 6:

Write a function *sumArray(array)* that takes in an array of numbers and returns the total sum of all the numbers.

Tests:

- *console.log(sumArray([5, 6, 4])); // => 15*
- *console.log(sumArray([7, 3, 9, 11])); // => 30*

Ex 7:

Write a function *productWithReduce(nums)* that takes in an array of numbers. The function should return the total product of multiplying all numbers of the array together. You can assume that *nums* will not be an empty array.

Tests:

- *console.log(productWithReduce([10, 3, 5, 2])); // 300*
- *console.log(productWithReduce([4, 3])); // 12*