JavaScript Functions Exercise

In this exercise, you'll create several JavaScript functions, and for some, you'll also create the JSDoc documentation. The names, parameters, and return values of the functions are provided below.

You'll create these functions in `exercises.js`. You can check your work by viewing `test.html` using Live Server in VS Code.

Step One: View initial unit test results

Open the folder where this README file is located in VS Code so you can use Live Server to view the unit test results.

Right-click on `test.html` and select "Open with Live Server." In the browser window that opens, you'll see the failing tests for the functions you need to complete.

Step Two: Complete functions 1-4

isAdmitted

Write a function called `isAdmitted`. It checks entrance scores and returns `true` if the student is admitted and `false` if rejected.

It takes three parameters:

- gpa
- satScore (optional)
- recommendation (optional)

Return `true` if any of the conditions below are true:

- gpa is above 4.0 OR
- SAT score is above 1300 OR
- gpa is above 3.0 and they have a recommendation OR
- SAT score is above 1200 and they have a recommendation

Otherwise, reject the student and return `false.`

If implemented correctly, all tests under "isAdmitted" pass.

useParameterToFilterArray

Write a function called `useParameterToFilterArray` that takes an anonymous function and uses it in the `unfilteredArray` filter function. Return the result.

If implemented correctly, all tests under "useParameterToFilterArray" pass.

makeNumber

Write a function called `makeNumber` that takes two strings of digits, concatenates them together, and returns the value as a number.

Example: if two strings are passed in "42293" and "443", this function returns the number 42293443.

If implemented correctly, all tests under "makeNumber" pass.

addAll

Write a function called `addAll` that takes an unknown number of parameters and adds all of them together. Return the sum.

If implemented correctly, all tests under "addAll" pass.

Step Three: Complete remaining functions and add JSDoc

makeHappy

Write and document a function called `makeHappy` that takes an array and prepends "Happy " to the beginning of all the words and returns them as a new array.

> Hint: Use the `map` function.

If implemented correctly, all tests under "makeHappy" pass.

getFullAddressesOfProperties

Write and document a function called `getFullAddressesOfProperties` that takes an array of JavaScript objects containing the following keys:

- `streetNumber`
- `streetName`
- `streetType`
- `city`
- `state`
- `zip`

The function returns an array of strings that turns the JavaScript objects into a mailing address in the form of:

```
streetNumber streetName streetType city state zip
```

> Hint: Use `map` and an anonymous function.

If implemented correctly, all tests under "getFullAddressesOfProperties" pass.

findLargest

Write and document a function called `findLargest`. Using `forEach`, find the largest element in an array. It must work for strings and numbers.

If implemented correctly, all tests under "findLargest" pass.

Challenge: getSumOfSubArrayValues

Write and document a function called `getSumOfSubArrayValues`. The function takes an array of arrays, adds up all sub values, and returns the sum.

For example, if you got this array as a parameter:

```
[
    [1, 2, 3],
    [2, 4, 6],
    [5, 10, 15]
];
```

`getSumOfSubArrayValues` returns 48. To do this, you'll use two nested `reduce` calls with two anonymous functions.

Read the tests to verify you have the correct behavior.

If implemented correctly, all tests under "getSumOfSubArrayValues" pass.

Once you complete all tasks, all tests pass.