Project 2: JAVASCRIPT

Problem 1

Make sure that make-multi-filter.js is the name of the new file that you create in the project2 dire ctory.

This file is going to have the code for the Multi Filter Function in it.

MakeMultiFilter should be a global function that can receive an array as an input and return a fu nction that can be used to filter the array's contents. This function should be declared and given t he name MakeMultiFilter. The function that was created as a result (arrayFilterer) keeps track of an internal notion that is called currentArray. currentArray begins with the exact same configurat ion as originalArray when it is first created. arrayFilterer's input parameters may be either two fu nctions or an array. These are the following:

A kind of function that takes an element of an array as its input and returns a boolean value. This function is called for each individual member of currentArray, and the contents of currentArray a re modified to reflect the outcomes of the filterCriteria operation. In the event that filterCriteria g ives a false result for an element, that element has to be removed from the currentArray. In such case, it will stay in the currentArray. In the event that filterCriteria is not a function, the function that is returning the result (arrayFilterer) is obligated to return the value of currentArray without doing any filtering.

callback refers to a function that is activated after the filtering process is finished. callback is will ing to take the value of currentArray in its place of an argument. This should make a reference to the value of the originalArray inside of the callback function. If the callback parameter is not a fu nction, you should disregard it. callback does not have a return value.

If the filterCriteria option is not given, the arrayFilterer function should return itself; otherwise, it should return the currentArray. If filterCriteria is specified, the method should return itself. It is e ssential that the execution of many arrayFilterer methods in parallel be possible.

The code that follows provides an example of how the functions that you develop for this proble m might be used.

```
// Invoking MakeMultiFilter() with originalArray = [1, 2, 3] returns a
// function, saved in the variable arrayFilterer1, that can be used to
// repeatedly filter the input array
var arrayFilterer1 = MakeMultiFilter([1, 2, 3]);
```

```
// Call arrayFilterer1 (with a callback function) to filter out all the numbers
// not equal to 2.
arrayFilterer1(function (elem) {
  return elem !== 2; // check if element is not equal to 2
}, function (currentArray) {
  // 'this' within the callback function should refer to originalArray which is [1, 2, 3]
  console.log(this); // prints [1, 2, 3]
  console.log(currentArray); // prints [1, 3]
});

// Call arrayFilterer1 (without a callback function) to filter out all the
// elements not equal to 3.
arrayFilterer1(function (elem) {
  return elem !== 3; // check if element is not equal to 3
});

// Calling arrayFilterer1 with no filterCriteria should return the currentArray.
var currentArray = arrayFilterer1();
console.log('currentArray', currentArray); // prints [1] since we filtered out 2 and 3

// Since arrayFilterer returns itself, calls can be chained
function filterTwos(elem) { return elem !== 2; }
function filterThrees(elem) { return elem !== 3; }
var arrayFilterer2 = MakeMultiFilter([1, 2, 3]);
var currentArray2 = arrayFilterer2(filterTwos)(filterThrees)();
console.log('currentArray2', currentArray2); // prints [1] since we filtered out 2 and 3

// Multiple active filters at the same time
var arrayFilterer3 = MakeMultiFilter([1, 2, 3]);
var arrayFilterer4 = MakeMultiFilter([4, 5, 6]);
console.log(arrayFilterer3(filterTwos)()); // prints [1, 3]
console.log(arrayFilterer4(filterThrees)()); // prints [4, 5, 6]
```

Problem 2: Template Processor

-template-processor.js should be the name of the new file that you create and save in the project2 directory. The source code for the Template Processor is going to be stored in this file.

Build a class called TemplateProcessor that will hold the template string parameter as well as the fillIn method. When the fillIn method is used with a dictionary object as an argument, it generates a string that contains the template with the appropriate values taken from the dictionary object. The usual JavaScript function Object(), sometimes known as "native code," and the prototype structure are both required for the development of the TemplateProcessor.

The fillIn method will return the template string, but any text of the form property will have the relevant dictionary value substituted with it.

If the template calls for a property that does not exist in the dictionary object, that property should be changed to an empty string if it is specified in the template. If the property is situated between two words, replacing it with an empty string will result in two whitespaces appearing in a row if it is in that location. For example, "This undefined property is wonderful" and "This" both mean the same thing. This may be considered appropriate. It is not necessary for you to worry about getting rid of the extra whitespace.

Your system just needs to handle property values that are presented in the appropriate format. In the following scenarios, the behaviour of it could be left unexplained since we won't be specifically checking for it to behave in those ways.

nested properties - {{foo {{bar}}}} or {{{{bar}}}} or {{{bar}}}
Bar denotes unequal brackets.
any property string that has stray brackets, day or day
The code that follows provides an example of how the functions that you develop for this problem might be used.

```
var template = 'My favorite month is {{month}} but not the day {{day}} or the year {{year}}';
var dateTemplate = new TemplateProcessor(template);

var dictionary = {month: 'July', day: '1', year: '2016'};
var str = dateTemplate.fillIn(dictionary);

assert(str === 'My favorite month is July but not the day 1 or the year 2016');

//Case: property doesn't exist in dictionary
var dictionary2 = {day: '1', year: '2016'};
```

```
var str = dateTemplate.fillIn(dictionary2);


assert(str === 'My favorite month is  but not the day 1 or the year 2016');
```