JavaScript Quiz

1. The Array class in modern browsers has a built in function “map” that can be called to map an array of values into another array. The way it works is you call map on the original array and pass in a function to translate each value in the array to a new value. Map returns the new array containing all the translated values.

For example, calling map on the array [2,4,6], and passing it a function that doubles what it is given as the mapping function, will return the NEW array [4,8,12]. (Note - this is just an example, challenge is NOT to double numbers in an array, challenge is to create a mapping function that can take any transformation callback function. Could be a function to triple a number, or uppercase a string or whatever. Our example is to use a callback that will double a number.)

Write your OWN implementation of the map method and test it as described above. Pass it an array like [2,4,6] and it should return a new array like [4,8,12], After calling it, write out to the console both the original array still showing the original values and the new array showing the new values.

(This is very similar to the many examples we did earlier, forEach, every, some, filter, etc… Obviously each is a little different or there would only be one though…)

* 1. Create a counter module/IIFE. The module/IIFE should keep a count. Users of the module must be able to increment the count and get the current count. Reminder – keep global / top level scope data to a minimum, and keep private data, private. The module should be contained within an outer module (or namespace) “app”.
  2. Create a second module/IIFE in another JavaScript file. This module/IIFE is exactly the same as the first except that this module instead of BEING a counter, creates counters for you, and returns them to you for use. Additionally, it should keep track of how many counters have actually been created. This module should also be contained within an outer module (or namespace) “app”.

I.e. In the first case (a) there is only one and only ever will be one counter. You just use that one counter. In the second case (b) you ask for as many counters as you like. Each counter of course keeps its own count. (Recall how we created multiple people without having to repeat the person creation code multiple times).

Again reminder – keep global / top level data to a minimum, and keep private data, private, and of course this second module should be in the same parent outer module as the first, “app”

Note that a and b are very similar, if you can do one, you can do the other. You may just need to think about it a little. If you find this confusing. it might be easier to first make each work without worrying about putting it in a module/IIFE and then once you have the basic idea working, wrap it up in module/IIFE.

* 1. In yet a third JavaScript file write code to increment the counter from the first module 10 times.

Write code to create two counters from the second module and increment one 5 times, and one 15 times.

Write code at the END of this third file (not spread out throughout) that gets and prints out the final current counts of all the counters to the console proving that they are indeed at 10, 5, and 15 as expected.

* 1. Load all three JavaScript files into one HTML page and ensure the console shows what you expect.

Please write quality code. Make sure your code is formatted nicely (as I’m sure you always do). Make sure that there are no lint errors (where your linter is set to reasonable settings), no errors printed to console, etc.…