MoreJavaScript

Due Oct 7 at 11:59pm **Points** 10 **Questions** 3
Available Oct 7 at 12am - Oct 7 at 11:59pm 1 day **Time Limit** 45 Minutes

This quiz was locked Oct 7 at 11:59pm.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	15 minutes	10 out of 10

Score for this quiz: **10** out of 10
Submitted Oct 7 at 2:53pm
This attempt took 15 minutes.

Question 1

5 / 5 pts

Please write a function that takes two arrays as parameters and returns a new array that contains **all** the elements from the **first** array and **only** elements from the second array that are **less than 10**. You can use this snippet of code as a starting point if you want:

```
function mergeArrays(firstArray, secondArray){
  let resultArray = [];
  // Your code goes here
  return resultArray;
}
```

Here are a couple examples of inputs and outputs:

```
mergeArrays([1,2,3], [4,5,6]); // returns [1,2,3,4,5,6]
mergeArrays([14, 13, 12], [11, 10, 9]); // returns [14, 13, 12, 9] because 1
1 and 10 are not less than 10
```

Your Answer:

```
function mergeArrays(firstArray, secondArray){
   let resultArray = [];
   // Your code goes here
   for(item of firstArray){
      resultArray.push(item);
   }
   for(item of secondArray){
      if(item<10){
         resultArray.push(item);
      }
   }
   return resultArray;
 }
```

Here's one approach that works

```
function mergeArrays(firstArray, secondArray){
  let resultArray = [];
  for(const element of firstArray) {
    resultArray.push(element);
  }
  for(const element of secondArray) {
    if(element < 10) {
      resultArray.push(element);
    }
  }
  return resultArray;
}
```

Question 2

3 / 3 pts

This code is supposed to make 10, "mostly copies" of an object, with all the same properties except for a "copyId" property that is different for each copy. Does it work? If not, why not?

Hint: Try running the code in the developer tools. You may need to click on the result array in the developer tools to expand it.

```
function generateTenCopies() {
  let initialObject = { name: "Ryan", age: 28, isProfessor: true };
  let arrayOfCopies = [];
  for(let i = 1; i <= 10; i++) {
    let currentCopy = initialObject;
    currentCopy.copyId = i;
    arrayOfCopies.push(currentCopy)
  }
  return arrayOfCopies;
}
```

Your Answer:

It does not work because for each iteration of the loop, the copy is pointing to the same object(initialObject). Therefore, all copies will have an ID of 10 since it is the last iteration of the loop.

This code does not work. Remember that variables are just labels for data. This is especially important for objects and arrays. In the code above, when I set currentCopy = initialObject, it does not actually copy initialObject. It just gives it another label. When I set the copyId to i, it updates the copyId in initialObject. In fact, every element in the arrayOfCopies is just another window into the exact same piece of data (rather than 10 separate "mostly copies")

Question 3

2 / 2 pts

What are the two main types of for loops that we covered in class? Why would you use each one? How is a while loop different and when would you use it instead?

Your Answer:

The main two types of for loops are

1. for(const num of numbers) This is good when you don't know how many times you need to loop.

2. for(let i=0; i<10;i++) This is good for running a loop a certain amount of times.

A while loop is different because it runs until a certain condition is met. You should use it when you don't know how many times you need to do something, but want to stop when some condition is met.

One type of for loop that we covered in class looks like:

```
for(const element of array) {
  // Code
}
```

It is good for looping over each element of an array or the characters in a string

Another type of array looks like this:

```
for(let i = 0; i < 10; i++) {
  // Code
}
```

It's good for when you want to loop a certain number of times but don't necessarily have an array or string that is driving that decision. It also lets you do fancier things like setting up a loop for every even number between 20 and 50.

A while loop loops as long as a condition is true. It is good for running a repeated operation where you don't know exactly how many times it needs to run but you have a way of telling when it's done.

Quiz Score: **10** out of 10