Loops

7 min

In order to sort an

Preview: Docs Loading link description

array

, we'll need to visit pairs of elements and check if they should be moved or kept at their current

Preview: Docs Loading link description

index

. To accomplish this we'll use two loops:

- One loop that will execute an inner loop depending on the state of a variable representing whether the input array might be sorted or not
- An inner loop to compare and swap pairs of elements in the array

Instructions

1. Checkpoint 1 Passed

1.

Begin by taking a look at the bubbleSort() function given to you in **bubbleSort.js**. Note that it takes one argument, an array to be sorted.

To start sorting, we will use a variable to store the condition of the input array as a Boolean value: true, meaning our input array might still be unsorted and need additional swaps of elements and we'll later change it to false, meaning the input array does not need anymore swapping to sort it.

Add a variable inside bubbleSort() called swapping and assign it the value true.

Hint

Remember that we'll be changing the value of this swapping from true to false later on. Use the keyword that creates a variable that lets you assign a new value to it without throwing an error.

2. Checkpoint 2 Passed

2.

Below the line where you declared swapping, create a while loop. This is the outer loop of our program that only runs if the input array might not be sorted and needs swapping, (the condition stored in swapping).

Use swapping which is currently set to true as the while condition.

This ensures that we'll start running the while loop and run it at least once, since we need to loop through the input array at least one time to determine if it's already sorted or needs swapping.

Hint

A while loop without a condition will throw an error. Make sure you set a condition for your loop.

3. Checkpoint 3 Passed

3.

If we find that we don't need to swap any of the elements, it means that the array is already sorted from smallest to largest and we can stop running our code and return the sorted array. To stop our while loop we only need to change the while condition to false.

Inside of the while loop we created, set swapping to false.

(We'll add code later that will restart the loop if we might have to keep swapping to "bubble up" elements to the end of the array.)

4. Checkpoint 4 Passed

4.

Create a for loop nested inside the while loop under the line where you reassigned the value of swapping.

The for loop should visit every element in the input array starting from the first element and stopping at the second-to-last element. Setting the condition for the loop this way allows us to stay within the bounds of our input array and only check elements that exist.

Since the index is going to change, make sure to make it a let variable.

Hint

When looping over an array, don't forget that arrays are "zero indexed", meaning indices are counted from 0 and not 1.

5. Checkpoint 5 Passed

5.

Lastly, bubbleSort() should return a sorted input array, (we'll do the actual sorting in a later exercise).

Add code to return the sorted input array if we've exited our while loop.

bubbleSort.js

```
const swap = require('./swap');


const bubbleSort = input => {
  let swapping = true;
  while (swapping) {
    swapping = false;
    for (let i = 0; i < input.length - 1; i++) {


    }
  }
  return input;
};


module.exports = bubbleSort;
```

swap.js

```
const swap = (arr, indexOne, indexTwo) => {


}


module.exports = swap;
```