```
1  // sample of arrays to sort
2  var arrayRandom = [9, 2, 5, 6, 4, 3, 7, 10, 1, 8];
3  var arrayOrdered = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
4  var arrayReversed = [10, 9, 8, 7, 6, 5, 4, 3, 2, 1];
5
6  // swap function helper
7  function swap(array, i, j) {
8    var temp = array[i];
9    array[i] = array[j];
10   array[j] = temp;
11 }
12
13 // be careful: this is a very basic implementation which is nice to understand the
   deep principle of bubble sort (going through all comparisons) but it can be greatly
   improved for performances
14 function bubbleSortBasic(array) {
15   var countOuter = 0;
16   var countInner = 0;
17   var countSwap = 0;
18
19   for(var i = 0; i < array.length; i++) {
20     countOuter++;
21     for(var j = 1; j < array.length; j++) {
22       countInner++;
23       if(array[j - 1] > array[j]) {
24         countSwap++;
25         swap(array, j - 1, j);
26       }
27     }
28   }
29
30   console.log('outer:', countOuter, 'inner:', countInner, 'swap:', countSwap);
31   return array;
32 }
33
34 bubbleSortBasic(arrayRandom.slice()); // => outer: 10 inner: 90 swap: 21
35 bubbleSortBasic(arrayOrdered.slice()); // => outer: 10 inner: 90 swap: 0
36 bubbleSortBasic(arrayReversed.slice()); // => outer: 10 inner: 90 swap: 45
37
38 // correct implementation: this is the usual implementation of the bubble sort
   algorithm. Some loops execution are avoided if not they are not needed
39 function bubbleSort(array) {
40   var countOuter = 0;
41   var countInner = 0;
42   var countSwap = 0;
43
44   var swapped;
45   do {
46     countOuter++;
47     swapped = false;
48     for(var i = 0; i < array.length; i++) {
49       countInner++;
50       if(array[i] && array[i + 1] && array[i] > array[i + 1]) {
51         countSwap++;
52         swap(array, i, i + 1);
53         swapped = true;
54       }
55     }
56   } while(swapped);
57
```

```
  console.log('outer:', countOuter, 'inner:', countInner, 'swap:', countSwap);
  return array;
}

bubbleSort(arrayRandom.slice()); // => outer: 9 inner: 90 swap: 21
bubbleSort(arrayOrdered.slice()); // => outer: 1 inner: 10 swap: 0
bubbleSort(arrayReversed.slice()); // => outer: 10 inner: 100 swap: 45

```