Insertion Sort

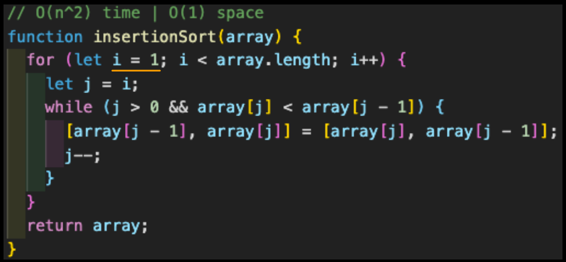
Input: array = [8, 5, 2, 9, 5, 6, 3]

Output: [2, 3, 5, 5, 6, 8, 9]

Input: An array of integers

Output: a sorted version of the input array

Use the insertion sort algorithm



```
// O(n^2) time | O(1) space
function insertionSort(array) {
  for (let i = 1; i < array.length; i++) {
    let j = i;
    while (j > 0 && array[j] < array[j - 1]) {
      [array[j - 1], array[j]] = [array[j], array[j - 1]];
      j--;
    }
  }
  return array;
}
```

Time : $O(n^2)$ (where n is the length of the array)
Since we loop through the array and at each element, we go backwards in the array to compare. $O(n)$ at best case (if array is sorted already)

Space : $O(1)$ since we are not storing any values and just swapping

The iteration starts at 1 because we assume the 0th index element is sorted. It will be the start of the sorted part of our array so we will start at index 1