Bubble Sort (Implementation)

(Reading time: 4 minutes)

We need to create a function that receives the array as an argument.

```
function bubbleSort(array) {
}
```

Next, it's important to know whether we should swap at all. If the array has already been swapped, meaning an entire pass has gone without swapping, we won't have to go through the array again. We need to create a variable that holds a boolean value, and only if that value is true, we want to swap values.

```
let swapped;
do {
  swapped = false;
  // Swap logic here! In here, swapped will be set to true.
} while (swapped);
```

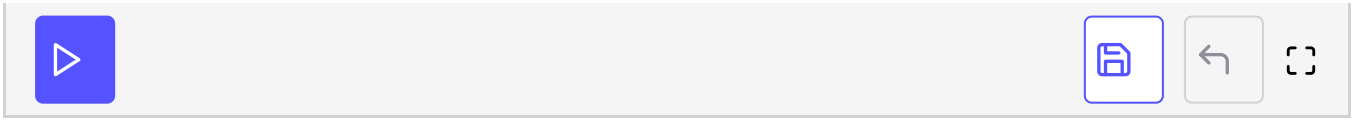
Right now, if swapped keeps on being false, the do-while will stop. We need to create the swapping logic. First, we need to loop over the array. Then, we check whether the current value, if it exists, is bigger than the next item's value, if that exists, meaning that they should swap.

```
for (let i = 0; i < array.length; i++) {
  if (array[i] && array[i + 1] && array[i] > array[i + 1]) {
     const temp = array[i];
     array[i] = array[i + 1];
     array[i + 1] = temp;
     swapped = true;
  }
}
```

If the if-statement returns true, we swap the items. We create a temporary variable that holds the value of the current element. It is necessary not to override this value in the second step! After the elements swapped, we set the

swapped variable to true, as at least two items swapped in this pass. The array is now sorted! We only need to return it.

```
function bubbleSort(array) {
  let swapped;
  do {
     swapped = false;
     for (let i = 0; i < array.length; i++) {
        if (array[i] && array[i + 1] && array[i] > array[i + 1]) {
          const temp = array[i];
          array[i] = array[i + 1];
          array[i + 1] = temp;
          swapped = true;
       }
     }
   } while (swapped);
  return array;
}
```



Now, let's discuss the time complexity of this algorithm.