Advanced internet programming

Project no. 3 – Sorting algorithms in Javascript

Hubert Bęczkowski, ID: 307290

The task at hand was to create a

- List of randomly generated numbers
- An algorithm that would sort the numbers:
  - Quick sort
  - Merge sort

Quick and Merge sort the randomly generated numbers

Generate random numbers

10,1,8,11,2,0,0,10,11,2,5,4

Try Quick Sort

0,0,1,2,2,4,5,8,10,10,11,11

Try Merge Sort

0,0,1,2,2,4,5,8,10,10,11,11

Quick sort

```
<script>
  function quickSort(arr) {
    if (arr.length <= 1) return arr;
    var pivot = arr[0];
    var before = [];
    var after = [];
    for (var i = 1; i < arr.length; i++) {
      arr[i] <= pivot ? before.push(arr[i]) : after.push(arr[i]);
    }
    return quickSort(before).concat(pivot, quickSort(after));
  }
  function quick() {
    const numbers = document
      .getElementById("num")
      .innerHTML.split(",")
      .map(Number);
    document.getElementById("q").innerHTML = quickSort(numbers);
  }
</script>
```

The algorithm is fairly simple, as it takes the arr[0] value as it pivots, and checks whether or not given values that are to be sorted are bigger or smaller than it. If sorted values are smaller, they are put on the left from the pivot, in other case they are put on the right.

Merge sort

```
<script>
  function merge(before, after) {
    let resultArray = [];
    while (before.length && after.length) {
      if (before[0] < after[0]) {
        resultArray.push(before.shift());
      } else {
        resultArray.push(after.shift());
      }
    }
    return [...resultArray, ...before, ...after];
  }
  function mergeSort(arr) {
    if (arr.length <= 1) return arr;
    var middle = Math.floor(arr.length / 2);
    var before = mergeSort(arr.slice(0, middle));
    var after = mergeSort(arr.slice(middle));
    return merge(before, after);
  }
  function mergeAll() {
    const numbers = document
      .getElementById("num")
      .innerHTML.split(",")
      .map(Number);
    document.getElementById("mer").innerHTML = mergeSort(numbers);
  }
</script>
```

Merge sort cuts the length of the array into smaller parts, which are later compared together and based on the comparison done by the algorithm are grouped depending on if they are bigger or smaller than eachother. The lower values are pushed to the left of the array, while the higher to the right. Then, the pieces are merged together, resulting into a sorted array.