1DA1611:A
Advanced Internet Programming

Project nr. 3 Report: 2023-03-09

Illia Priadko (309062)

Plan:

- Define a set of functions in JavaScript, responsible for:
  - creating an array of 10 random numbers
  - merge-sort algorithm
  - quick-sort algorithm
- Define an HTML page as a front-end to the JS code
- Testing

Given task:

Tasks

1. Generate a list of numbers in JavaScript using a Random function
2. Sort the numbers using one sorting algorithm (Bubble-Sort, Selection Sort, Quick-Sort, Merge Sort)
3. Print the obtained list of sorted number
4. Redo the points 1-3 using the second method

Illia	Priadko	Merge, Quick

1. Initial function definition

Initially the functions were called to console.log outputs, but after changing the scope to include a basic HTML layout, basic output was replaced to references to those elements.

Code is commented for readability:

```
    /* Generate and print initial list of numbers */
    function generateArray() {
    let arr = [];
    for (let i = 0; i < 10; i++) {
    arr.push(Math.floor(Math.random() * 100) + 1);
    }
    document.getElementById("originalArray").innerHTML =
    "Original Array: " + arr.join(", ");
    return arr;
}
    /* Merge two arrays function */
    function merge(lArray, rArray) {
        let sortedArr = [];
        while (lArray.length && rArray.length) {
        if (lArray[0] <= rArray[0]) {
            sortedArr.push(lArray[0]);
            lArray = lArray.slice(1);
        } else {
            sortedArr.push(rArray[0]);
```

```
            rArray = rArray.slice(1);
        }
        }
        while (lArray.length) sortedArr.push(lArray.shift());
        while (rArray.length) sortedArr.push(rArray.shift());
        return sortedArr;
    }
    /* Merge-sort algorithm function */
    function mergeSort(arr) {
        if (arr.length <= 1) {
        return arr;
        }
        else {
        let mid = Math.floor(arr.length / 2);
        let lArray = arr.slice(0, mid);
        let rArray = arr.slice(mid, arr.length);
        return merge(mergeSort(lArray), mergeSort(rArray));
        }
    }

    /* Quick-sort algorithm function */
    function quickSort(arr) {
        if (arr.length <= 1) {
        return arr;
        } else {
        let pivotIndex = Math.floor(arr.length / 2);
        let pivot = arr.splice(pivotIndex, 1)[0];
        let lArray = [];
        let rArray = [];
        for (let i = 0; i < arr.length; i++) {
            if (arr[i] < pivot) {
            lArray.push(arr[i]);
            } else {
            rArray.push(arr[i]);
            }
        }
        return quickSort(lArray).concat([pivot], quickSort(rArray));
        }
    }
    /* Action for when the button is pressed */
    function execute() {
    let arr = generateArray();
    let sorterMerge = mergeSort(arr);
    let sortedQuick = quickSort(arr);
    document.getElementById("sortedMerge").innerHTML =
    "Sorted Array (Merge Sort): " + sorterMerge.join(", ");
    document.getElementById("sortedQuick").innerHTML =
    "Sorted Array (Quick Sort): " + sortedQuick.join(", ");
}
```

2. Define an HTML page

The layout should include three basic elements:

- Header of the page
- “Execute” button
- Display of initial and sorted values

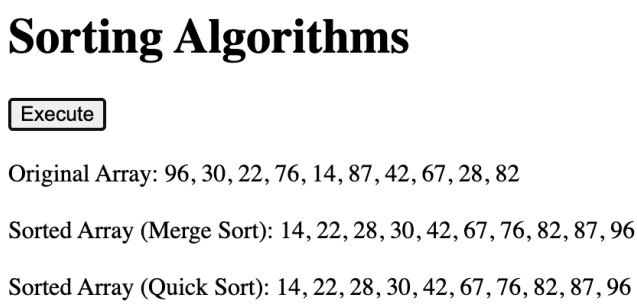
Appending to the above JS code is the following HTML layout:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Sorting Algorithms</title>
   </head>
   <body>
     <h1>Sorting Algorithms</h1>
     <button onclick="execute()">Execute</button>
     <br><br>
     <div id="originalArray"></div>
     <br>
     <div id="sortedMerge"></div>
     <br>
     <div id="sortedQuick"></div>
     <script>
```

```
  …
    </script>
  </body>
</html>
```

The page looks as follows:

And once the “Execute” button is pressed…



We can verify that both algorithms work as intended and produce the same result.

Testing

Testing was carried out inside of Chrome (or any Chromium-based browser for that matter, Edge is shown here).

The page is also hosted on http://10.44.99.99:23880/~309062/sort.html

Conclusion:

After having completed this project, I got familiar with JavaScript basic syntax for function definitions, as well as sorting algorithms. I was able to successfully use and implement these algorithms into a user-friendly HTML page using IDs.