Report Project n.3+4

First task

- Generate a list of numbers in JavaScript using a Random function

First of all I wrote the html part creating button that onclick call some function:

```
<label >Insert number of random number to be sorted:</label><input type="number"
id="number">
    <p>Random number generated: <span id="generate"></span></p>
  <p>Bubble sorted numbers: <span id="bubble"></span></p>
  <p>Time elaboration: <span id="timebubble"></span></p>
  <p>Quick sorted numbers: <span id="quick"></span></p>
  <p>Time elaboration: <span id="timequick"></span></p>
    <button onclick="generate()">Generate</button>
  <button onclick="BubbleFull()">Bubble</button>
  <button onclick="QuickFull()">Quick</button>
```

Than in JS part I initialized a global array variable:

```
var array=[];
```

Than I created the function to generate random number:

```
function generate(){
    array=[];
    var number = document.getElementById("number").value;

 for(i=0;i<number;i++){
   array[i]=Math.floor(Math.random() * 100) + 1;
 }
    document.getElementById("generate").innerHTML= array;
}
```

At the start I cleaned the array variable so it is possible to overwrite the generate number every time we press the “Generate” button.

Than I just took the value in input that represent the number of random numbers that we want to generate (the random numbers generated are in the range 0-100 and are integer number thanks to Math.floor function that round the number generated by Math.random)

Second task

- Sort the numbers using one sorting algorithm (Bubble-Sort, Selection Sort, Quick-Sort, Merge Sort)
- Print the obtained list of sorted number
- Redo the points 1-3 using the second method

For sorting I used Bubble sort (optimized method) and Quick sort

Implemented in this way:

Bubble sort:

```
function bubble(arr){
  var i, j;
  var len = arr.length;

  var isSwapped = false;

  for(i =0; i < len; i++){

    isSwapped = false;

    for(j = 0; j < len; j++){
        if(arr[j] > arr[j + 1]){
          var temp = arr[j]
          arr[j] = arr[j+1];
          arr[j+1] = temp;
          isSwapped = true;
        }
    }

    // IF no two elements were swapped by inner loop, then break

    if(!isSwapped){
      break;
    }
  }
  document.getElementById("bubble").innerHTML= array;
}
```

For printing the value I just changed the innerHTML (with the id=”bubble”)with the array sorted by the bubble method

For Quick sort before writing the code for sorting I implemented 2 more function:

```
function swap(items, leftIndex, rightIndex){
    var temp = items[leftIndex];
    items[leftIndex] = items[rightIndex];
    items[rightIndex] = temp;
}
function partition(items, left, right) {
    var pivot   = items[Math.floor((right + left) / 2)], //middle element
        i       = left, //left pointer
        j       = right; //right pointer
    while (i <= j) {
        while (items[i] < pivot) {
            i++;
        }
        while (items[j] > pivot) {
            j--;
        }
        if (i <= j) {
            swap(items, i, j); //sawpping two elements
            i++;
            j--;
        }
    }
    return i;
}
```

And then the quick sort code:

```
function quickSort(items, left, right) {
    var index;
    if (items.length > 1) {
        index = partition(items, left, right); //index returned from partition
        if (left < index - 1) { //more elements on the left side of the pivot
            quickSort(items, left, index - 1);
        }
        if (index < right) { //more elements on the right side of the pivot
            quickSort(items, index, right);
        }
    }
    document.getElementById("quick").innerHTML= array;
}
```

With at the end the same method to save the array in the HTML part and print it.

Project n.4

- The same task as for Project no. 3 but with time measuring

For this task I created new function that calculate the time of processing of each method:

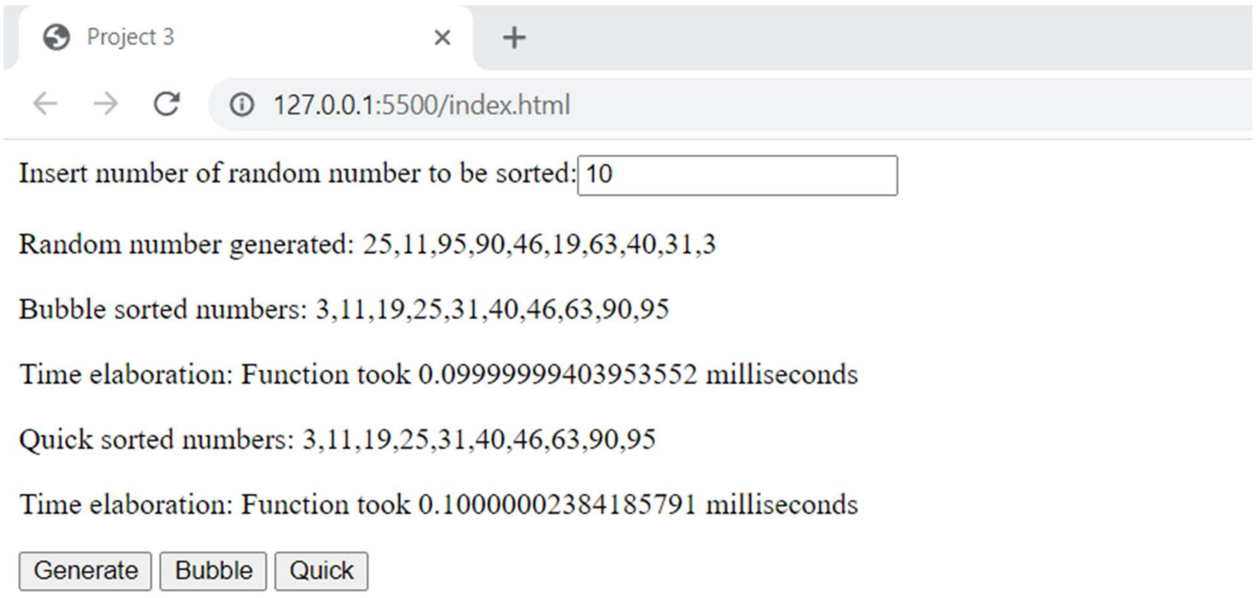
```
function BubbleFull(){
    start = performance.now();
    bubble(array);
    end = performance.now();
    timeTaken = end - start;
    document.getElementById("timebubble").innerHTML="Function took " + timeTaken
+ " milliseconds";

}
function QuickFull(){
    start = performance.now();
    quickSort(array, 0, array.length - 1);
    end = performance.now();
    timeTaken = end - start;
    document.getElementById("timequick").innerHTML="Function took " + timeTaken +
" milliseconds";

}
```

The now() method of the performance interface returns a high-resolution timestamp whenever it is called during the program. The time can be measured by getting the starting time before the function and the ending time after the function and then subtracting both of them. This gives the time elapsed for the function.

Final result:



```
Insert number of random number to be sorted: 10

Random number generated: 25,11,95,90,46,19,63,40,31,3

Bubble sorted numbers: 3,11,19,25,31,40,46,63,90,95

Time elaboration: Function took 0.09999999403953552 milliseconds

Quick sorted numbers: 3,11,19,25,31,40,46,63,90,95

Time elaboration: Function took 0.10000002384185791 milliseconds
```