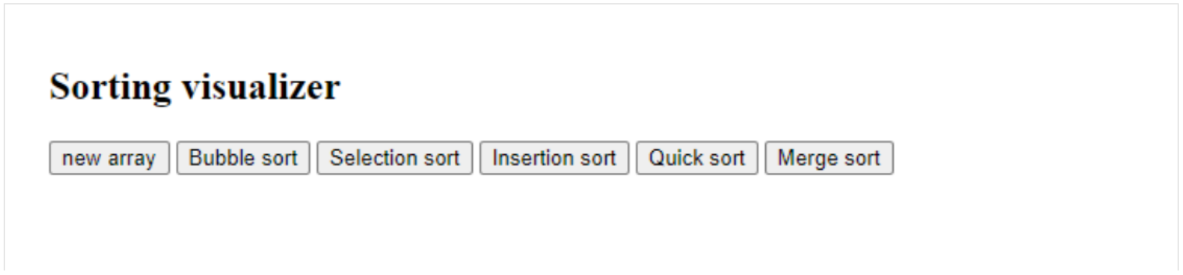
Step 1

- First component of the website is to give a heading using the HTML heading tag.
- Then the main components are to create 5 buttons for running the sorting algorithms (bubble sort, selection sort, insertion sort, quick sort, merge sort) and another button to generate new arrays. Create all these buttons using the HTML button tag.
- And wrap them with the appropriate id's and classes which will then be used for reference in styling in CSS and to select them and also to add event listeners in Javascript code (to be done in the upcoming milestones).

Expected Outcome -

Since only HTML has been used the site should look something like this.



Step 2

- Give a background color to the website using CSS
- Use Bootstrap to add navbar for the top part of the web app and inside this navbar class provide all the buttons.
- Give all the appropriate class names and id to all the relevant sub structures like this (to be done in HTML code).

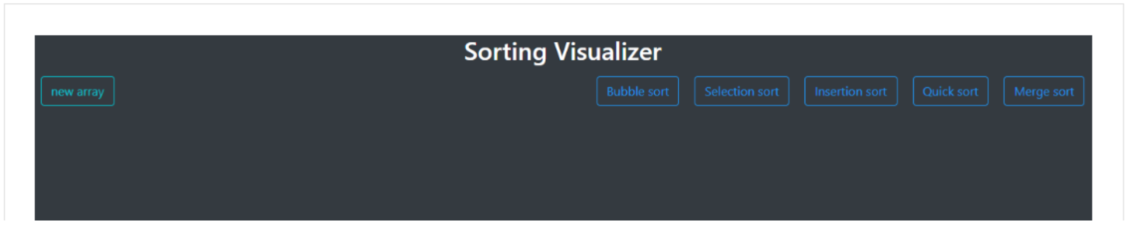
```
<button type="button" class="btn btn-dark bubble sort" >Bubble sort</button>
```

- For styling purposes you can refer to the image in the `Expected Outcome` section as your starter template. Do not think about the bars and other components except the buttons. Bars and other components will be

- addressed in the upcoming milestones. Feel free to innovate and come up with your own styles.

Expected Outcome

After styling using CSS and Bootstrap the site should look something like this.

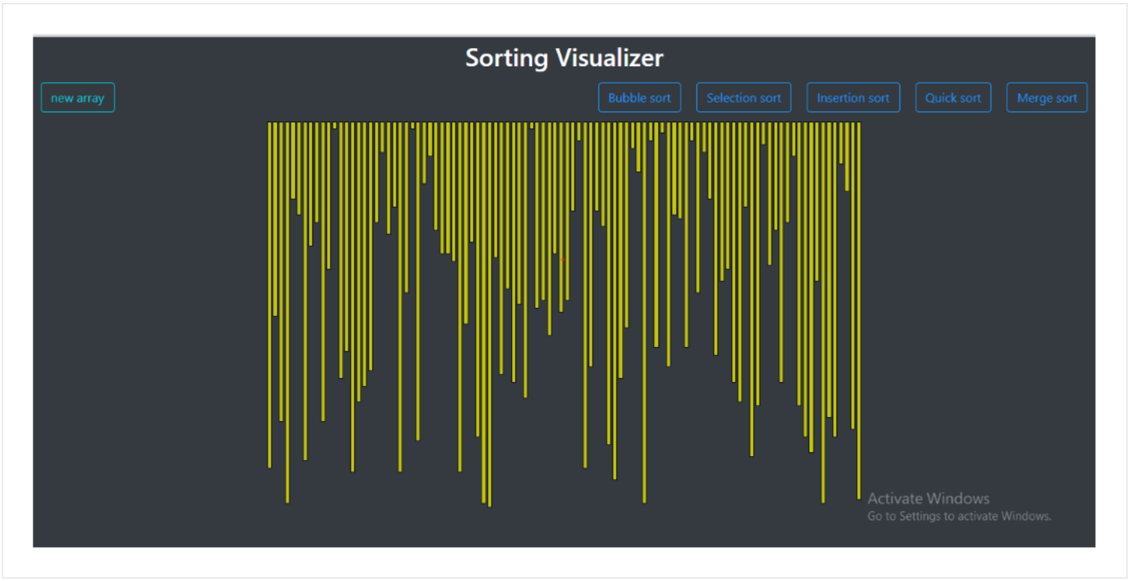


Step 3

- In the JS file just create an array and push 100 numbers (Don’t worry we will implement the number of bars changing functionality in the upcoming modules).
- Create 100 numbers using random function , convert those numbers to an integer number in the range 0-100 (you may take any range).
- The array integers should be the height of bars.
- Now inside the HTML file under the navbar create a division (div) and also give an id in this division where we will be placing all the bars' components.
- Now coming to the JS file we will create 100 div elements (creating elements using JS).
- Using JS add a particular class to all divs (so that we can add styles to all the div in CSS) and all div will have different heights equal to array elements (choose an appropriate scale) (Changing the CSS property using JS).
- Push every bar in that particular div , defined in the HTML file under navbar using JS.
- Wrap all this in a function and make a call to that function.
- Also add event listeners to the new array button and inside that call the function . So that you can use that button to create a new bar every time without refreshing the page.
- In the CSS file you can add styles to the bars inside the class for bars.

Expected Outcome -

After adding the bars the site looks something like this.



Step 4

- The most important thing to do in every sorting algorithm is to swap elements. To make swap two elements in HTML using JS you can do it this way.

```
function swap(el1,el2)
{

    const style1 = window.getComputedStyle(el1);
    const style2 = window.getComputedStyle(el2);


    const transform1 = style1.getPropertyValue("height");
    const transform2 = style2.getPropertyValue("height");

    el1.style.height = transform2;
    el2.style.height = transform1;

}
```

- Now apply the simple bubble sort algorithm. During the comparison of two elements make the background color red of both the bars and after the comparison convert the background color again to the default one for both the bars. You may use the following logic -

```
 special[j].style.background="red";
  special[j+1].style.background="red";
```

- At the end of every iteration when the highest bar will be taken to the right corner then to show that this bar is placed at it's perfect position make the background color Green in the above way.
- Now when you run this you will notice that there is no delay in swapping and the other iterations. So you have to add delay before the swaps in order to watch how changes are happening. For delay may use the following logic -

```
await new Promise(resolve => setTimeout(() => {resolve(), delay(2)}));
```

- Wrap this whole thing in a function and pass this into the event listener of the bubble sort button.

Implementation of remaining Sorting function

Again before starting this task understand the Selection Sort, Insertion Sort, quick Sort, Merge sort algorithms thoroughly.

Changing the number of bars and speed

Now as you must have observed from the earlier app's demo we need to change the number and speed of the bars. This can be done mainly by attributing each bar with a relative value, so that it becomes a pictorial representation of the array's elements that are being sorted.

Requirements

- For this we can use the input element in the navbar. In the HTML file input should be like this.

```
   <span>

              //no of bars
```

```
            <input id="arr_sz" type="range"  min=20 max=120 step=1
value=60>

    </span>
```

- And then in the javascript code part, using DOM we will select the input tag and take the value from that and pass it onto the create bar function. Instead of 100 we will use the number of bars as the inputted (to be taken) value from the input tag and in the delay function pass the delay time as the taken input from the speed input like the below code.

```
  var arr_size=document.querySelector("#arr_sz");



 var no_of_bar=arr_size.value;
```

- Also add event listeners with no bar and pass create bar function this way.

```
 arr_size.addEventListener("input",create_bars);
```

Tips :

- Add an event listener to the number of bars because when we will use that the no of bars should change instantly.