Sorting Visualizer

Project Description

If you are searching for a new JavaScript Project idea and want to dive deep into JavaScript or want to take your JavaScript skills to the next level, this is the perfect project for you. At the end of this module you will have a platform where anyone can visualize how sorting algorithms works and you also can showcase your HTML, CSS, Bootstrap, JavaScript Skills.

Author

Swapnoneel Dutta Majumdar

Collaborator(s)

Kevin Paulose, Kiran Suresh

Project Language(s)

JavaScript

Difficulty

Intermediate

Duration

20 hours

Prerequisite(s)

HTML, CSS, JavaScript (introductory level)

Skills to be learned

JavaScript, Sorting Algorithms, Web application development, Web Hosting

Overview

Objective

Create a web application using HTML, CSS, Javascript to visualize how various sorting algorithms work. This project's functionality will be similar to this application.

Project Context :

We have learnt sorting algorithms like bubble sort, selection sort, insertion sort, quick sort. But often we fail to understand the core idea of a particular algorithm maybe because we are unable to visualize how they work. So the most important thing to understand about these algorithms is visualization.

That's why we are making this project to let everyone understand how these algorithms work and through this project you also will get a deep understanding of such sorting algorithms.

This project will guide you step by step to complete this project and at the end of this project you will have an immense grip on some core concepts of Javascript as well. Adding this project on your resume will showcase your skills and add a great value to your profile.

This project is a good start for beginners and a refresher for professionals who have dabbled in data structures and algorithms using Javascript before and also web developers. The methodology can be applied to showcase any algorithm of one's choosing, so feel free to innovate!

High-Level Approach

- Creating the website's User Interface (UI) using HTML, CSS and enhancing it further using Bootstrap; without actually implementing any of the app's core features.
- Implementation of animations, effects and core functionalities (sorting algorithms) using JavaScript.
- Publish to GitHub and host your project live using Netlify.

At the end of this project you will have built an app which would function like this -

https://www.youtube.com/embed/tIfR6bHXMCw

Task 1

Getting Started

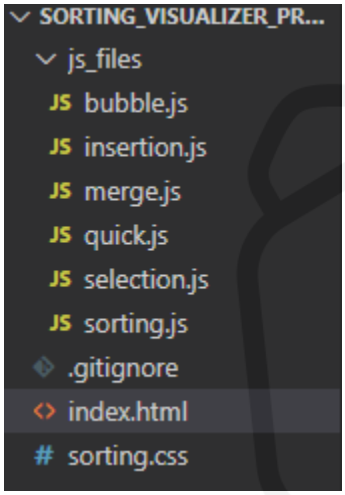
First validate the idea by doing a low level implementation (Proof of concept) of the components involved in the project.

- Get more clarity around the unknowns. Eg: XML vs HTML and why to use HTML5/CSS3 not the other versions, advantages of using Bootstrap.

- Get a better understanding of the stages involved in the project. Eg: By doing a proof of concept you will understand that there are multiple stages such as creating the basic layout, styling it and implementing the functionalities.

Requirements

- This is a typical JavaScript Project , so you need a code editor like VScode (recommended), Atom, Sublime text, etc. with necessary plugins.
- Then create an appropriate project folder with essential files. It's a good practice to follow the suggested file structure (shown below).



Reference

- Getting started with HTML
- Getting started with CSS
- JavaScript Crash Course
- About .gitignore file

Task 2

Create the website's UI

In this milestone the basic structure of this website will be made. In this milestone you will mainly use HTML. Then in the next milestone we will add Bootstrap and CSS for styling purposes.

Requirement

- 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).

Reference

- HTML Cheat Sheet

Expected Outcome -

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

Task 3

Improving UI using CSS and Bootstrap

The web app's basic skeleton UI was created in the previous task. To make the app more attractive and interactive we will employ CSS and Bootstrap for styling purposes.

Requirement

- Give a background color to the website using CSS
- Use Bootstrap to add a 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 substructures 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.

Reference

- Bootstrap Docs
- CSS Tips
- CSS Cheat Sheet

Tip

- To check your site's styling you should use the chrome developer tool as it provides minute-specific details of your site's components.
- To understand how to use Bootstrap just go through `Bootstrap Docs` given in the references. You can simply copy the code for your favourite outcome from the website and paste it to your code editor. Then you can customize elements in your own way.

Expected Outcome

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

Crio.Do

Task 4

Creating Bars Using JavaScript

From this milestone onwards we will start implementing the animations and other core functionalities of the application. In this milestone, we will create bars of different heights; which basically indicates the array that we will sort. Through these bars, we will visualize how sorting algorithms work.

Requirement

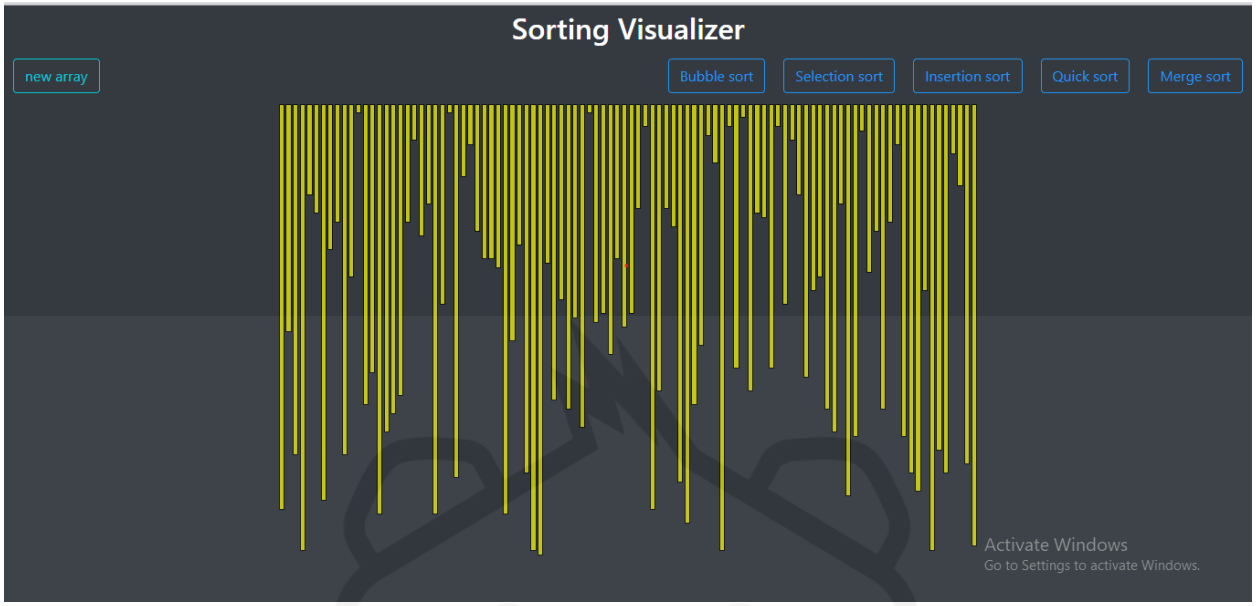
- In the JS file just create an array and push 100 numbers (Don't worry we will implement the number of bars changing functionality in the upcoming modules).
- Create 100 numbers using a random function, convert those numbers to an integer number in the range 0-100 (you may take any range).
- The array 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 divs 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.

References

- Event listener
- JavaScript HTML DOM changing css
- Creating new element
- JavaScript HTML DOM changing HTML

Expected Outcome -

After adding the bars the site looks something like this.



Task 5

Implementing Bubble Sort Algorithm

Before starting this task, understand the Bubble Sort algorithm thoroughly.

Requirements

- 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 for 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 its 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 a 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.

References-

- Async func
- Await

Task 6

Implementation of remaining Sorting function

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

Tips

- These sorting algorithms' implementations are the same as for the Bubble sort algorithm. In every algorithm's implementation to distinguish every comparison, swaps and iterations just change the colors and animation effects in your own way.

Task 7

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.

Expected Outcome

After implementing all the functionalities the end result of your app should be like this -

https://www.youtube.com/embed/tIfR6bHXMCw

Task 8

Host your website live

After completing all the milestones we have our application ready to be deployed and hosted live onto the web.

Start off by pushing your code to your GitHub account with a good README.md to publish your project.

Host your app live using Netlify and share its link among your peers and finally do add this project to your resume.

Reference

- How to publish on GitHub
- How to deploy on Netlify