Ajax Assignment

Deadline: Friday, JUL 24 2020, 09:00 AM

GitHub Classroom Link: https://classroom.github.com/a/UhD4quSd

REMEMBER TO SET REPO TO PRIVATE UNTIL AFTER DUE DATE

Introduction

This assignment is meant to challenge your knowledge of using AJAX/AXIOS to fetch data from an API. In industry, you will often be fetching data from an API for a variety of reasons: whether it be to analyze real-time data from another source, or taking mock data to fill up your application for the purpose of testing.

Requirements

- ❏ Fetch at least 3 different sets of data from one of the following APIs (you may choose any API you wish) https://github.com/public-apis/public-apis
- ❏ The data you fetch must be displayed on the user's browser in a neat and orderly fashion
- ❏ Include in your repository a screenshot of the result of postman

Challenges

- ❏ Add a search function for the user to fetch specific sets of data (For example: If you have a weather API, allow the user to enter their location and get the data specific to their location)
- ❏ Make it look nice with CSS
- ❏ Process the data received from the API before outputting to the user
- ❏ Use Axios instead of FETCH (See Axios documentation online)

Hints

- Look over what was done in class! It should contain all the tools necessary to get started on this assignment
- Remember that you need an index.html and js/scripts.js file in your project
- Focus on the requirements before moving on to the challenges! Challenges are extra
- Comment your code and commit/push your work regularly

Rubric

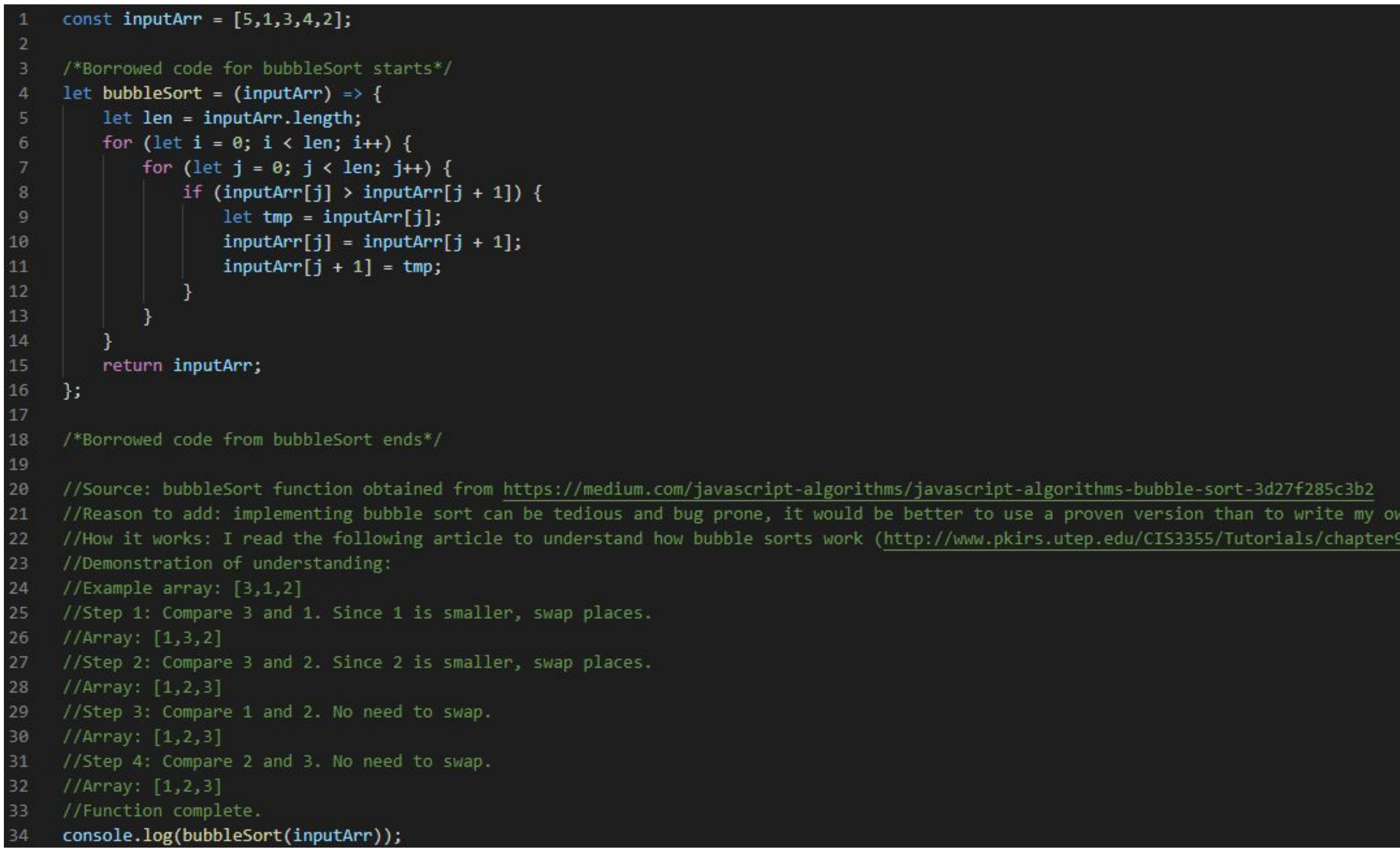
You will be evaluated on the following points. You must get all 3 of the Mandatory points to pass:

Requirement	**Points**
MANDATORY: ● Fetch at least 3 different sets of data from one of the following APIs (you may choose any API you wish) https://github.com/public-apis/public-apis ● The data you fetch must be displayed on the user's browser in a neat and orderly fashion ● Include in your repository a screenshot of the result of postman	3
CHALLENGE: ● Add a search function for the user to fetch specific sets of data (For example: If you have a weather API, allow the user to enter their location and get the data specific to their location) ● Make it look nice with CSS ● Process the data received from the API before outputting to the user ● Use Axios instead of FETCH (See Axios documentation online)	4
Total:	3

Citation Guide

Whenever you borrow code, the following information must be included:

- ❏ Comments to indicate both where the borrowed code begins and ends.
- ❏ A source linking to where you found the code.
- ❏ Your reason for adding the code to your assignment/project instead of writing it out yourself
- ❏ How it works. Explain to us how the code is supposed to work, include links to documentation/articles you read to help you understand.
- ❏ A small demonstration to prove you understand how the code works.



```javascript
const inputArr = [5,1,3,4,2];

/*Borrowed code for bubbleSort starts*/
let bubbleSort = (inputArr) => {
    let len = inputArr.length;
    for (let i = 0; i < len; i++) {
        for (let j = 0; j < len; j++) {
            if (inputArr[j] > inputArr[j + 1]) {
                let tmp = inputArr[j];
                inputArr[j] = inputArr[j + 1];
                inputArr[j + 1] = tmp;
            }
        }
    }
    return inputArr;
};

/*Borrowed code from bubbleSort ends*/

//Source: bubbleSort function obtained from https://medium.com/javascript-algorithms/javascript-algorithms-bubble-sort-3d27f285c3b2
//Reason to add: implementing bubble sort can be tedious and bug prone, it would be better to use a proven version than to write my o
//How it works: I read the following article to understand how bubble sorts work (http://www.pkirs.utep.edu/CIS3355/Tutorials/chapter
//Demonstration of understanding:
//Example array: [3,1,2]
//Step 1: Compare 3 and 1. Since 1 is smaller, swap places.
//Array: [1,3,2]
//Step 2: Compare 3 and 2. Since 2 is smaller, swap places.
//Array: [1,2,3]
//Step 3: Compare 1 and 2. No need to swap.
//Array: [1,2,3]
//Step 4: Compare 2 and 3. No need to swap.
//Array: [1,2,3]
//Function complete.
console.log(bubbleSort(inputArr));
```