University of St Andrews

CS5002- Programming Principles and Practice

Assignment 3: Calculator

Report

Student Id: 180029803

Table of Contents

1. Introduction ………………….……………………………………………………………………2
2. Design ……………………..………………………………………………………………………3
   2.1 Task 1 ……………………………………………………………………………………….3
   2.2 Task 2 ……………………………………………………………………………………...4
3. Implementation ……………………………………………………………………………...4
4. Testing ………………….……………………………………………………………………….4
5. Difficulties ………………………………………………………………………………………5
6. Selection Sort vs Bubble Sort ……………………………………………………………….6
7. Conclusion ……………………………………………………………………………………….7

1. Introduction

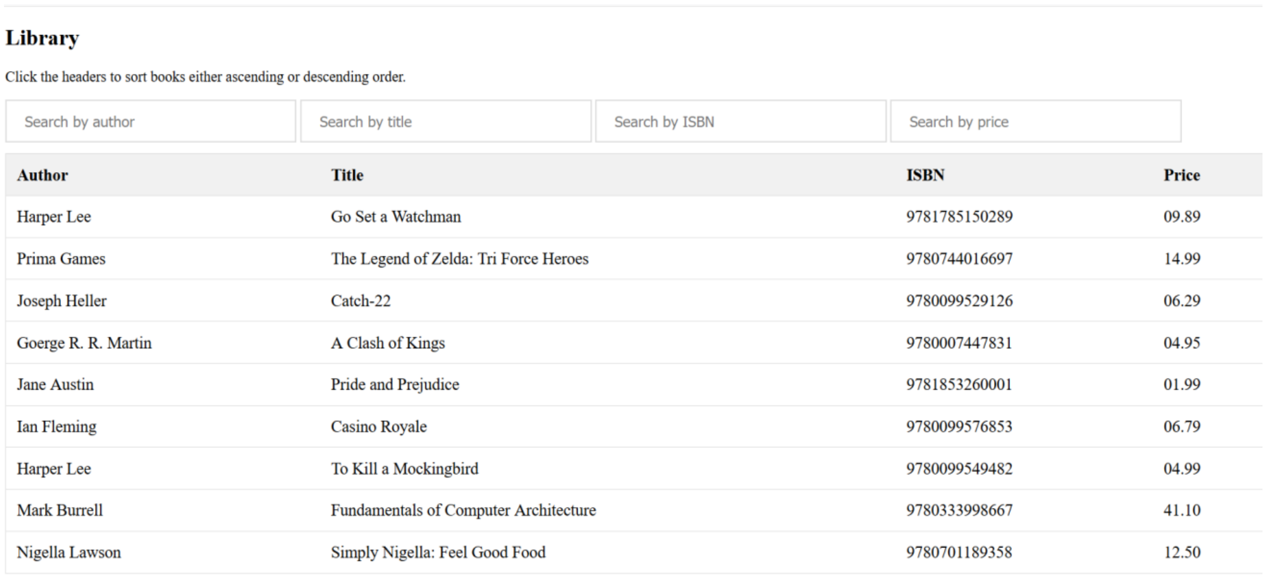
The aim of the assignment is creating functions that are directed by the various commands to display the desired book details. In order to implement those functions, a small database for the book store is provided. The necessary features of these functions are being able to display the book details when the user searches by author, title, price or ISBN. Also, the list of the books is expected to be sorted either by ascending or descending. Moreover, the second task asks to demonstrate calculations by conditions through if functions. All of the tasks require knowledge about arrows, conditions, calling functions, deleting the arrays or the element of arrays to perform them. After completing the previous tasks and achieving deep idea about sorting and filtering the arrays, the third and the last task requires to do a research on two simple methods of sorting numbers which are Selection Sort and Bubble Sort and demonstrate the difference between them if there is by the obtained information through both research and experience with the tasks.

This report deeply explains the method of creating the functions for demonstrating the arrays, the calculations to verify the elements of arrays, the drawbacks of the selected codes and the difficulties in implementing. Also, it supports the provided arguments by giving necessary evidences including screenshots of the tests that are done on the console and the browser and references for the research.

2. Design

2.1 Task 1

The task 1 asks to print all of the attributes of the books shown over the desired selection of the user. In order to ease the search of the books for the user and increase the functionality for the future use, instead of performing the requirements separately, the structure of the JavaScript is designed as including the sort and the filter functions at the same time. The system displayed on the browser does not just provide a single book but show all of them to the user in one time. However, if the database would be bigger than the current one, the user will not be able to view the details easily. Therefore, the scripts include a textbox where the user can enter the input in order to search the book by author, title, price or ISBN through. The written code for the textbox not only eases the search by filtering them, but also gives hint to user by showing the output while typing the input since it doesn't require a system similar to submitting the forms which cannot be seen before submitting the input. Moreover, searching is eased even more by providing a sorting function which orders the data from A to Z, Z to A, small numbers to bigger, and bigger numbers to smaller. The design is supported by an external HTML and CSS files to draw a more preferable looking and while at the same time, allowing developers to read the codes and use them easily in the future. In short, although meeting all of the 5 requirements of the first task (printAllBooks, printByTitle, printByAuthor, printOverPrice) is more difficult than performing them separately, especially with displaying on the browser rather than console, the selected design is more convenient by being easy to give outputs and running the codes faster than writing functions for printing them separately.



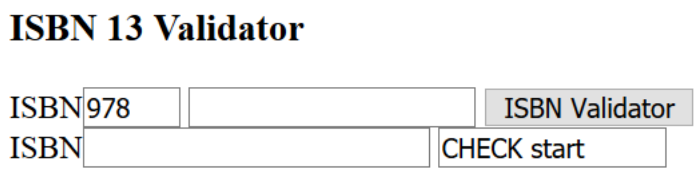
Library

Click the headers to sort books either ascending or descending order.

Author	Title	ISBN	Price
Harper Lee	Go Set a Watchman	9781785150289	09.89
Prima Games	The Legend of Zelda: Tri Force Heroes	9780744016697	14.99
Joseph Heller	Catch-22	9780099529126	06.29
Goerge R. R. Martin	A Clash of Kings	9780007447831	04.95
Jane Austin	Pride and Prejudice	9781853260001	01.99
Ian Fleming	Casino Royale	9780099576853	06.79
Harper Lee	To Kill a Mockingbird	9780099549482	04.99
Mark Burrell	Fundamentals of Computer Architecture	9780333998667	41.10
Nigella Lawson	Simply Nigella: Feel Good Food	9780701189358	12.50

2.2 Task 2

For the same reason with task 1, in the task 2 the text box is provided to the users in order to allow them to know whether the ISBN is valid or not. The design small text box on the left side includes 3-digit numbers which are the initial numbers of the 13-digit ISBN because by this way, the user obtains a hint while entering the number. The text box located below prints the same input with the entered one by adding the first 3 digits and the box with a text 'Check start' changes according to the validity of the entered ISBN by letting the user know that the ISBN is valid or not.



ISBN 13 Validator

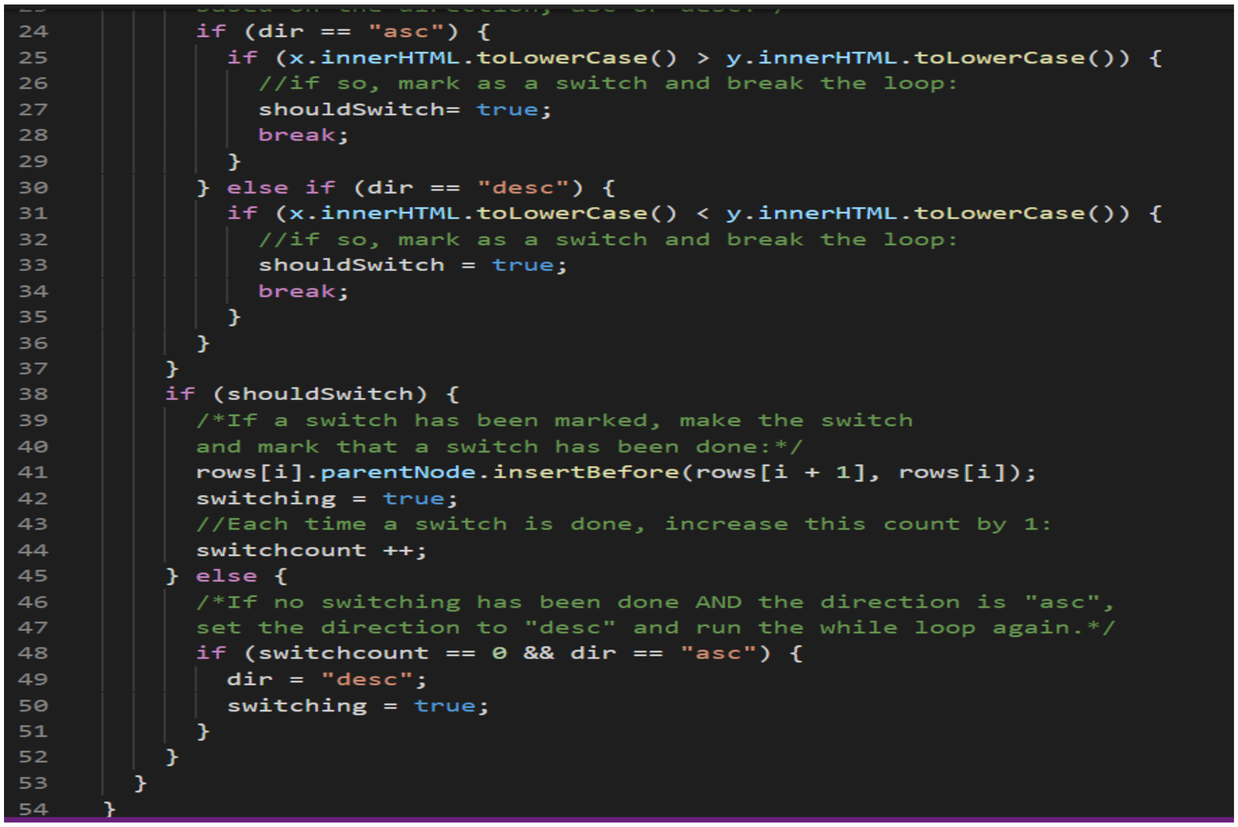
Since making the calculations for the ISBN validity is complex providing functions that can be viewed via browsers are difficult because the codes which are used to create the functions for the first requirement of task 2 block the functions of the second requirement. Therefore, another script file that can be tested on the console was created to perform the function.

3. Implementing

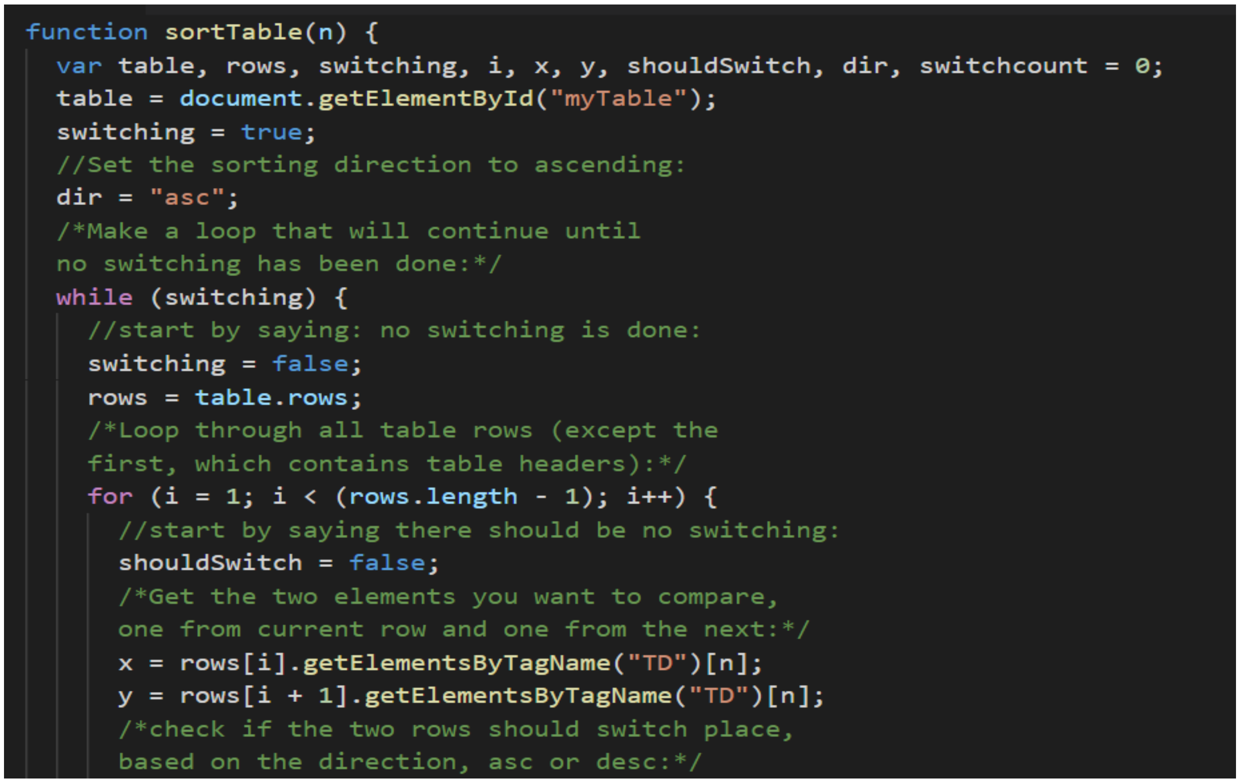
3.1 Task 1

The first two screenshots show the SortTable function that defines the conditions for the sorting order of the rows in the table through using if and else attributes. The function indicates that if the length of the lower-cased element is higher than the element below element, the function gives command to sort the list ascending (A to Z), if not, to sort the list descending (Z to A). GetElementbyId is used to call the code for system to realize whether the sorting is happened. The function can be used both for numbers and strings. The user can sort the lists by clicking on the header which s/he desires to view.

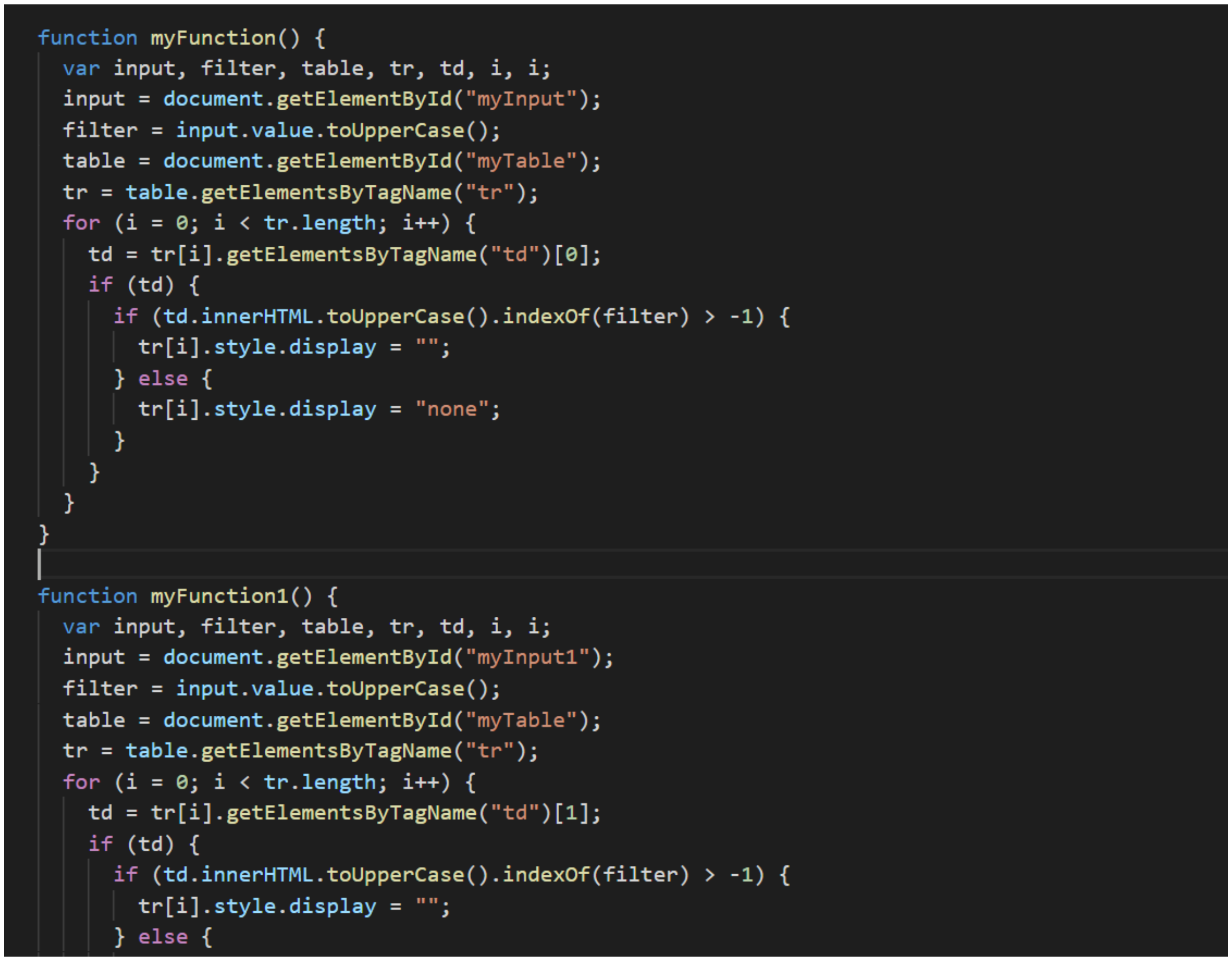
The last screenshot demonstrates the function that works to filter the elements of the rows in the lists. When the user enters the input into the textbox that s/he wants to search by, the filtering system analyzes it, searches in the whole database and provides the output that is same with the input. Since getElementbyId cannot be used for more than one row, there are 4 functions that filters 4 different rows of the table by the filtering system.



```
                if (dir == "asc") {
                  if (x.innerHTML.toLowerCase() > y.innerHTML.toLowerCase()) {
                    //if so, mark as a switch and break the loop:
                    shouldSwitch= true;
                    break;
                  }
                } else if (dir == "desc") {
                  if (x.innerHTML.toLowerCase() < y.innerHTML.toLowerCase()) {
                    //if so, mark as a switch and break the loop:
                    shouldSwitch = true;
                    break;
                  }
                }
              }
              if (shouldSwitch) {
                /*If a switch has been marked, make the switch
                and mark that a switch has been done:*/
                rows[i].parentNode.insertBefore(rows[i + 1], rows[i]);
                switching = true;
                //Each time a switch is done, increase this count by 1:
                switchcount ++;
              } else {
                /*If no switching has been done AND the direction is "asc",
                set the direction to "desc" and run the while loop again.*/
                if (switchcount == 0 && dir == "asc") {
                  dir = "desc";
                  switching = true;
                }
              }
            }
          }
```



```
function sortTable(n) {
  var table, rows, switching, i, x, y, shouldSwitch, dir, switchcount = 0;
  table = document.getElementById("myTable");
  switching = true;
  //Set the sorting direction to ascending:
  dir = "asc";
  /*Make a loop that will continue until
  no switching has been done:*/
  while (switching) {
    //start by saying: no switching is done:
    switching = false;
    rows = table.rows;
    /*Loop through all table rows (except the
    first, which contains table headers):*/
    for (i = 1; i < (rows.length - 1); i++) {
      //start by saying there should be no switching:
      shouldSwitch = false;
      /*Get the two elements you want to compare,
      one from current row and one from the next:*/
      x = rows[i].getElementsByTagName("TD")[n];
      y = rows[i + 1].getElementsByTagName("TD")[n];
      /*check if the two rows should switch place,
      based on the direction, asc or desc:*/
```



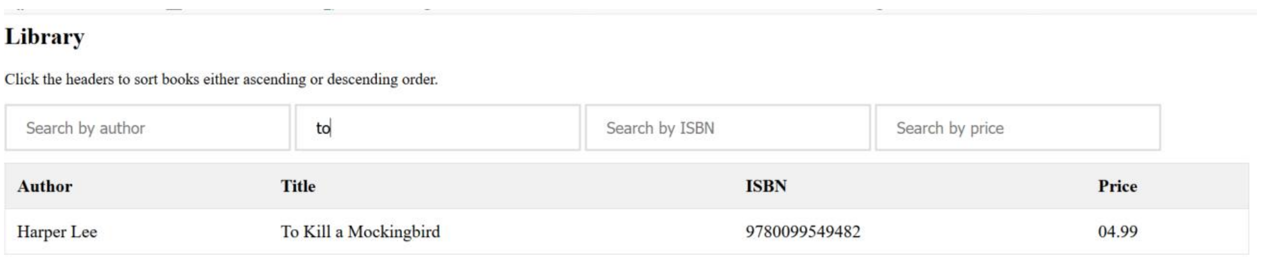
```
function myFunction() {
  var input, filter, table, tr, td, i, i;
  input = document.getElementById("myInput");
  filter = input.value.toUpperCase();
  table = document.getElementById("myTable");
  tr = table.getElementsByTagName("tr");
  for (i = 0; i < tr.length; i++) {
    td = tr[i].getElementsByTagName("td")[0];
    if (td) {
      if (td.innerHTML.toUpperCase().indexOf(filter) > -1) {
        tr[i].style.display = "";
      } else {
        tr[i].style.display = "none";
      }
    }
  }
}

function myFunction1() {
  var input, filter, table, tr, td, i, i;
  input = document.getElementById("myInput1");
  filter = input.value.toUpperCase();
  table = document.getElementById("myTable");
  tr = table.getElementsByTagName("tr");
  for (i = 0; i < tr.length; i++) {
    td = tr[i].getElementsByTagName("td")[1];
    if (td) {
      if (td.innerHTML.toUpperCase().indexOf(filter) > -1) {
        tr[i].style.display = "";
      } else {
```

3.2 Task 2

4. Testing

After taking off the HTML code from the JavaScript, the Task 1 started not to sort the list except filtering it as below:

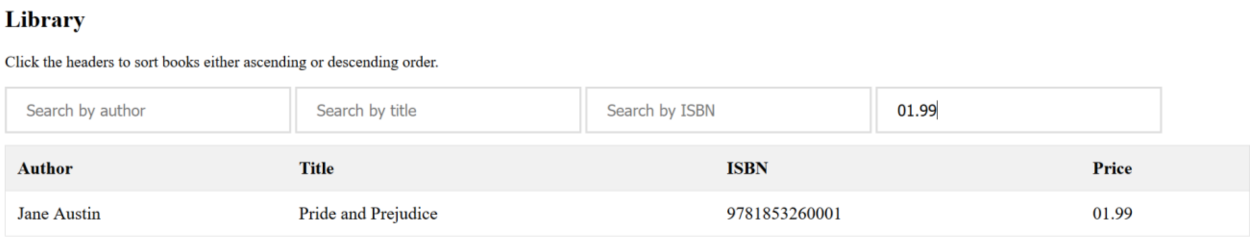


Library

Click the headers to sort books either ascending or descending order.

Search by author	to	Search by ISBN	Search by price

Author	Title	ISBN	Price
Harper Lee	To Kill a Mockingbird	9780099549482	04.99

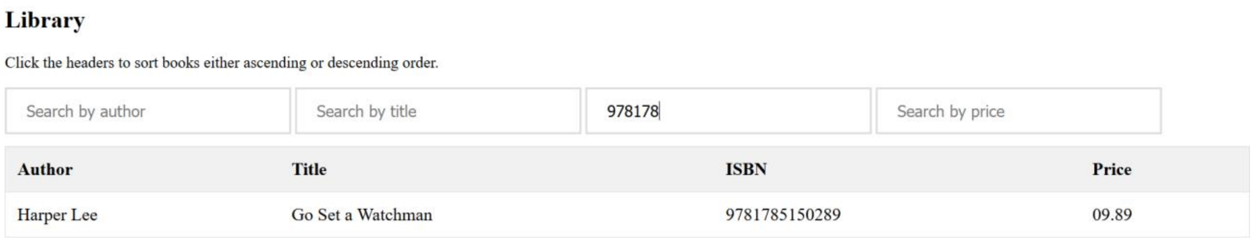


Library

Click the headers to sort books either ascending or descending order.

Search by author	Search by title	Search by ISBN	01.99

Author	Title	ISBN	Price
Jane Austin	Pride and Prejudice	9781853260001	01.99

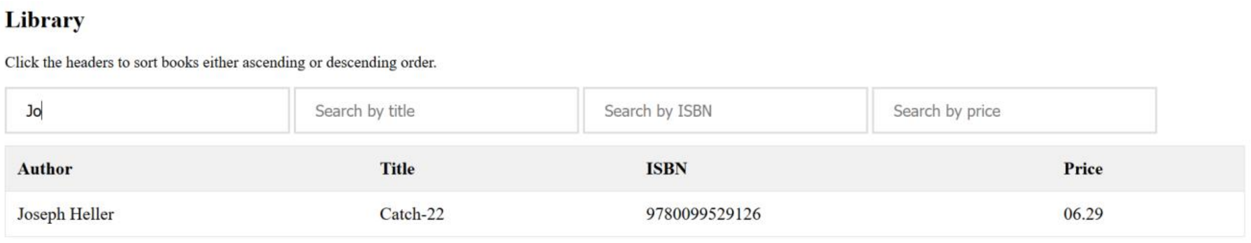


Library

Click the headers to sort books either ascending or descending order.

Search by author	Search by title	978178	Search by price

Author	Title	ISBN	Price
Harper Lee	Go Set a Watchman	9781785150289	09.89



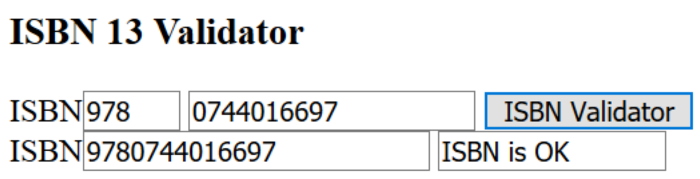
Library

Click the headers to sort books either ascending or descending order.

Jo	Search by title	Search by ISBN	Search by price

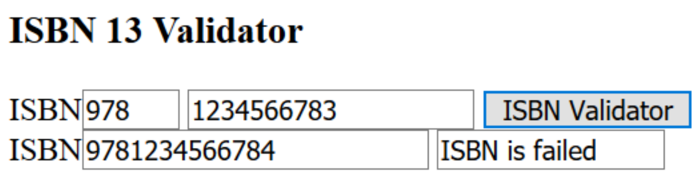
Author	Title	ISBN	Price
Joseph Heller	Catch-22	9780099529126	06.29

The text box for the task 2 works properly by warning whether the ISBN is valid or not as it is shown below. Also, when the input length is shorter or longer than it should be, the system alerts the user with an alert window.


ISBN 13 Validator

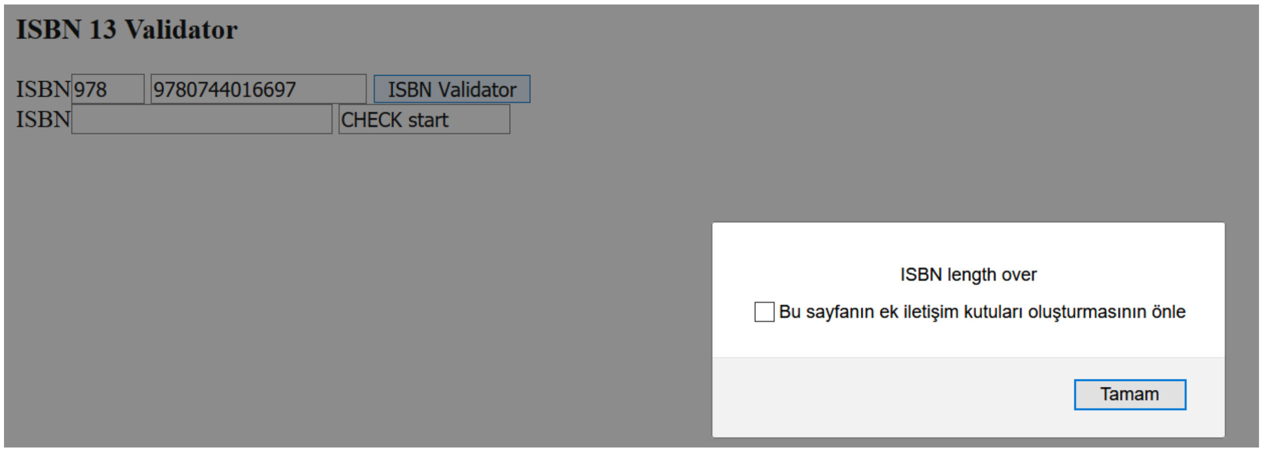
ISBN 978 0744016697 ISBN Validator

ISBN 9780744016697 ISBN is OK


ISBN 13 Validator

ISBN 978 1234566783 ISBN Validator

ISBN 9781234566784 ISBN is failed


ISBN 13 Validator

ISBN 978 9780744016697 ISBN Validator

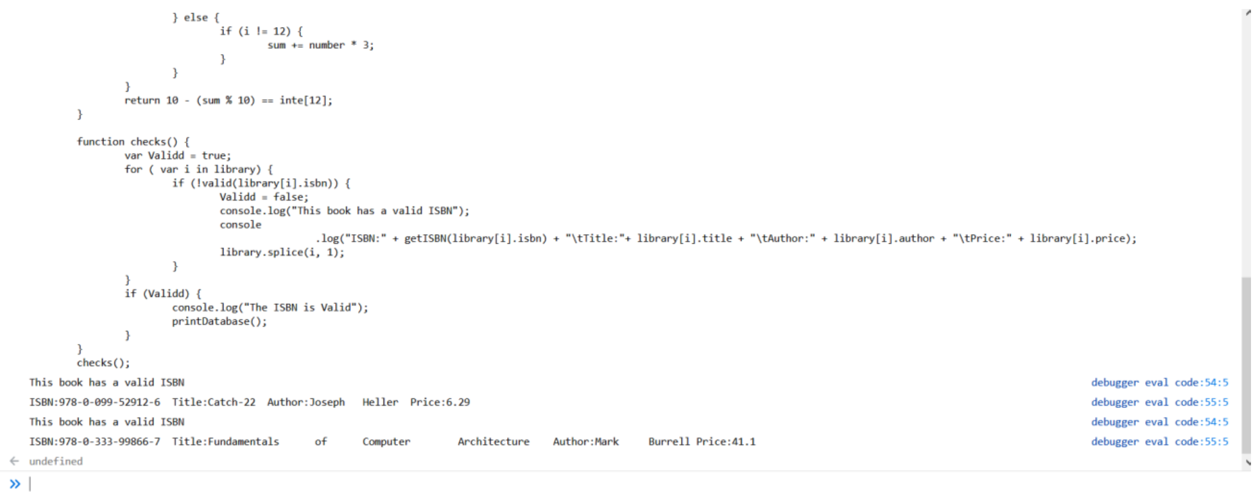
ISBN CHECK start

ISBN length over

Bu sayfanın ek iletişim kutuları oluşturmasının önle

Tamam

The script and the function for the second requirement of the task 2 runs properly.



```
                } else {
                        if (i != 12) {
                                sum += number * 3;
                        }
                }
        }
        return 10 - (sum % 10) == inte[12];
}

function checks() {
        var Validd = true;
        for ( var i in library) {
                if (!valid(library[i].isbn)) {
                        Validd = false;
                        console.log("This book has a valid ISBN");
                        console
                                        .log("ISBN:" + getISBN(library[i].isbn) + "\tTitle:"+ library[i].title + "\tAuthor:" + library[i].author + "\tPrice:" + library[i].price);
                        library.splice(i, 1);
                }
        }
        if (Validd) {
                console.log("The ISBN is Valid");
                printDatabase();
        }
}
checks();
This book has a valid ISBN                                                          debugger eval code:54:5
ISBN:978-0-099-52912-6  Title:Catch-22  Author:Joseph   Heller  Price:6.29          debugger eval code:55:5
This book has a valid ISBN                                                          debugger eval code:54:5
ISBN:978-0-333-99866-7  Title:Fundamentals     of     Computer       Architecture    Author:Mark     Burrell Price:41.1    debugger eval code:55:5
← undefined
» |
```

5. Difficulties

- Combining the functions for the requirements to put in a same JavaScript file inhibited functions to run.
- Since calculating the ISBN to check its validity requires good mathematical knowledge and it might lead inaccuracies if the data that is dealing with will be bigger.
- Defining the difference between bubble sort and selection by testing it is hard as long as there isn't any big data.

6. Selection Sort vs Bubble Sort

Selection sort is a sorting algorithm, specifically an in-place comparison sort. It is inefficient on large lists, and generally performs worse than the similar insertion sort. Selection sort is noted for its simplicity, and it has performance advantages over more complicated algorithms in certain situations, particularly where auxiliary memory is limited.

Bubble sort is a simple sorting algorithm that repeatedly steps through the list to be sorted, compares each pair of adjacent items and swaps them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted.