Refilwe nqatyelwa

IWA_19 Capstone

About IWA_19 Capstone

I was required to to do a full audit of the existing javascript code as well as fix all critical issues, The goal is to highlight areas where improvements can be made to performance, readability and future maintainability.

I had a week to make sure that the user story is provided is met so that the app can work as intended.

Contents

```
index.html        styles.css        JS scripts.js M  X                    JS data.js  X

JS scripts.js > ...                                  JS data.js > ...
  1   //Imported these from the data.js file           1   // added const to these variables and e
  2   import { books, authors,                          2   export const BOOKS_PER_PAGE = 36;
      BOOKS_PER_PAGE, genres } from "./data.            3
      js";                                              4 > export const authors = { ...
  3                                                     92   }
  4                                                     93
  5                                                     94 > export const genres = { ...
  6                                                     163  }
  7                                                     164
  8                                                     165 > export const books = [ ...
  9                                                    25413   ]
 10                                                    25414
 11

JS orgnal.js 9+, M  X

JS orgnal.js > ...
  1   matches = books
  2   page = 1;
  3
  4
```
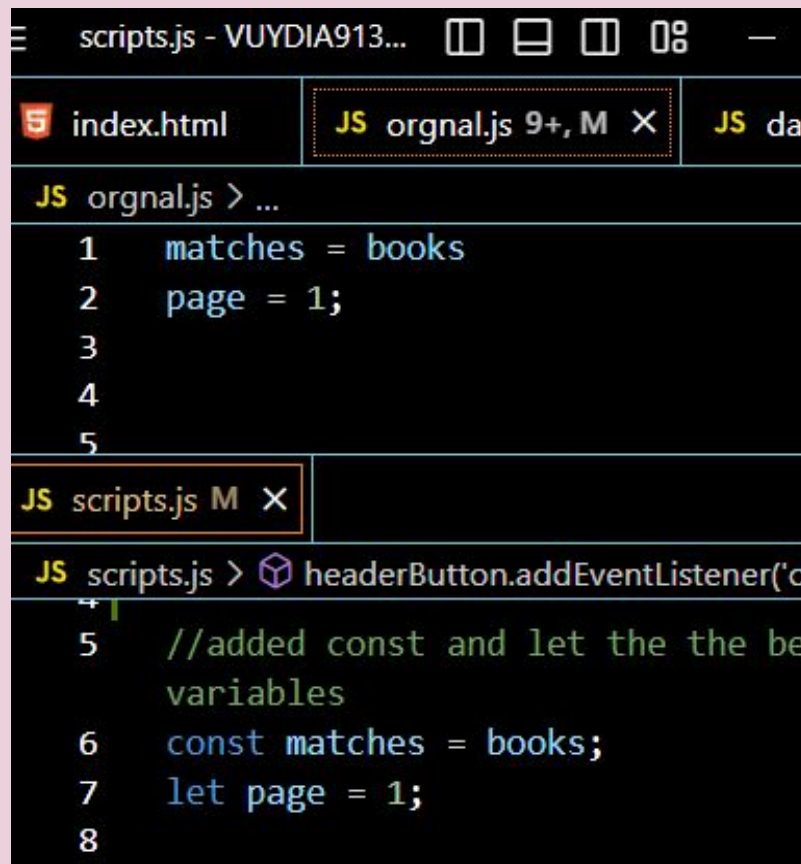
Exports and Imports

- 'BOOKS_PER_PAGE, books, authors, and genres' have to be imported from the 'data.js file'.
- Added 'const' and 'export' to BOOKS_PER_PAGE, books, authors, and genres.

EXAMPLE OF CONST

- Added 'const' to 'matches' on the 'scripts.js' file so that the variable can be defined.

EXAMPLE OF AN Error message

```
//scripts.js
// display message if no books match filters
if (filteredBooks .length > 0) {
document .querySelector ('[data-list-message]' ).innerText =
'';
document .querySelector ('[data-list-button]' ).disabled =
true;
document .querySelector ('[data-list-message]' ).style.marginT
op = '-125px';
            } else {
document .querySelector ('[data-list-message]' ).innerText =
'No results found. Your filters might be too narrow.';

document .querySelector ('[data-list-button]' ).disabled =
true;
 }

//orgnal.js
if (!books && !Array.isArray(books)) throw new
Error('Source required' );
if (!range && range.length < 2) throw new
Error('Range must be an array with two numbers' );
```

- This code checks if any books have passed the filters by checking the length of the filteredBooks array.

- If there are filtered books, it sets the text content of an element with the [data-list-message] attribute to an empty string, disables a button with the [data-list-button] attribute, and adjusts the margin of the [data-list-message] element.

- If there are no filtered books, it sets the text content of the [data-list-message] element to a message that tells the user that there are no results based on their filter criteria, and disables the [data-list-button] button.

FRAGMENT

- Added 'let' to 'fragment' as it will be changing.

- Fixed errors on the 'const' objects of the extracted and added 'let' to the fragment variable.

- Fixed the for loop syntax by removing the ';' and added 'of' for readability.

- The 'const preview' removed to be more clean and readable.

- Fixed the syntax and wrote code in a more correct and readable way.


```js
// orgnal.js > ...
19
20
21
22
23    fragment = document.
      createDocumentFragment()
24    const extracted = books.slice(0,
      36)
25
26
27    for ({ author, image, title,
      id }; extracted; i++) {
28        const preview = createPreview
          ({
29            author,
30            id,
31            image,
32            title
33        })
34
35
36        fragment.appendChild(preview)
37    }
38
39    data-list-items.appendChild
      (fragment)
40
41
42
43
44
45
46
47
48
49
```

```js
// scripts.js > ⬡ headerButton.addEventListener('click
9
10    // Showing 36 Books Per Page
11    let fragment = document.
      createDocumentFragment(); //
      added let to the fragment
12    const extracted = books.slice(0,
      36);
13
14
15    /**
16     * fixed errors on the const and
         added 'let' to the fragment
         variable
17     * fixed the syntax by removing
         the ';' and added 'of'on the
         for
18     * the const preview was
         re-ordered in a more readable
         manner
       */
19
20
21    for (const { author, title,
      image, id } of extracted) {
22        const preview = createPreview
          ({ author, id, image,
          title });
23        fragment.appendChild
          (preview);
24    }
25    //fixed the syntax and wrote
      code in a more readable way
26    document.querySelector("
      [data-list-items]").appendChild
      (fragment);
27
28
```

orgnal.js > ...

```js
49
50   genres = document.
     createDocumentFragment()
51   element = document.createElement
     ('option')
52   element.value = 'any'
53   element = 'All Genres'
54   genres.appendChild(element)
55
56
57   for ([id, name]; Object.entries
     (genres); i++) {
58       document.createElement('option')
59       element.value = value
60       element.innerText = text
61       genres.appendChild(element)
62   }
63
64   data-search-genres.appendChild
     (genres)
65
66
67
68
69
70
71
72
73
74
75
76
77
```

scripts.js > ⬡ headerButton.addEventListener('click') callback

```js
230
231
232       // Drop Down for Genres
233   const dataSearchGenres = document.
      querySelector("[data-search-genres]");
234   const allGenresOption = document.
      createElement("option");
235       allGenresOption.value = "any";
236       allGenresOption.innerText = "All
          Genres";
237       dataSearchGenres.appendChild
          (allGenresOption);
238   for (const [id, names] of Object.entries
      (genres)) {
239       const element = document.
          createElement("option");
240       element.value = id;
241       element.innerText = names;
242       dataSearchGenres.appendChild
          (element);
243   }
244
245   for (const [id, names] of Object.entries
      (genres)) {
246       const element = document.
          createElement("option");
247       element.value = id;
248       element.innerText = names;
249       dataSearchGenres.appendChild
          (element);
250   }
251
252
```

- Line element = 'All Genres' should be element.innerText = 'All Genres' to set the text of the option element.

-

- Fixed code in the line 'element' = 'All Genres' should be 'element.innerText' = 'All Genres' to set the text of the option element.

- Changed a for...of loop to iterate through the key-value pairs in the genres object for the code to use 'Object.entries' to create an array of key-value pairs, and then uses a 'for...of loop' to iterate through that array.

- Changed the 'var' to be 'const', to declare the loop variables.
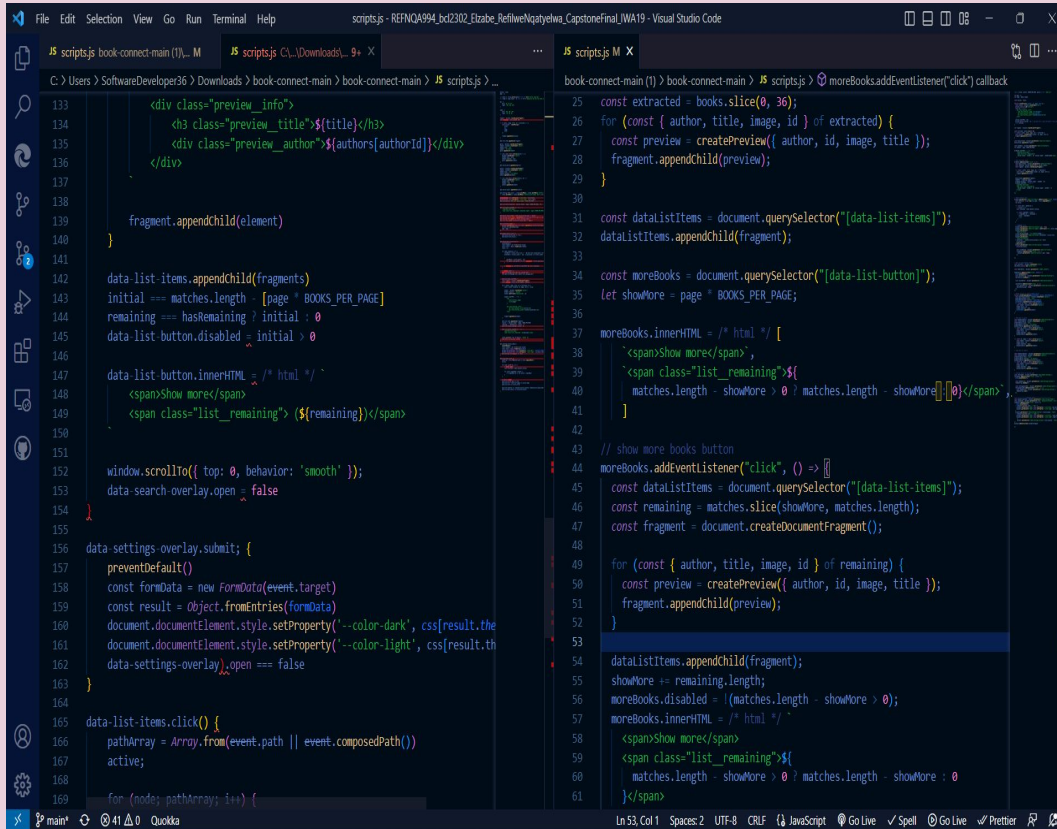
AUTHORS

- Changed the variable names.

- Added the correct attribute, instead of 'option', to ('[data-search-authors]')'

- Changed a for...of loop to iterate through the key-value pairs in the genres object for the code to use 'Object.entries' to create an array of key-value pairs, and then uses a 'for...of loop' to iterate through that array.

- Fixed the for...of loo and repeated the same code as for the genre code.

- Used 'dataSearchAuthors' as an element that will be added instead o 'data-search-authors'.

Show more books

- Added a let to fragment as it will be changing.

- The "showMore" variable tracks the number of items currently displayed and updates accordingly with each click. The "moreBooks" button is disabled if all book items have been displayed and its text changes to reflect how many items remain to be displayed.

- The "showMore" variable tracks the number of items currently displayed and updates accordingly with each click. The "moreBooks" button is disabled if all book items have been displayed and its text changes to reflect how many items remain to be displayed.

- // handle preview click is where i set the event listener for click events and searches through an array to find books that matches the preview dataset value, if it matches it populates the ovalay with the matched book element.
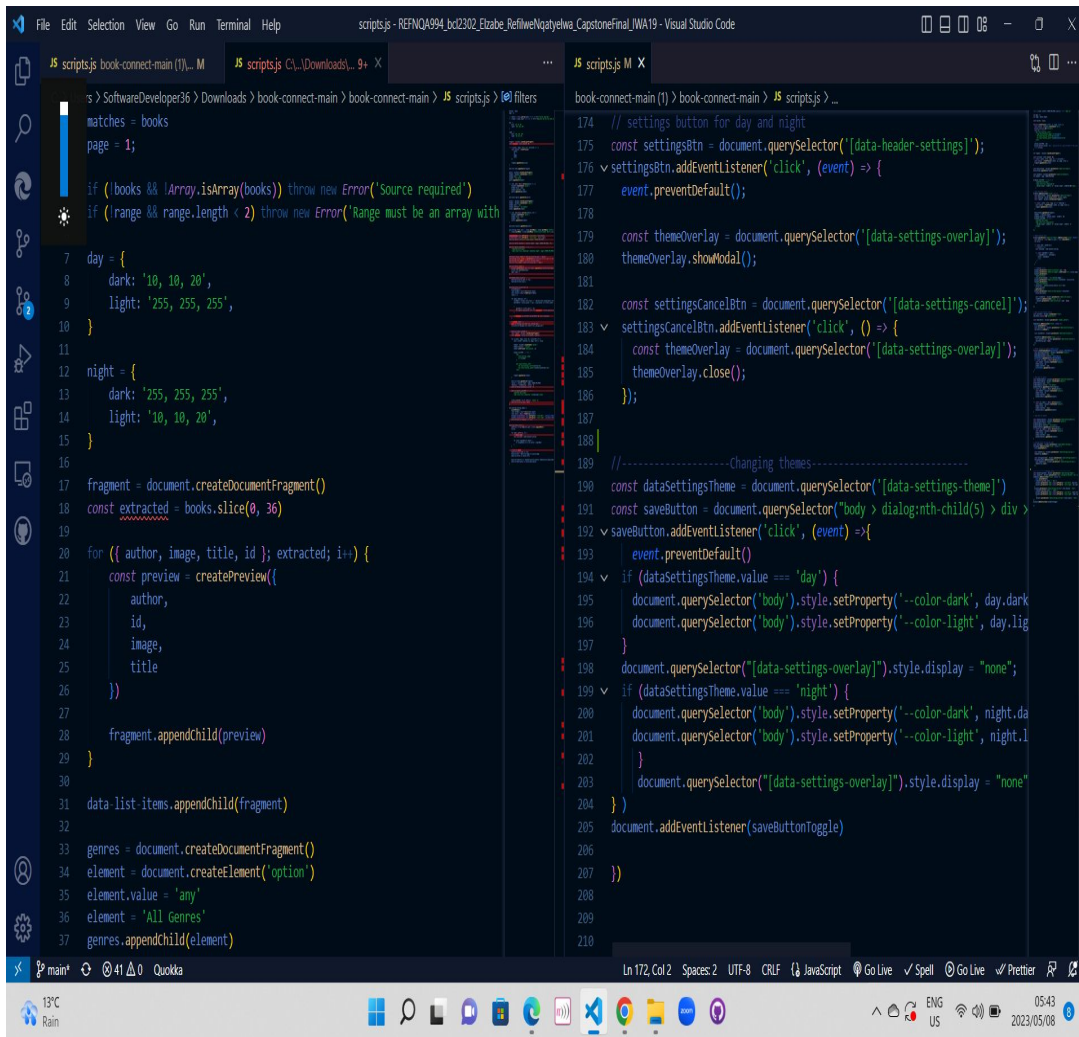
```javascript
                <div class="preview__info">
                    <h3 class="preview__title">${title}</h3>
                    <div class="preview__author">${authors[authorId]}</div>
                </div>

        fragment.appendChild(element)
    }

    data-list-items.appendChild(fragments)
    initial === matches.length - [page * BOOKS_PER_PAGE]
    remaining === hasRemaining ? initial : 0
    data-list-button.disabled = initial > 0

    data-list-button.innerHTML = /* html */ `
        <span>Show more</span>
        <span class="list_remaining"> (${remaining})</span>`

    window.scrollTo({ top: 0, behavior: 'smooth' });
    data-search-overlay.open = false
}

data-settings-overlay.submit; {
    preventDefault()
    const formData = new FormData(event.target)
    const result = Object.fromEntries(formData)
    document.documentElement.style.setProperty('--color-dark', css[result.the
    document.documentElement.style.setProperty('--color-light', css[result.th
    data-settings-overlay).open === false
}

data-list-items.click() {
    pathArray = Array.from(event.path || event.composedPath())
    active;

    for (node; pathArray; i++) {
```

```javascript
const extracted = books.slice(0, 36);
for (const { author, title, image, id } of extracted) {
    const preview = createPreview({ author, id, image, title });
    fragment.appendChild(preview);
}

const dataListItems = document.querySelector("[data-list-items]");
dataListItems.appendChild(fragment);

const moreBooks = document.querySelector("[data-list-button]");
let showMore = page * BOOKS_PER_PAGE;

moreBooks.innerHTML = /* html */ [
    `<span>Show more</span>`,
    `<span class="list_remaining">${
    matches.length - showMore > 0 ? matches.length - showMore : 0}</span>`
]

// show more books button
moreBooks.addEventListener("click", () => {
    const dataListItems = document.querySelector("[data-list-items]");
    const remaining = matches.slice(showMore, matches.length);
    const fragment = document.createDocumentFragment();

    for (const { author, title, image, id } of remaining) {
        const preview = createPreview({ author, id, image, title });
        fragment.appendChild(preview);
    }

    dataListItems.appendChild(fragment);
    showMore += remaining.length;
    moreBooks.disabled = !(matches.length - showMore > 0);
    moreBooks.innerHTML = /* html */ `
        <span>Show more</span>
        <span class="list_remaining">${
        matches.length - showMore > 0 ? matches.length - showMore : 0}
        </span>`
```

SEARCH BUTTON

- Created a search overlay form

- When button is clicked the search overlay is opened by calling its shoModal() method.

- The code allows the user to search for books either by any, genre or authour.

- The cancel button also has an event listener. When it is clicked, the event listener retrieves the search overlay element again and sets its "open" attribute to false, which closes the search overlay.

- The helper function is for the dropdown, the element takes the parent element to be added to the dropdownThe function creates an option element for the default option with a value of "any" and adds it to the parent element. It then iterates through the object of options and creates an option element for each option with a value of the option's id and text content of the option's name. These option elements are added to the parent element.

SETTING BUTTON

Using an if statement the settings button that toggles between day and night themes on a webpage. When the settings button is clicked, an overlay dialog is displayed. The user can then select a theme by clicking on either the day or night option. When the user clicks on the save button, the selected theme is applied to the webpage by changing the background colors of the page. Finally, the overlay dialog is closed.

CONCLUSSION

Ater I have debugged and fixed all the critical issues the app should be able to do all that the user asked for which are the following:

1. As a user, I want to view a list of book previews, by title and author, so that I can discover new books to read.
2. As a user, I want an image associated with all book previews so that I can recognize a book by the cover even if I forgot the name.
3. As a user, I want to have the option of reading a summary of the book so that I can decide whether I want to read it.
4. As a user, I want to have the option of seeing the date that a book was published so that I can determine how easy it is to obtain second-hand.
5. As a user, I want to find books based on specific text phrases so that I don't need to remember the entire title of a book.
6. As a user, I want to filter books by author so that I can find books to read by authors that I enjoy.
7. As a user, I want to filter books by genre so that I can find books to read in genres that I enjoy.
8. As a user, I want to toggle between dark and light modes so that I can use the app comfortably at night.