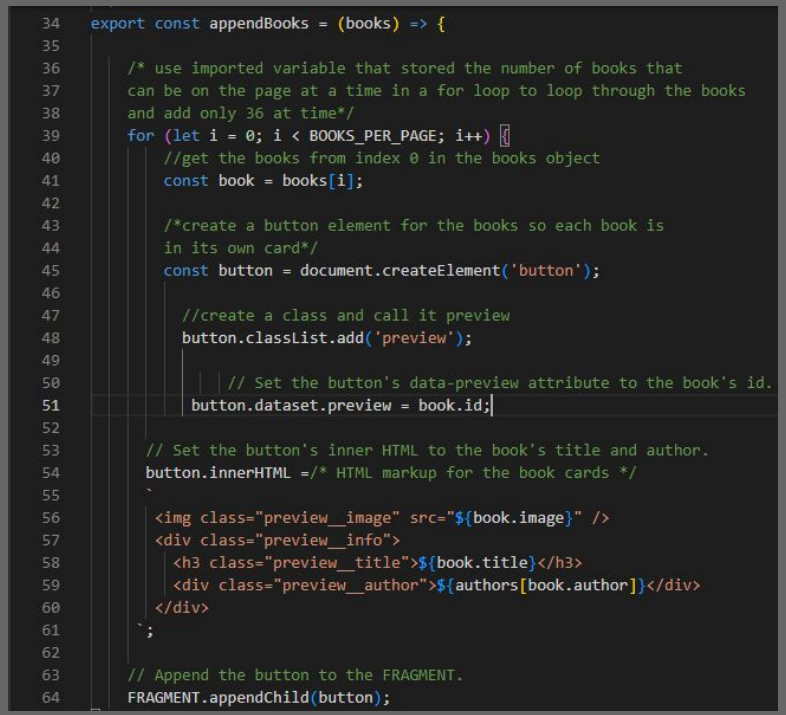
The Book Connect bug hunt

How I fixed Book Connect’s bugs.

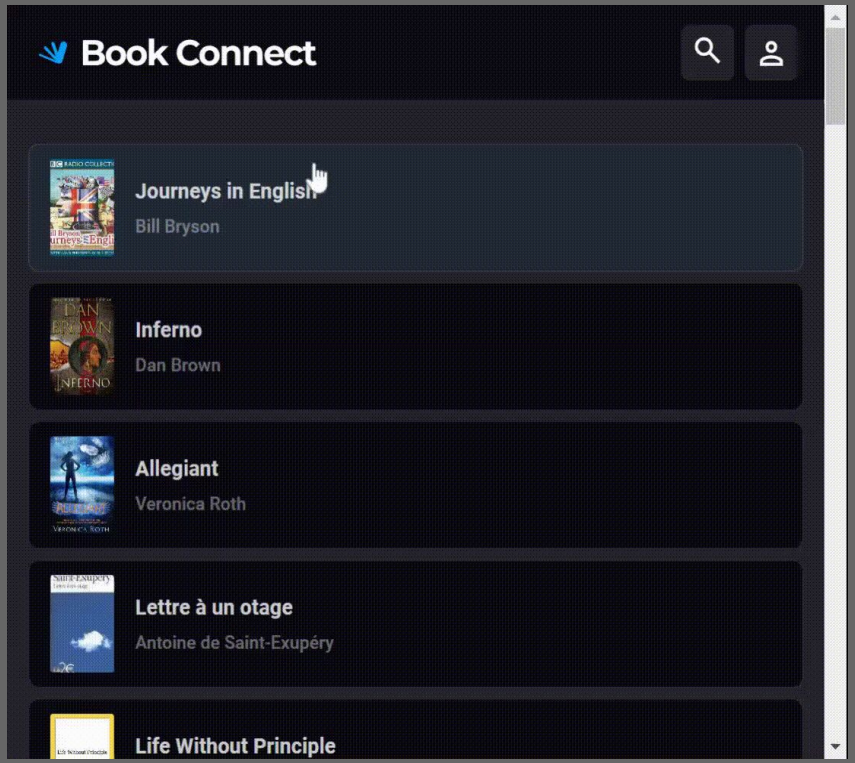
I want to view a list of book previews with an image, title and author.



```
export const appendBooks = (books) => {

    /* use imported variable that stored the number of books that
    can be on the page at a time in a for loop to loop through the books
    and add only 36 at time*/
    for (let i = 0; i < BOOKS_PER_PAGE; i++) {
        //get the books from index 0 in the books object
        const book = books[i];

        /*create a button element for the books so each book is
        in its own card*/
        const button = document.createElement('button');

          //create a class and call it preview
          button.classList.add('preview');

              // Set the button's data-preview attribute to the book's id.
          button.dataset.preview = book.id;

      // Set the button's inner HTML to the book's title and author.
      button.innerHTML =/* HTML markup for the book cards */
       `
       <img class="preview__image" src="${book.image}" />
       <div class="preview__info">
         <h3 class="preview__title">${book.title}</h3>
         <div class="preview__author">${authors[book.author]}</div>
       </div>
     `;

    // Append the button to the FRAGMENT.
    FRAGMENT.appendChild(button);
```

The home page book list was created using the **appendBooks** function. This function iterates through the **books** object, extracts the title, author, and image, and uses interpolation to generate button elements with the relevant information.

I want to view a list of book previews with an image, title and author.



The **appendBooks** function was added as an event handler for the “load” event on the root document element so that the books show on the page when the web page first loads.

```
60  /*-------------------------------------------------------HOME PAGE DISPLAY--------
61
62  // /* calling the function to load the page with book list using an event
63  // listener for when the page first loads  */
64  const root = document.documentElement;
65  root.addEventListener("load", appendBooks(books))
66
```

I want to view a list of book previews with an image, title and author.

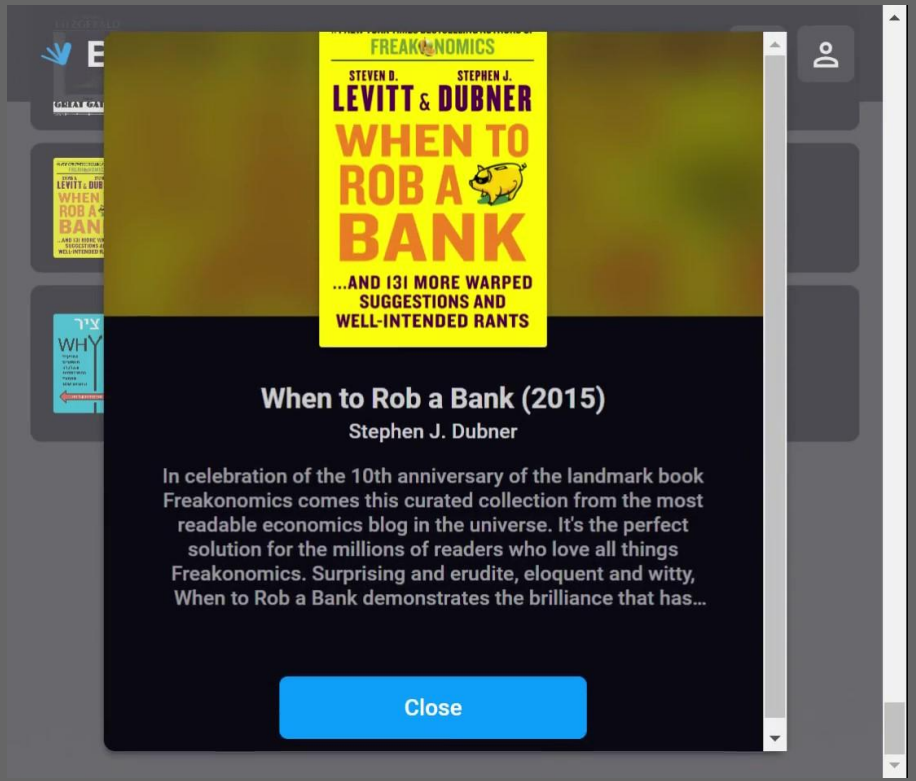
The **showMoreAction** function was used to add 36 more books to the page when the button is clicked.

```
export const showMoreAction = (event) => {
        event.preventDefault()
            /* fetch the books that are already on the page then count them and
        use the number of books left in the books object to add more books so the button
        can stop adding more books when all the books in the object have been added*/
        const booksOnPage = document.querySelectorAll('.preview');
        const booksOnPageCount = booksOnPage.length;
        //subtract books on page from total books in object
        const booksLeft = books.length - booksOnPageCount;
        //add the text to the button element

        //check if there are still books left in the books object
        if(booksLeft > 0) {
            /*add 36 more books to the page using the appendBooks function
            where the books object is altered by slicing out books
            from where the first function call ended to 36 more books*/
            appendBooks(books.slice(booksOnPageCount, booksOnPageCount + 36))
        }
            data.home.SHOW_MORE_BTN.innerHTML = `Show more <span class="list__remaining">(${booksLeft - BOOKS_PER_PAGE})</span>`

            /* make the summary overlay show when a book is clicked
    Used a for loop to iterate over all the book buttons so that
    each one can be clicked on
    NOTE - added here too so it can still work after the first
    36 books are added*/
    const bookList = document.querySelectorAll('.preview')
    for (let z = booksOnPageCount; z < books.length; z++ ) {
        bookList[z].addEventListener("click", descriptionOverlay )
    }
        };

```

```

/*use event listener to make button load more books with the
showMoreAction function*/
SHOW_MORE_BTN.addEventListener("click", showMoreAction)

```

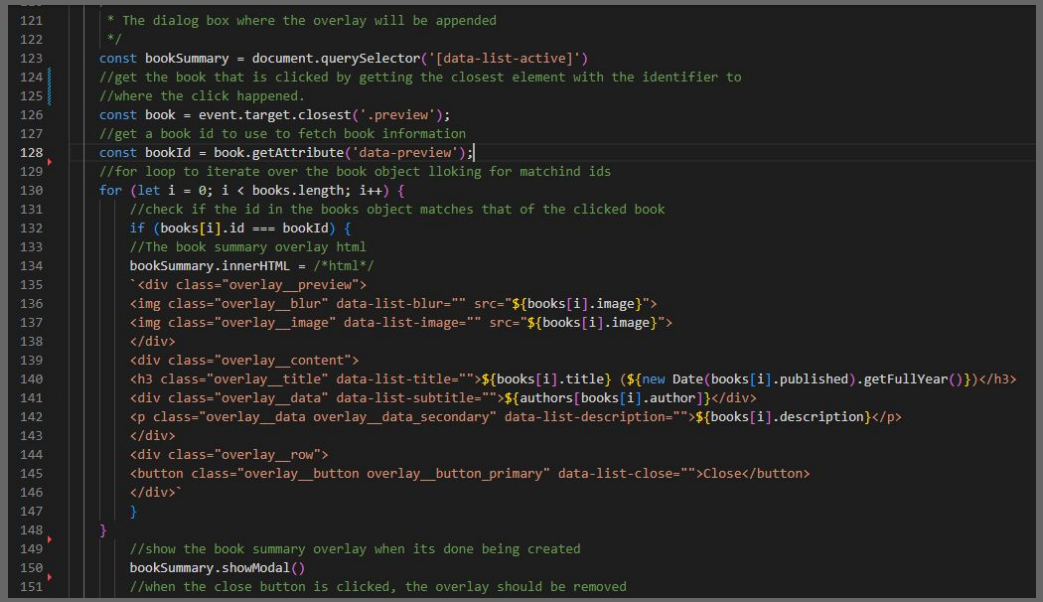
It will check that there number of books still not shown are more than 0, then run the **appendBooks** and **descriptionOverlay** functions again.

I WANT TO HAVE THE OPTION OF READING THE BOOK SUMMARY



The **descriptionOverlay** function provides the book summary overlay feature. It first identifies the clicked book by id and then uses a for loop to find a matching id in the books object. Once found, the function uses interpolation to generate an html framework that displays the book information and activates the summary overlay.

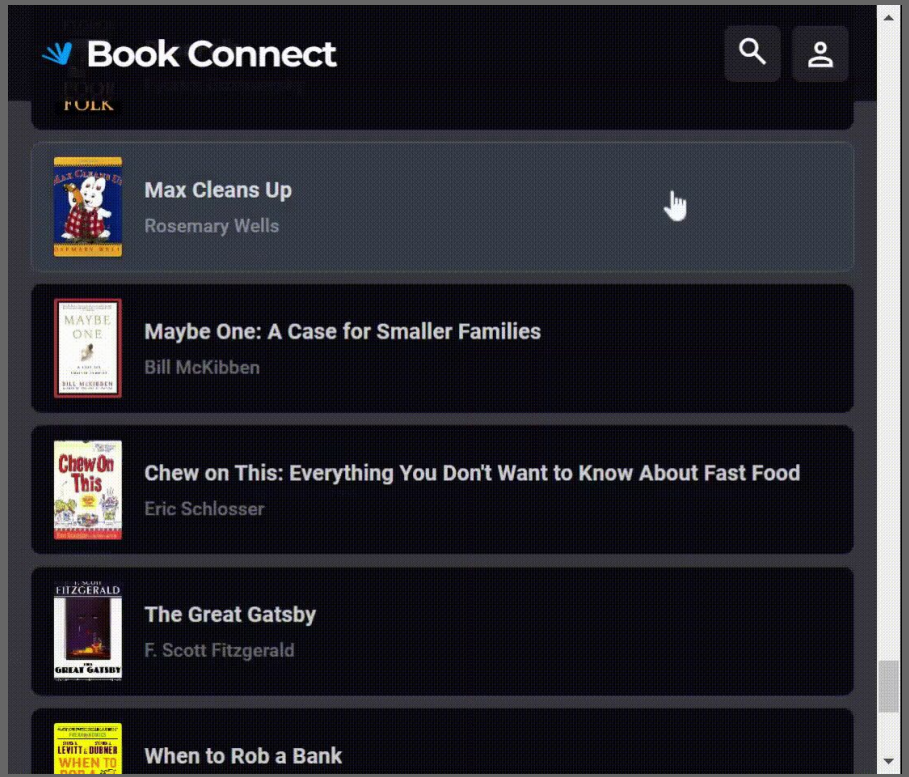
I WANT TO HAVE THE OPTION OF READING THE BOOK SUMMARY



```
 * The dialog box where the overlay will be appended
 */
const bookSummary = document.querySelector('[data-list-active]')
//get the book that is clicked by getting the closest element with the identifier to
//where the click happened.
const book = event.target.closest('.preview');
//get a book id to use to fetch book information
const bookId = book.getAttribute('data-preview');
//for loop to iterate over the book object lloking for matchind ids
for (let i = 0; i < books.length; i++) {
    //check if the id in the books object matches that of the clicked book
    if (books[i].id === bookId) {
    //The book summary overlay html
    bookSummary.innerHTML = /*html*/
    `<div class="overlay__preview">
    <img class="overlay__blur" data-list-blur="" src="${books[i].image}">
    <img class="overlay__image" data-list-image="" src="${books[i].image}">
    </div>
    <div class="overlay__content">
    <h3 class="overlay__title" data-list-title="">${books[i].title} (${new Date(books[i].published).getFullYear()})</h3>
    <div class="overlay__data" data-list-subtitle="">${authors[books[i].author]}</div>
    <p class="overlay__data overlay__data_secondary" data-list-description="">${books[i].description}</p>
    </div>
    <div class="overlay__row">
    <button class="overlay__button overlay__button_primary" data-list-close="">Close</button>
    </div>`
    }
}
    //show the book summary overlay when its done being created
    bookSummary.showModal()
    //when the close button is clicked, the overlay should be removed
```

The **descriptionOverlay** function provides the book summary overlay feature. It first identifies the clicked book by id and then uses a for loop to find a matching id in the books object. Once found, the function uses interpolation to generate an html framework that displays the book information and activates the summary overlay.

I want to be able to search for a book with a partial text input



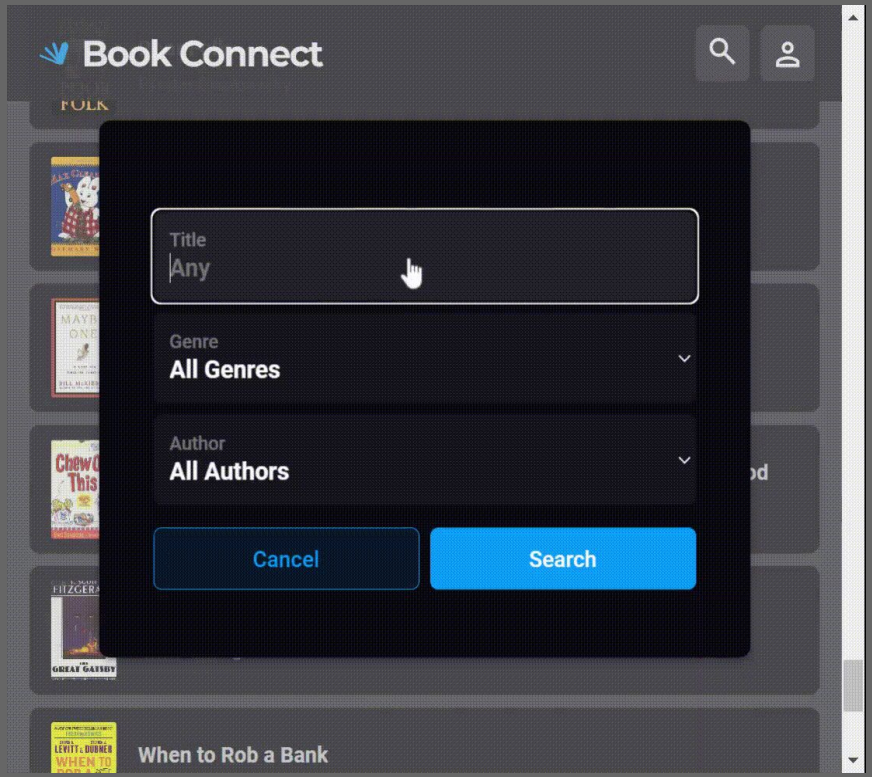
The **searchBooks** function enables searching for books with partial title matches. It fetches input values from the search overlay form, and searches the books array for matches based on text input. Matching books are compiled into an array and used to generate button elements for display on the html page.

I want to be able to search for a book with a partial text input

```
168 export const searchBooks = (event) => {
169     event.preventDefault();
170 
171     const searchText = document.querySelector('[data-search-title]').value.toLowerCase().trim();
172     const selectedGenre = document.querySelector('[data-search-genres]').value;
173     const selectedAuthor = document.querySelector('[data-search-authors]').value;
174 
175     let filteredBooks = books;
176 
177      /* apply the search filter on the genres so that if there is no
178     genre selected, it will not run the code here.
179     The Object.keys() function returns an array of the number ids in the
180     genres object then the find() method will find the first key in the array
181     that has a value that matches the selected Genre.
182     The filter function will look if there is a key called genres in the
183     books array and and return that book */
184     if (selectedGenre !== "All Genres") {
185       const genreId = Object.keys(genres).find(key => genres[key] === selectedGenre);
186       filteredBooks = filteredBooks.filter(book => book.genres.includes(genreId.toString()));
187     }
188 
189     /* apply the search filter on the authors so that if there is no
190     author selected, it will not run the code here. */
191     if (selectedAuthor !== "All Authors") {
192       const authorId = Object.keys(authors).find(key => authors[key] === selectedAuthor);
193       filteredBooks = filteredBooks.filter(book => book.author.includes(authorId));
194     }
195 
196     // Apply the text search filter so that there is no search if no text
197     if (searchText !== "") {
198       filteredBooks = filteredBooks.filter(book => book.title.toLowerCase().includes(searchText));
199     }
```

The **searchBooks** function enables searching for books with partial title matches. It fetches input values from the search overlay form, and searches the books array for matches based on text input. Matching books are compiled into an array and used to generate button elements for display on the html page.

I WANT TO BE ABLE TO FILTER MY SEARCH BY AUTHOR AND/OR GENRE



The **searchBooks** function provides filtering options for author and genre. It fetches the id key for the selected author or genre from the respective objects, and uses it to iterate through the books object and find matches.

The matching books are added to an array called **filteredBooks**. The function then generates button elements for each book in the filteredBooks array, displaying the book's image, title, and author, and enabling summary overlay functionality.

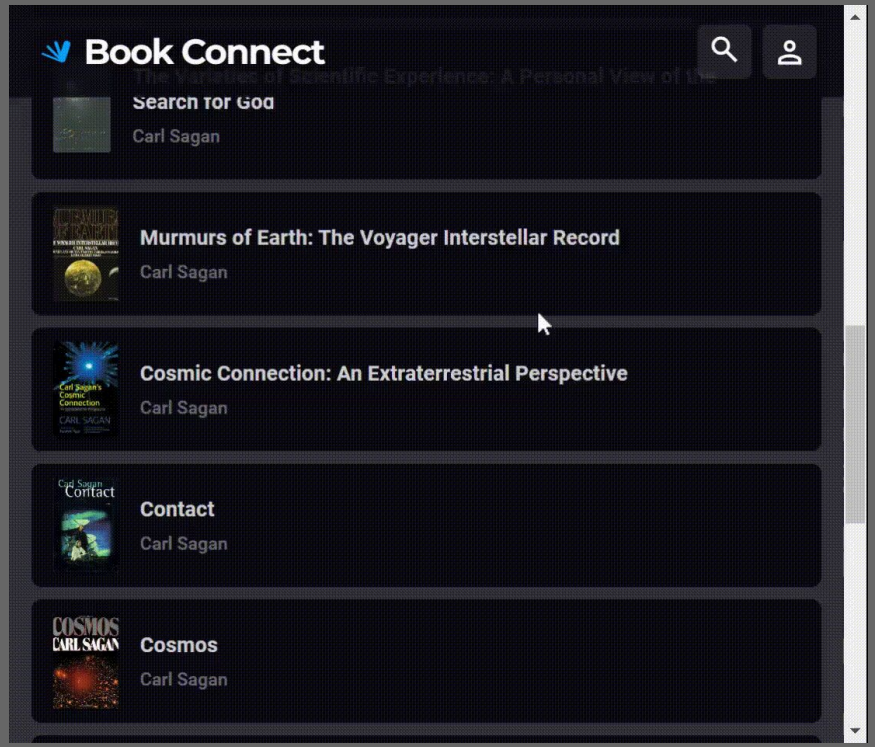
I want to be able to filter my search by author and/or genre

```
200
201     //check if there were any books found
202     let booksFound = filteredBooks.length > 0;
203     //if there are no books found then should print message
204     if (!booksFound) {
205       // Clear the book list on the homepage
206         document.querySelector('[data-list-items]').innerHTML = "";
207       //print this to the page
208         document.querySelector('[data-list-items]').innerHTML = `<div class = "list__message list__message_show" data-list-message = "">
209                                                                  <p>No results found.
210                                                                  Your filters may be too narrow, try again</p>
211                                                                </div>`;
212
213   // disable the show more button for the results page
214     SHOW_MORE_BTN.disabled = true;
215
216       return filteredBooks
217     }
218
219     // Clear the book list on the homepage
220     document.querySelector('[data-list-items]').innerHTML = "";
221
222     // Append the filtered books to the book list, used BOOKs_perpage to show only 36 books per page
223     filteredBooks.slice(0, BOOKS_PER_PAGE).forEach(book => {
224       const button = document.createElement('button');
225       button.classList.add('preview');
226       button.dataset.preview = book.id;
227       button.innerHTML = `
228         <img class="preview__image" src="${book.image}" />
229         <div class="preview__info">
230           <h3 class="preview__title">${book.title}</h3>
```

The **searchBooks** function provides filtering options for author and genre. It fetches the id key for the selected author or genre from the respective objects, and uses it to iterate through the books object and find matches.

The matching books are added to an array called **filteredBooks**. The function then generates button elements for each book in the filteredBooks array, displaying the book's image, title, and author, and enabling summary overlay functionality.

I want to be able to toggle between dark and light so I can use the app comfortably.



- The **changeTheme** function enables switching between day and night themes. It iterates over the array of option values in the drop-down list and uses an if statement to check whether the selected value is "day". If so, it executes a code block that sets the colors to the day theme. Otherwise, the else statement is executed which applies the night theme colors.

Recommendations for better code readability and performance.

- Ensure that the rules of JavaScript syntax are respected.

  Declare variables, commas, semi-colons etc.

- Group code by function

  Variables that are used for specific functions should be placed together with the functions to increase code readability.

- Global variables should be capitalized and placed at the top of the JavaScript code.
- Group DOM elements in objects for easier access.
- Use comments and JSDocs to explain the purpose of functions and variables.

THE END