Book Connect

Table of contents

→ Introduction

→ Code Audit Findings

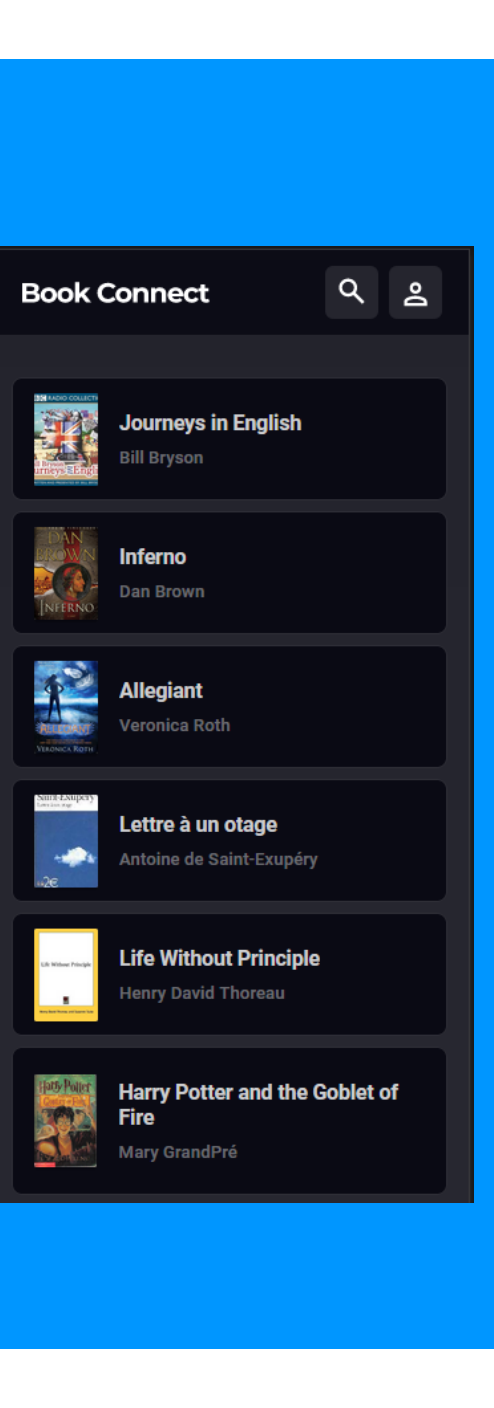
→ Fixes and Improvements

→ Fixing the Initial Problems

→ User Stories

→ Conclusion

Introduction


Book Connect
Journeys in English
Inferno
Allegiant
Lettre à un otage
Life Without Principle
Harry Potter and the Goblet of Fire

→ Given a week to fix the JavaScript code for the app

→ Challenges: Fix Errors, improve performance, readability and maintainability

→ Goals: Prioritize completing user stories for the app

Code Audit Findings

→ Improper variable declarations

→ Incorrect Syntax

→ Lack of proper error handling

→ Improper DOM manipulations (data-list-item)

→ Lack of Imports

→ No DRY Principle used (Don't Repeat Yourself))

Fixes & Improvements

→ Fixed improper variable declarations and it's name (let & const)

→ Implemented proper error handling & performance increases (Guard Clauses in for loops using continue keyword)

→ Created an object named html to store all the DOM elements

→ Added JSDocs for documentation for improved Readability (/** */)

→ Improved Maintainability with modules (type="module")

→ Followed DRY Principle (Reusable code)

Fixing the Initial Problems

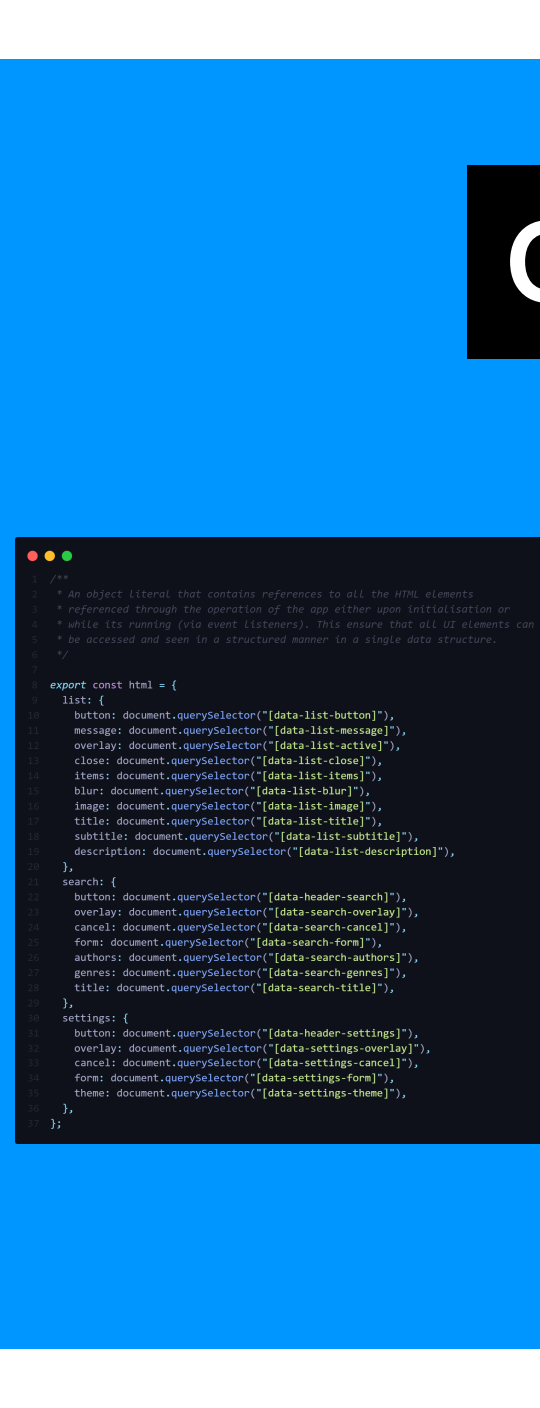
→ Fix script tag (type="module" defer)

→ Add an object named html to reference all elements

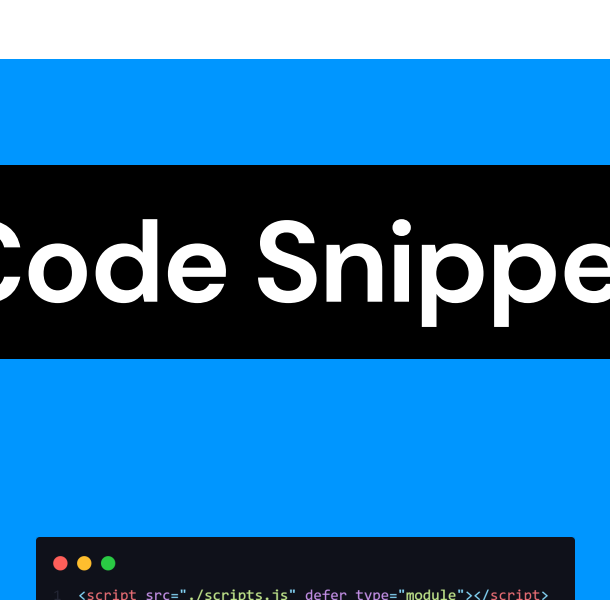
→ Importing correctly

→ Implement event Listeners for all open/close buttons

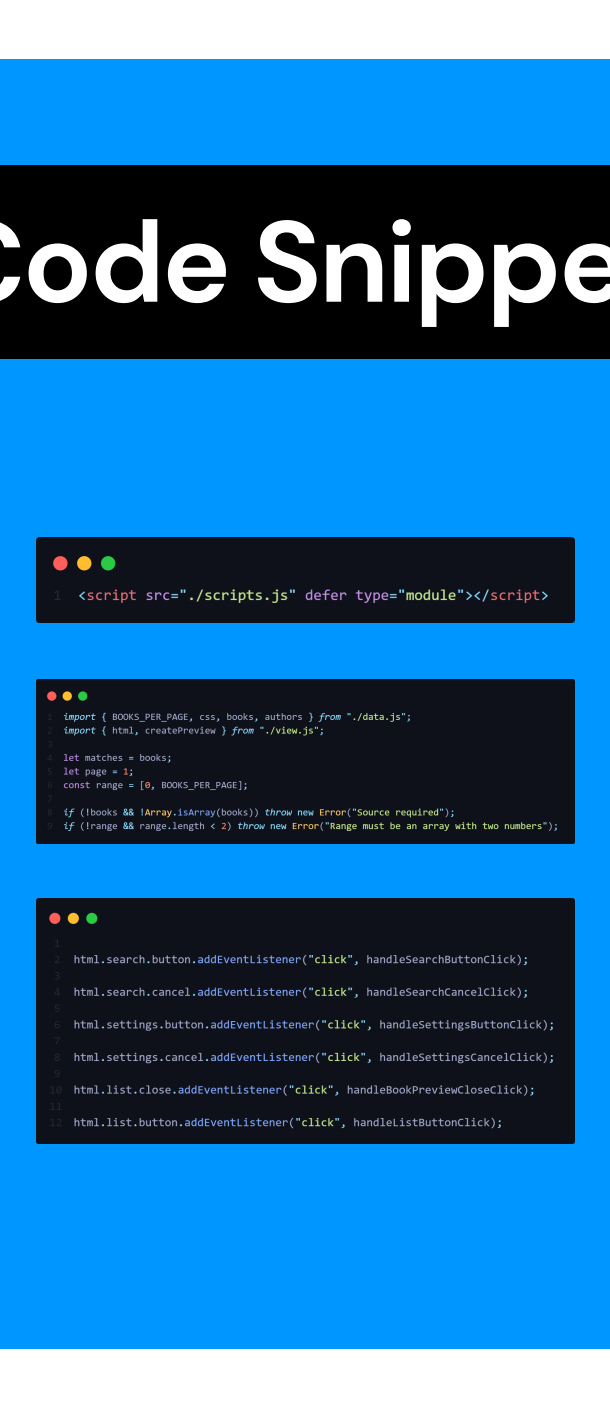
Code Snippets



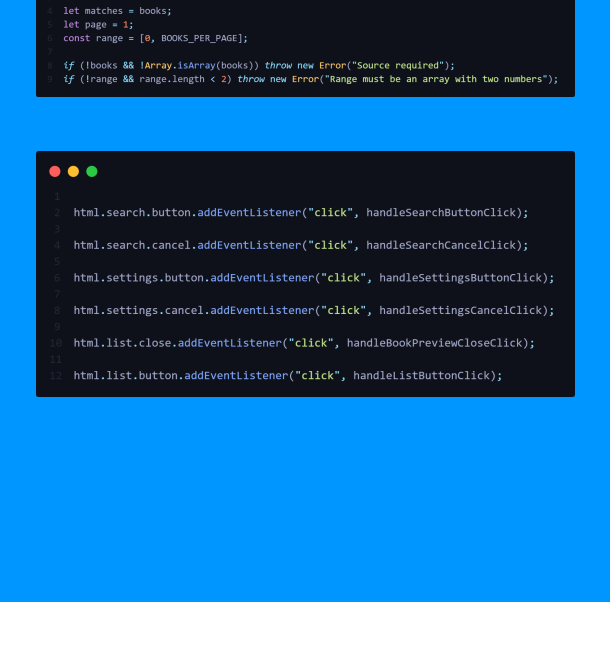
```
/**
 * An object literal that contains references to all the HTML elements
 * referenced through the operation of the app either upon initialisation or
 * while its running (via event listeners). This ensure that all UI elements can
 * be accessed and seen in a structured manner in a single data structure.
 */

export const html = {
  list: {
    button: document.querySelector("[data-list-button]"),
    message: document.querySelector("[data-list-message]"),
    overlay: document.querySelector("[data-list-active]"),
    close: document.querySelector("[data-list-close]"),
    items: document.querySelector("[data-list-items]"),
    blur: document.querySelector("[data-list-blur]"),
    image: document.querySelector("[data-list-image]"),
    title: document.querySelector("[data-list-title]"),
    subtitle: document.querySelector("[data-list-subtitle]"),
    description: document.querySelector("[data-list-description]"),
  },
  search: {
    button: document.querySelector("[data-header-search]"),
    overlay: document.querySelector("[data-search-overlay]"),
    cancel: document.querySelector("[data-search-cancel]"),
    form: document.querySelector("[data-search-form]"),
    authors: document.querySelector("[data-search-authors]"),
    genres: document.querySelector("[data-search-genres]"),
    title: document.querySelector("[data-search-title]"),
  },
  settings: {
    button: document.querySelector("[data-header-settings]"),
    overlay: document.querySelector("[data-settings-overlay]"),
    cancel: document.querySelector("[data-settings-cancel]"),
    form: document.querySelector("[data-settings-form]"),
    theme: document.querySelector("[data-settings-theme]"),
  },
};
```



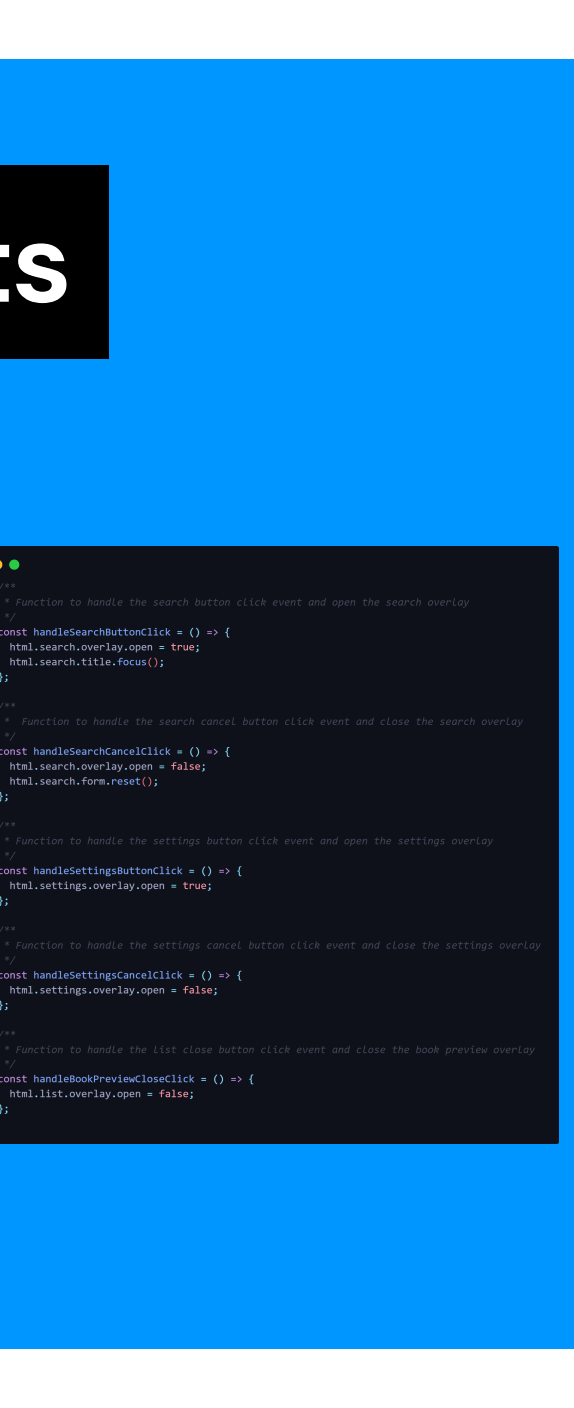
```
<script src="./scripts.js" defer type="module"></script>
```



```
import { BOOKS_PER_PAGE, css, books, authors } from "./data.js";
import { html, createPreview } from "./view.js";

let matches = books;
let page = 1;
const range = [0, BOOKS_PER_PAGE];

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");
```



```
html.search.button.addEventListener("click", handleSearchButtonClick);

html.search.cancel.addEventListener("click", handleSearchCancelClick);

html.settings.button.addEventListener("click", handleSettingsButtonClick);

html.settings.cancel.addEventListener("click", handleSettingsCancelClick);

html.list.close.addEventListener("click", handleBookPreviewCloseClick);

html.list.button.addEventListener("click", handleListButtonClick);
```



```
/**
 * Function to handle the search button click event and open the search overlay
 */
const handleSearchButtonClick = () => {
  html.search.overlay.open = true;
  html.search.title.focus();
};

/**
 *  Function to handle the search cancel button click event and close the search overlay
 */
const handleSearchCancelClick = () => {
  html.search.overlay.open = false;
  html.search.form.reset();
};

/**
 * Function to handle the settings button click event and open the settings overlay
 */
const handleSettingsButtonClick = () => {
  html.settings.overlay.open = true;
};

/**
 * Function to handle the settings cancel button click event and close the settings overlay
 */
const handleSettingsCancelClick = () => {
  html.settings.overlay.open = false;
};

/**
 * Function to handle the list close button click event and close the book preview overlay
 */
const handleBookPreviewCloseClick = () => {
  html.list.overlay.open = false;
};
```

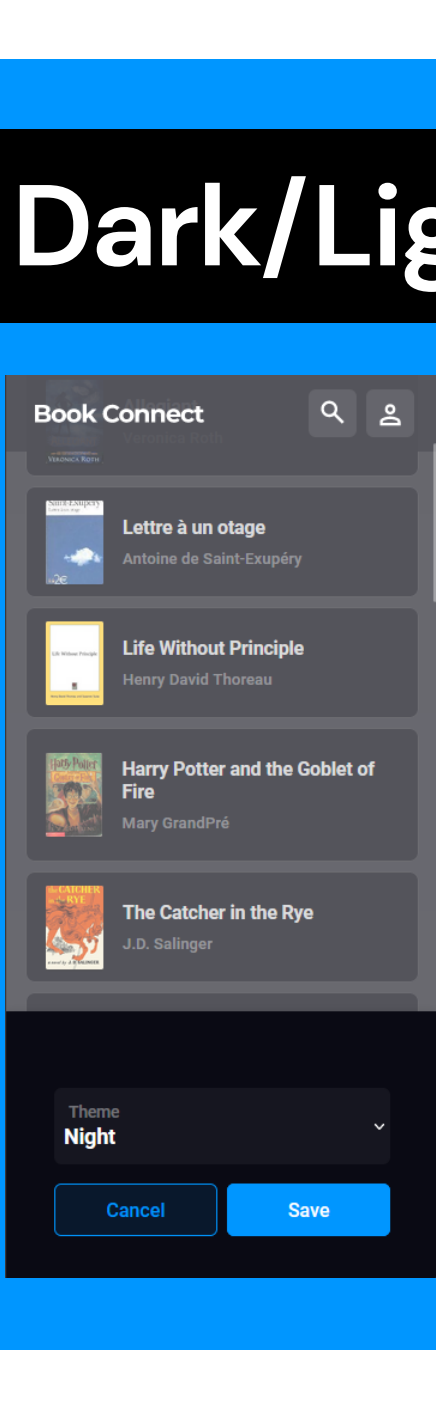
User Stories

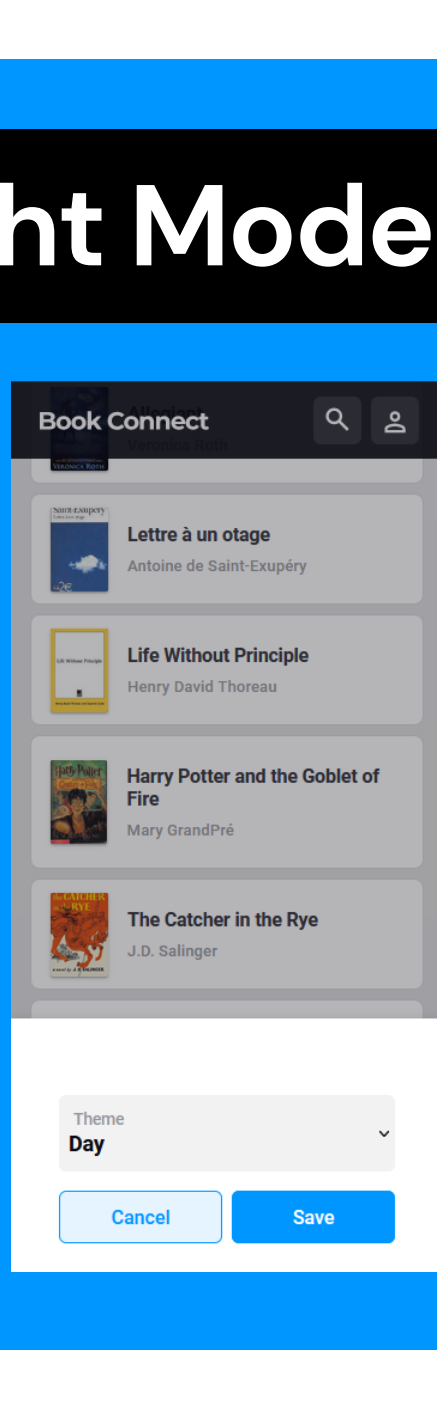
→ Show all the features I worked on

→ Provide images of the feature and it's corresponding code

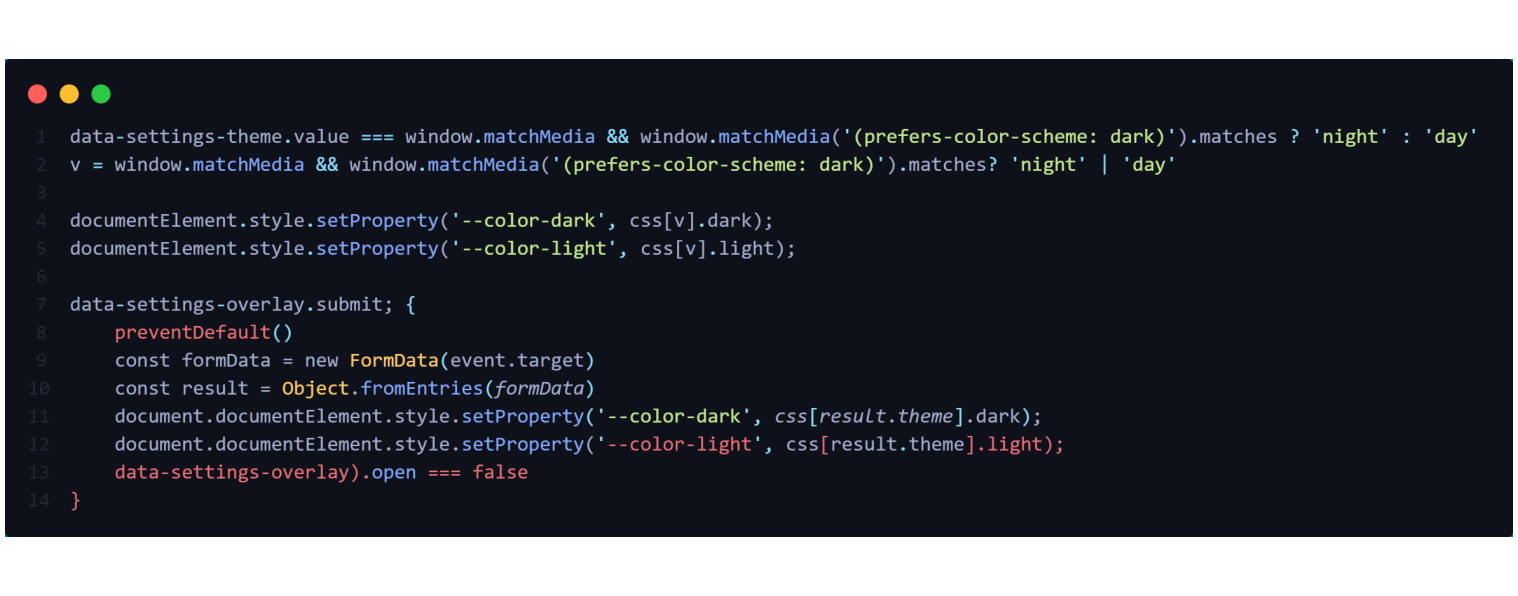
→ Show initial code vs current code

Dark/Light Mode


Book Connect
Lettre à un otage
Life Without Principle
Harry Potter and the Goblet of Fire
The Catcher in the Rye
Theme
Night
Cancel
Save


Book Connect
Lettre à un otage
Life Without Principle
Harry Potter and the Goblet of Fire
The Catcher in the Rye
Theme
Day
Cancel
Save

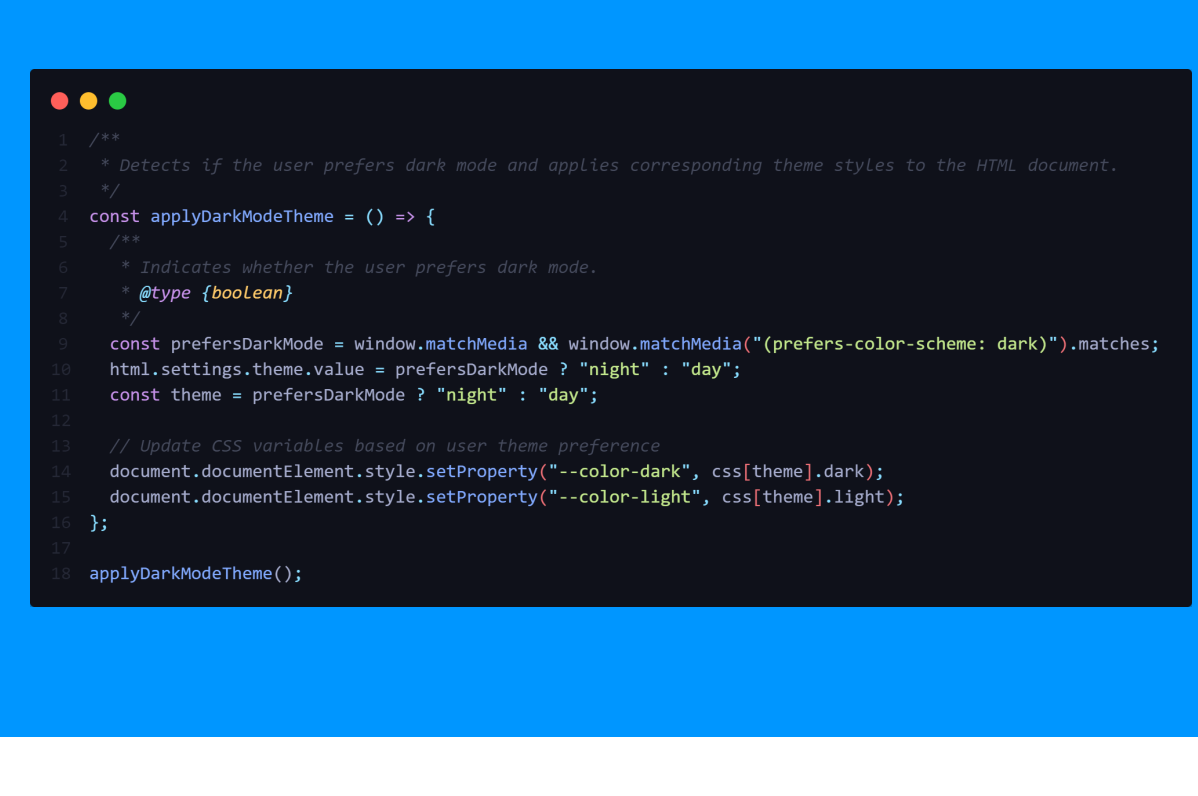
Old Code



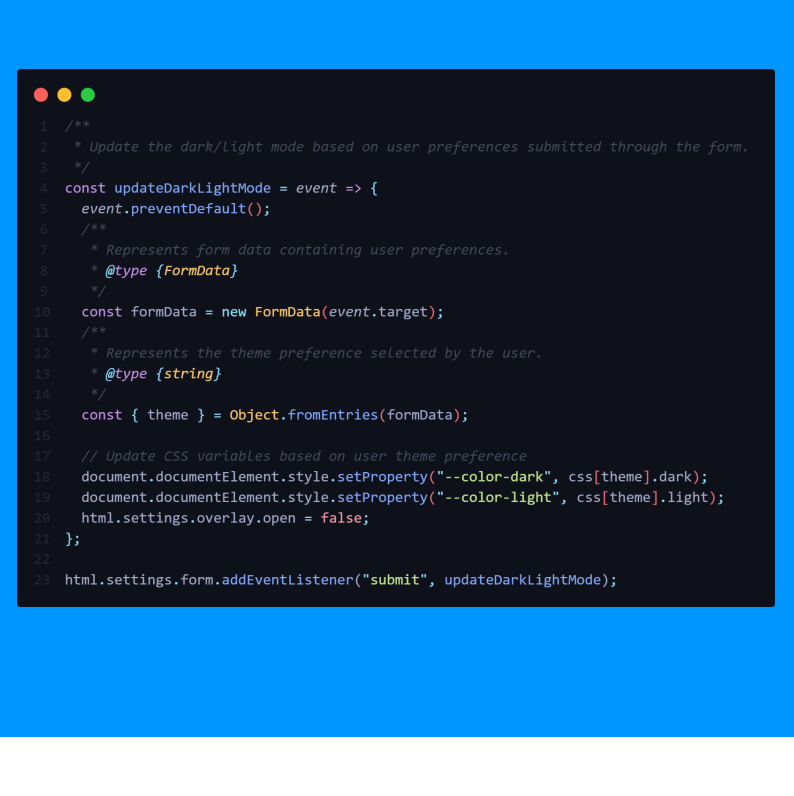
```
data-settings-theme.value === window.matchMedia && window.matchMedia('(prefers-color-scheme: dark)').matches ? 'night' : 'day'
v = window.matchMedia && window.matchMedia('(prefers-color-scheme: dark)').matches? 'night' | 'day'

documentElement.style.setProperty('--color-dark', css[v].dark);
documentElement.style.setProperty('--color-light', css[v].light);

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

New Code

When app is opened it detects if user prefers Dark mode and then sets it to dark theme and second function updates theme based on users submitted preference.

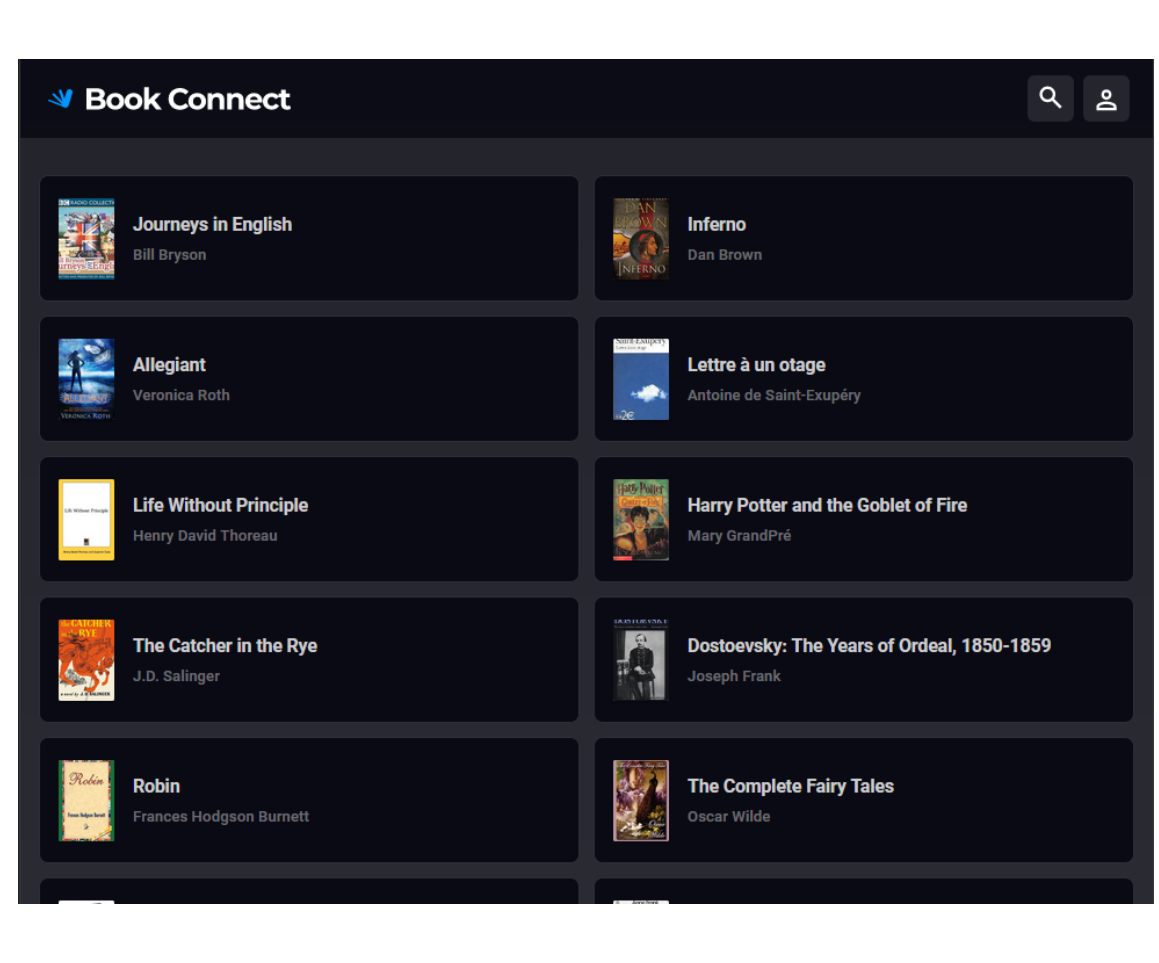


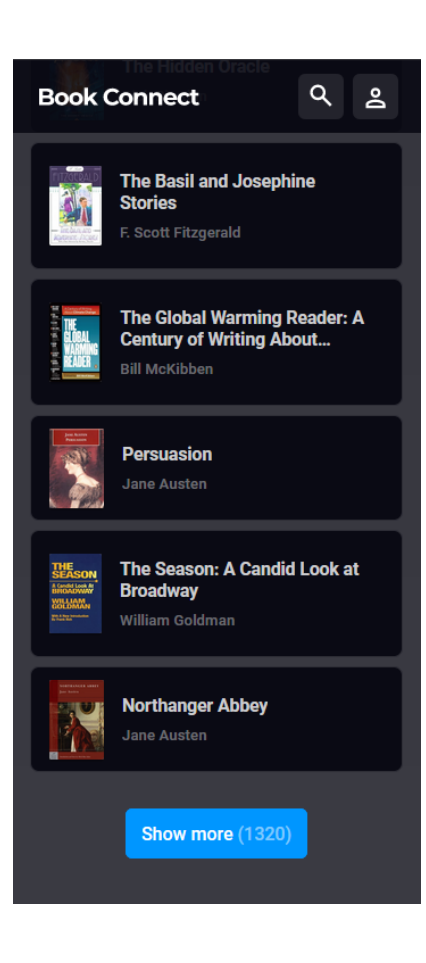
```
/**
 * Detects if the user prefers dark mode and applies corresponding theme styles to the HTML document.
 */
const applyDarkModeTheme = () => {
  /**
   * Indicates whether the user prefers dark mode.
   * @type {boolean}
   */
  const prefersDarkMode = window.matchMedia && window.matchMedia("(prefers-color-scheme: dark)").matches;
  html.settings.theme.value = prefersDarkMode ? "night" : "day";
  const theme = prefersDarkMode ? "night" : "day";

  // Update CSS variables based on user theme preference
  document.documentElement.style.setProperty("--color-dark", css[theme].dark);
  document.documentElement.style.setProperty("--color-light", css[theme].light);
};

applyDarkModeTheme();
```



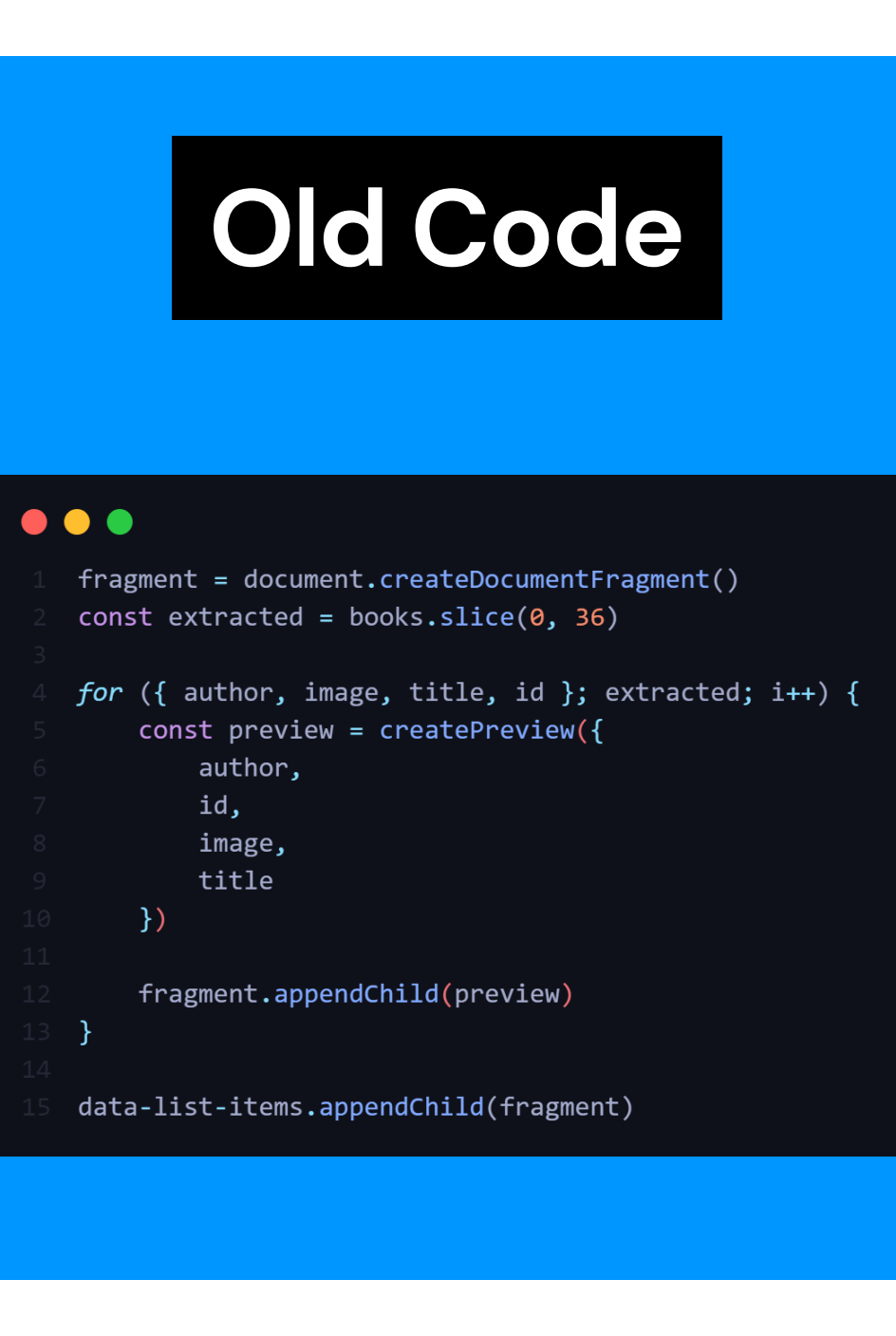
```
/**
 * Update the dark/light mode based on user preferences submitted through the form.
 */
const updateDarkLightMode = event => {
  event.preventDefault();
  /**
   * Represents form data containing user preferences.
   * @type {FormData}
   */
  const formData = new FormData(event.target);
  /**
   * Represents the theme preference selected by the user.
   * @type {string}
   */
  const { theme } = Object.fromEntries(formData);

  // Update CSS variables based on user theme preference
  document.documentElement.style.setProperty("--color-dark", css[theme].dark);
  document.documentElement.style.setProperty("--color-light", css[theme].light);
  html.settings.overlay.open = false;
};

html.settings.form.addEventListener("submit", updateDarkLightMode);
```

Create Book List


Book Connect
Journeys in English
Inferno
Allegiant
Lettre à un otage
Life Without Principle
Harry Potter and the Goblet of Fire
The Catcher in the Rye
Dostoevsky: The Years of Ordeal, 1850-1859
Robin
The Complete Fairy Tales


Book Connect
The Basil and Josephine Stories
The Global Warming Reader: A Century of Writing About...
Persuasion
The Season: A Candid Look at Broadway
Northanger Abbey
Show more

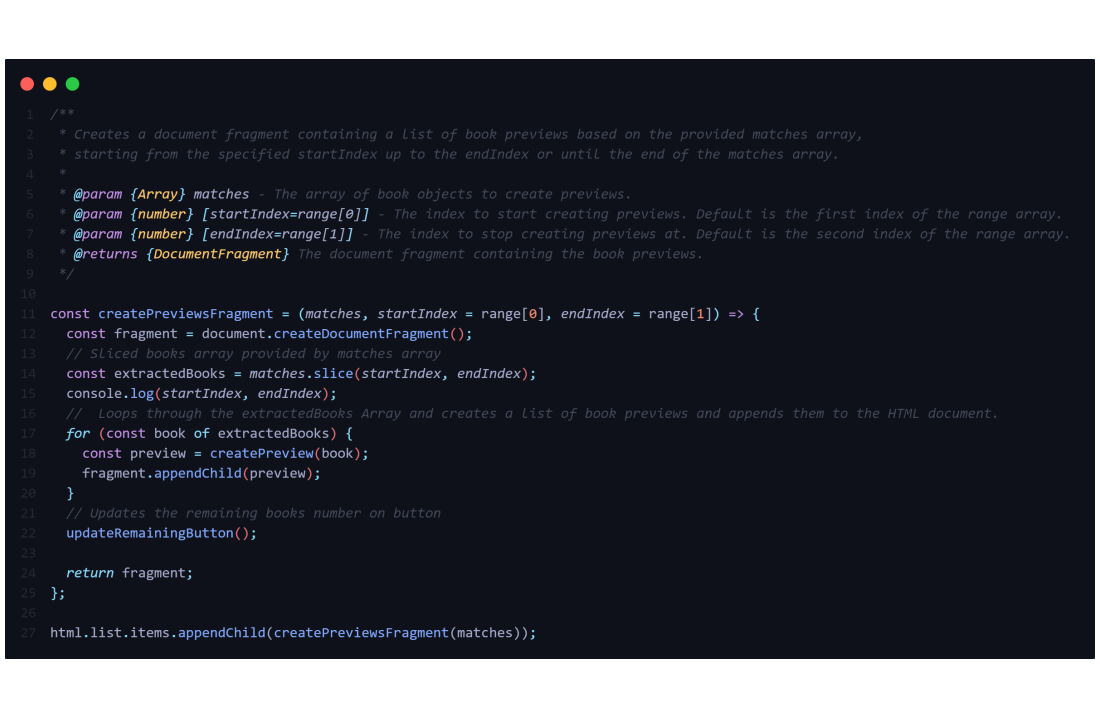
Old Code



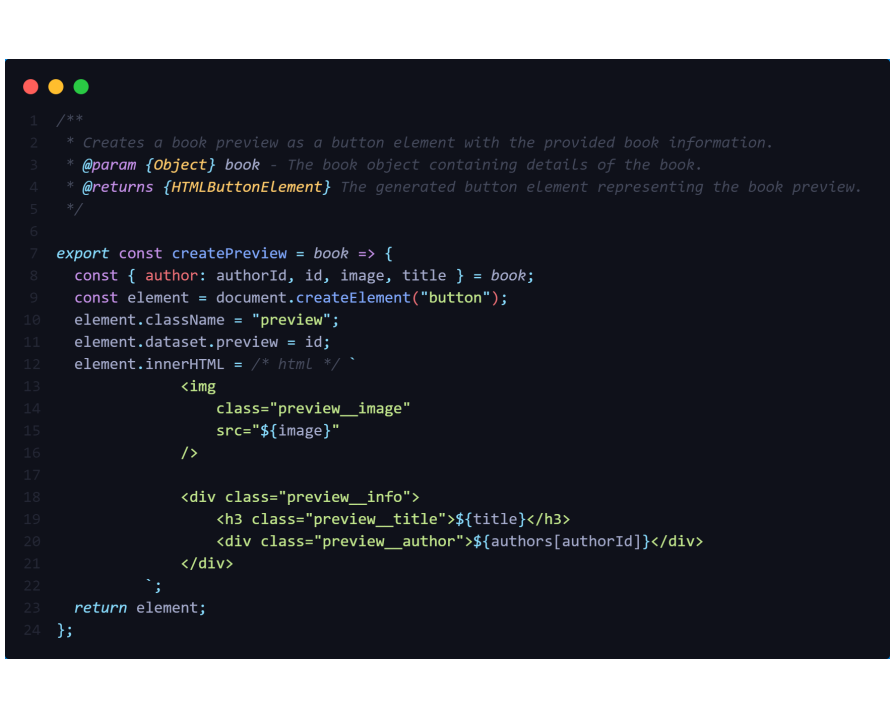
```
fragment = document.createDocumentFragment()
const extracted = books.slice(0, 36)

for ({ author, image, title, id }; extracted; i++) {
    const preview = createPreview({
        author,
        id,
        image,
        title
    })

    fragment.appendChild(preview)
}

data-list-items.appendChild(fragment)
```

New Code

Reusable function that creates a book list that's displays on the page

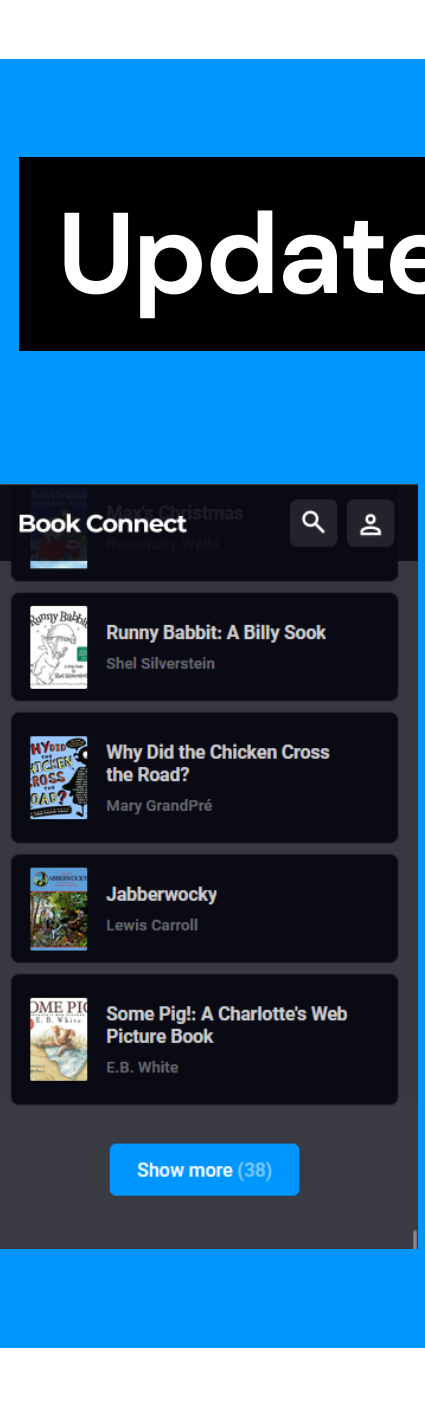


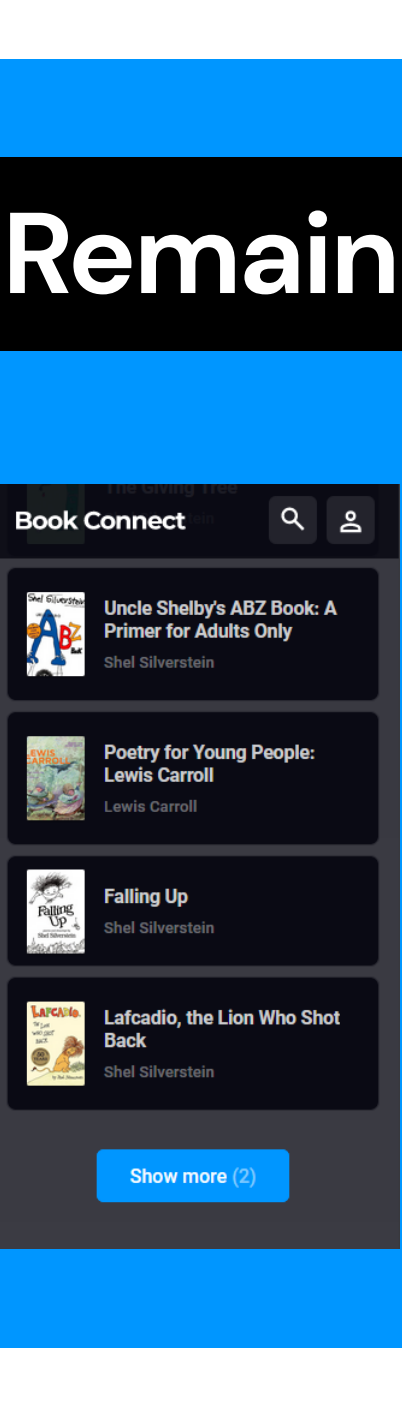
```
/**
 * Creates a document fragment containing a list of book previews based on the provided matches array,
 * starting from the specified startIndex up to the endIndex or until the end of the matches array.
 *
 * @param {Array} matches - The array of book objects to create previews.
 * @param {number} [startIndex=range[0]] - The index to start creating previews. Default is the first index of the range array.
 * @param {number} [endIndex=range[1]] - The index to stop creating previews at. Default is the second index of the range array.
 * @returns {DocumentFragment} The document fragment containing the book previews.
 */

const createPreviewsFragment = (matches, startIndex = range[0], endIndex = range[1]) => {
  const fragment = document.createDocumentFragment();
  // Sliced books array provided by matches array
  const extractedBooks = matches.slice(startIndex, endIndex);
  console.log(startIndex, endIndex);
  //  Loops through the extractedBooks Array and creates a list of book previews and appends them to the HTML document.
  for (const book of extractedBooks) {
    const preview = createPreview(book);
    fragment.appendChild(preview);
  }
  // Updates the remaining books number on button
  updateRemainingButton();

  return fragment;
};

html.list.items.appendChild(createPreviewsFragment(matches));
```

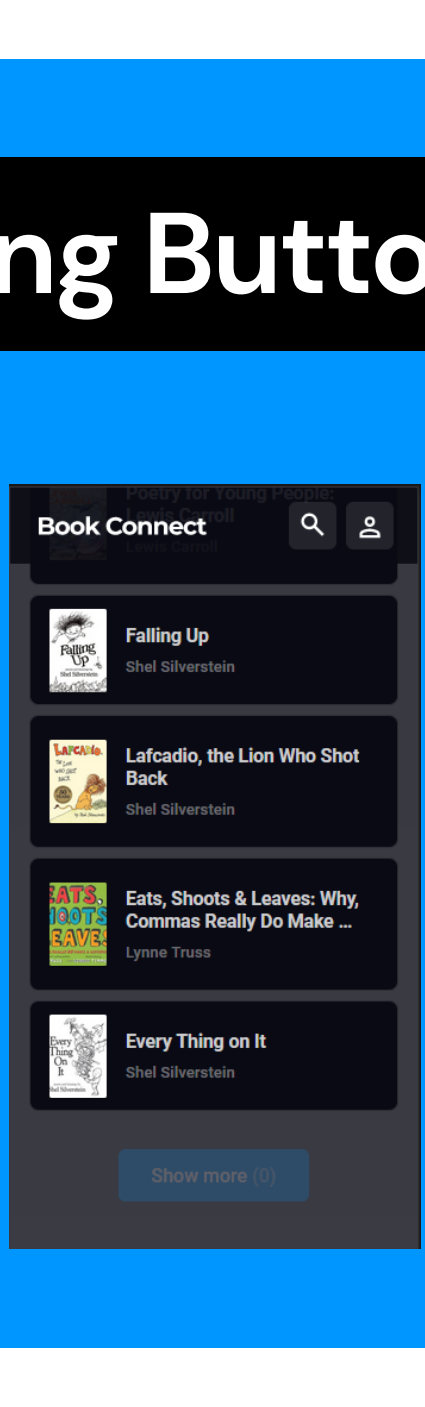


```
/**
 * Creates a book preview as a button element with the provided book information.
 * @param {Object} book - The book object containing details of the book.
 * @returns {HTMLButtonElement} The generated button element representing the book preview.
 */

export const createPreview = book => {
  const { author: authorId, id, image, title } = book;
  const element = document.createElement("button");
  element.className = "preview";
  element.dataset.preview = id;
  element.innerHTML = /* html */ `
            <img
                class="preview__image"
                src="${image}"
            />

            <div class="preview__info">
                <h3 class="preview__title">${title}</h3>
                <div class="preview__author">${authors[authorId]}</div>
            </div>
        `;
  return element;
};
```

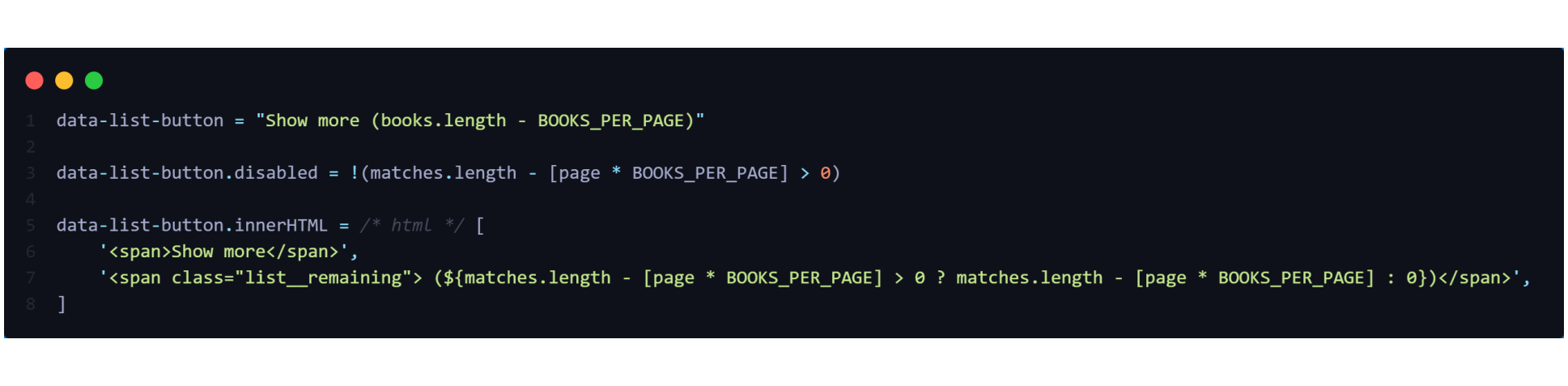
Update Remaining Button


Book Connect
Runny Babbit: A Billy Sook
Why Did the Chicken Cross the Road?
Jabberwocky
Some Pig!: A Charlotte's Web Picture Book
Show more (38)


Book Connect
Uncle Shelby's ABZ Book: A Primer for Adults Only
Poetry for Young People: Lewis Carroll
Falling Up
Lafcadio, the Lion Who Shot Back
Show more (2)


Book Connect
Falling Up
Lafcadio, the Lion Who Shot Back
Eats, Shoots & Leaves: Why, Commas Really Do Make ...
Every Thing on It
Show more (0)

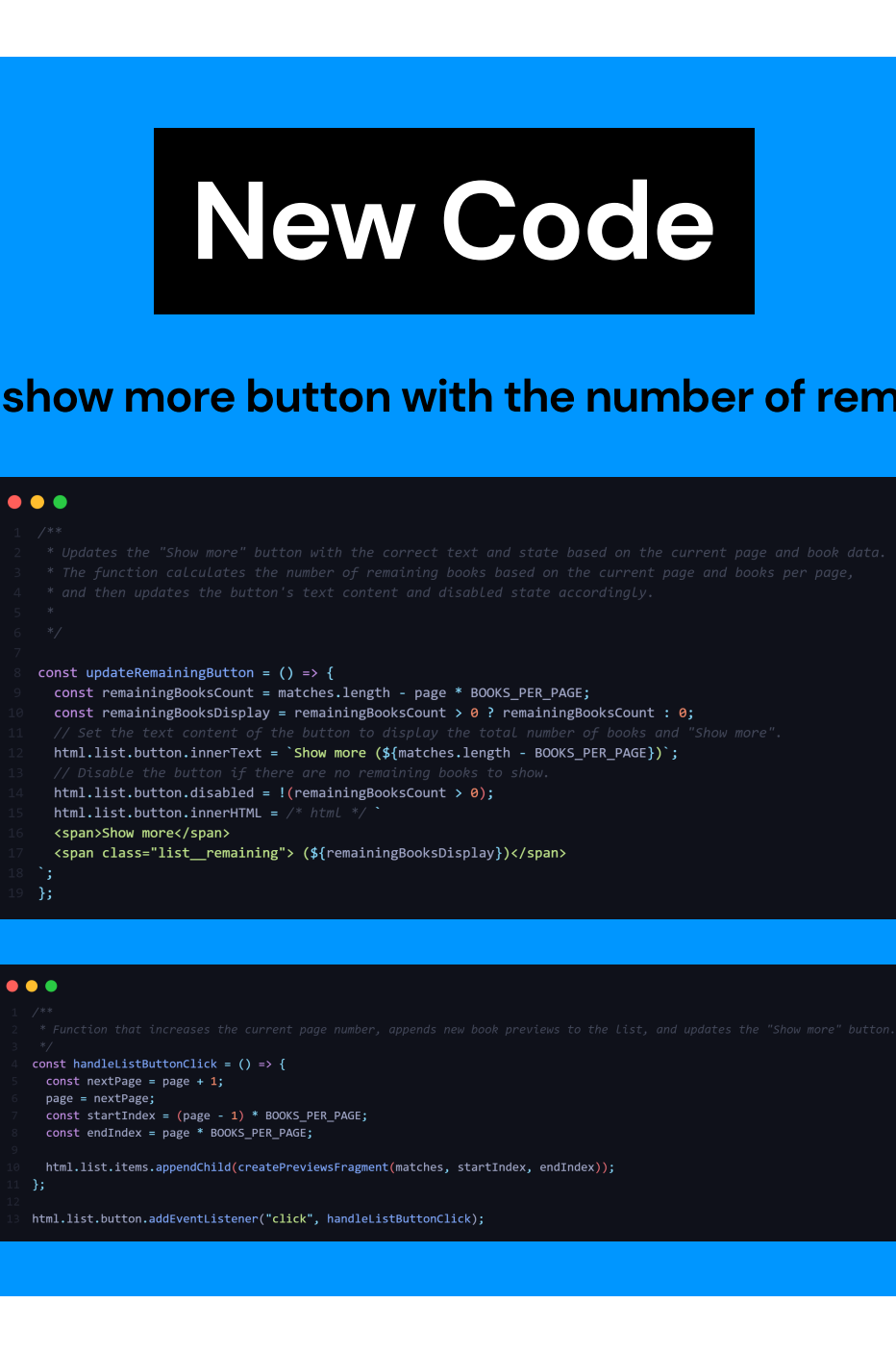
Old Code



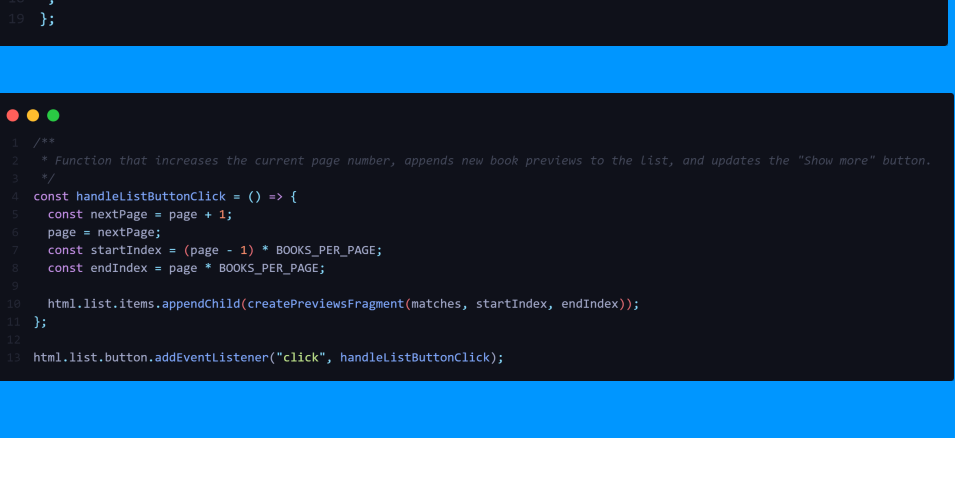
```
data-list-button = "Show more (books.length - BOOKS_PER_PAGE)"

data-list-button.disabled = !(matches.length - [page * BOOKS_PER_PAGE] > 0)

data-list-button.innerHTML = /* html */ [
    '<span>Show more</span>',
    '<span class="list__remaining"> (${matches.length - [page * BOOKS_PER_PAGE] > 0 ? matches.length - [page * BOOKS_PER_PAGE] : 0})</span>',
]
```

New Code

Updates the show more button with the number of remaining books.

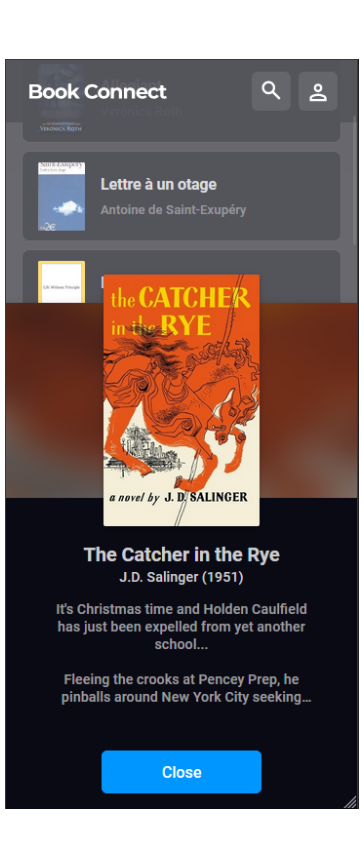


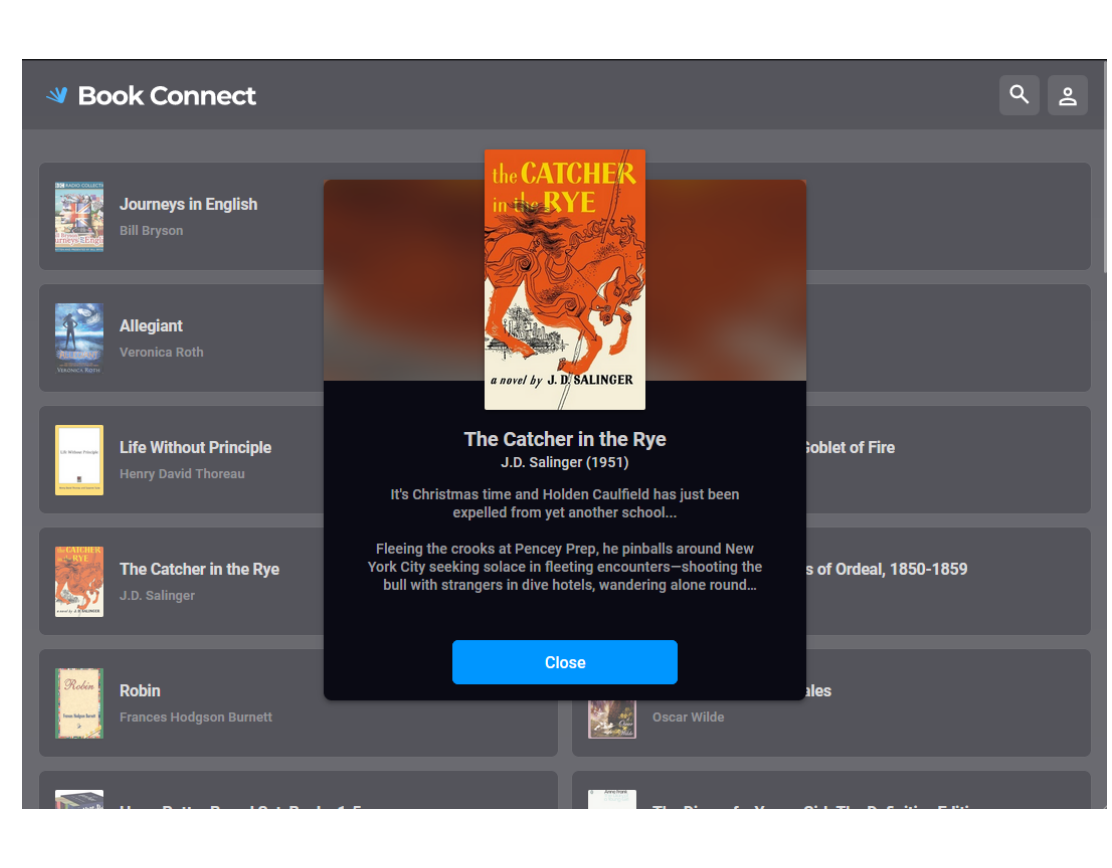
```
/**
 * Updates the "Show more" button with the correct text and state based on the current page and book data.
 * The function calculates the number of remaining books based on the current page and books per page,
 * and then updates the button's text content and disabled state accordingly.
 *
 */

const updateRemainingButton = () => {
  const remainingBooksCount = matches.length - page * BOOKS_PER_PAGE;
  const remainingBooksDisplay = remainingBooksCount > 0 ? remainingBooksCount : 0;
  // Set the text content of the button to display the total number of books and "Show more".
  html.list.button.innerText = `Show more (${matches.length - BOOKS_PER_PAGE})`;
  // Disable the button if there are no remaining books to show.
  html.list.button.disabled = !(remainingBooksCount > 0);
  html.list.button.innerHTML = /* html */ `
  <span>Show more</span>
  <span class="list__remaining"> (${remainingBooksDisplay})</span>
`;
};
```



```
/**
 * Function that increases the current page number, appends new book previews to the list, and updates the "Show more" button.
 */
const handleListButtonClick = () => {
  const nextPage = page + 1;
  page = nextPage;
  const startIndex = (page - 1) * BOOKS_PER_PAGE;
  const endIndex = page * BOOKS_PER_PAGE;

  html.list.items.appendChild(createPreviewsFragment(matches, startIndex, endIndex));
};

html.list.button.addEventListener("click", handleListButtonClick);
```

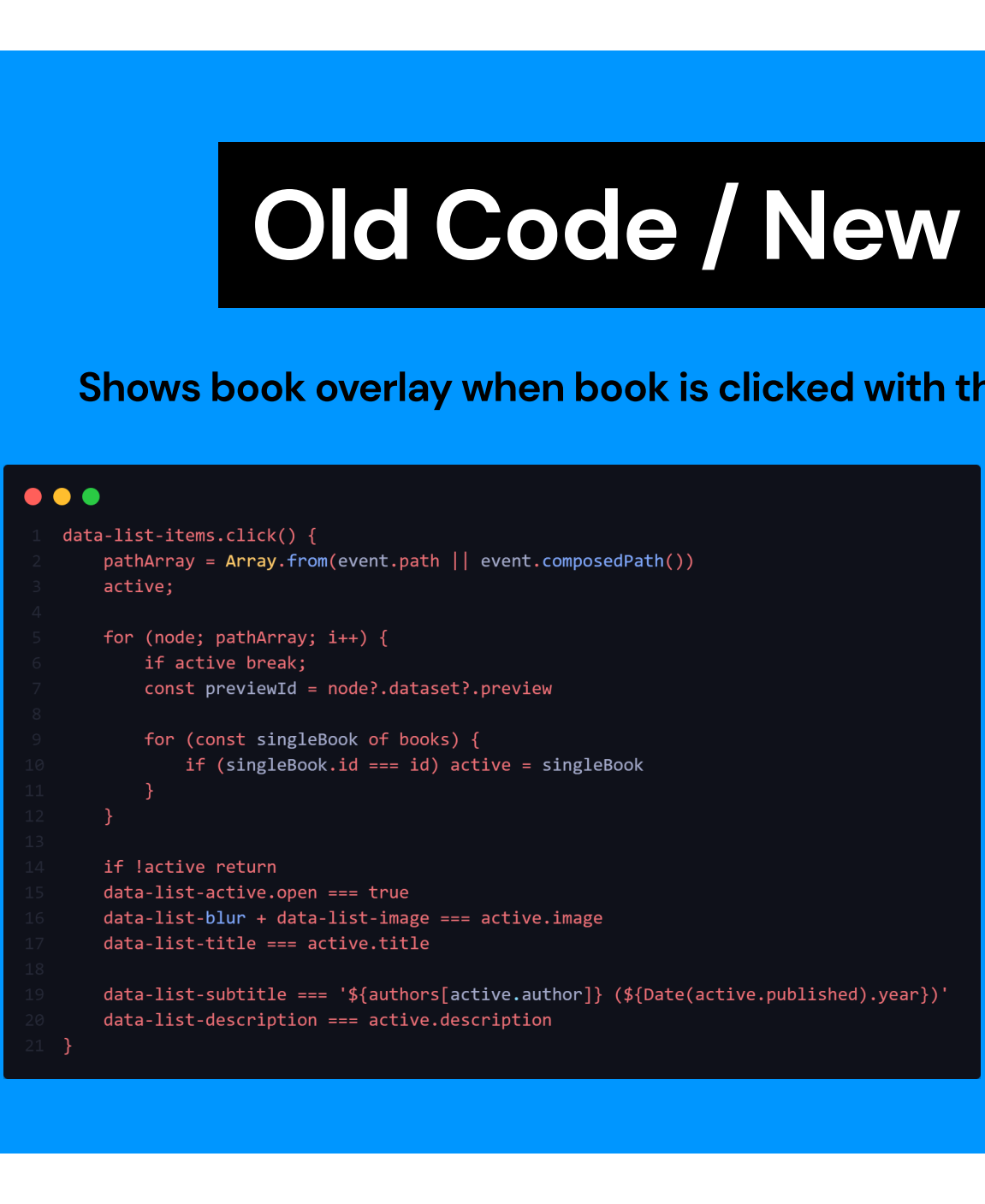
Book Preview Overlay


Book Connect
Lettre à un otage
The Catcher in the Rye
J.D. Salinger (1951)
It's Christmas time and Holden Caulfield has just been expelled from yet another school...
Fleeing the crooks at Pencey Prep, he pinballs around New York City seeking...
Close

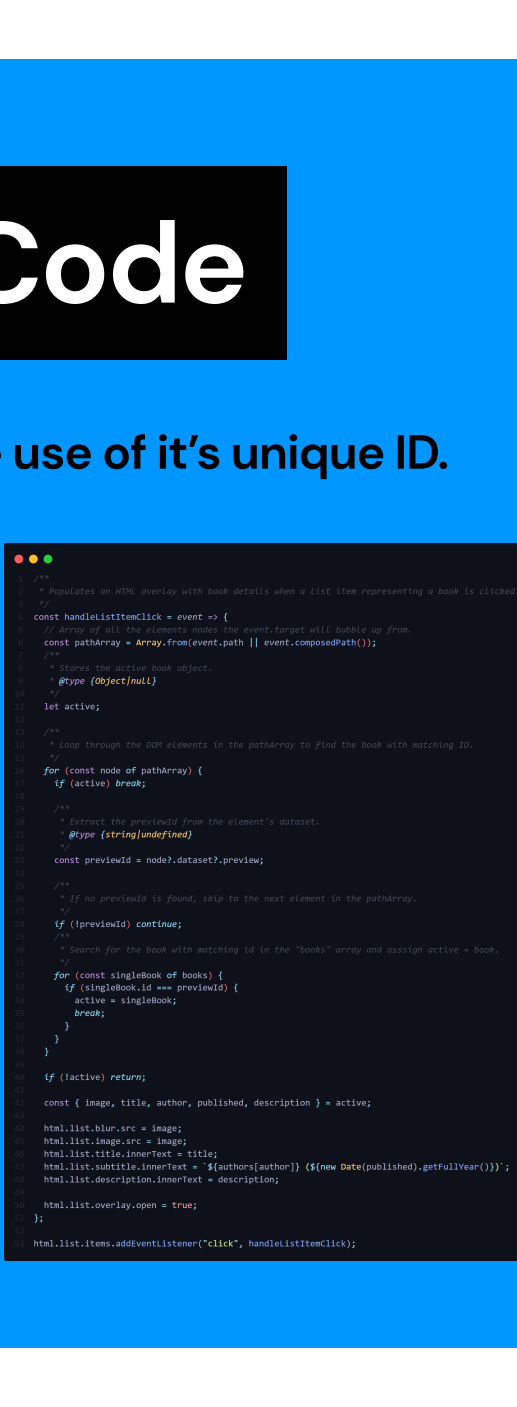

Book Connect
Journeys in English
Allegiant
Life Without Principle
The Catcher in the Rye
Robin
The Catcher in the Rye
J.D. Salinger (1951)
It's Christmas time and Holden Caulfield has just been expelled from yet another school...
Fleeing the crooks at Pencey Prep, he pinballs around New York City seeking solace in fleeting encounters—shooting the bull with strangers in dive hotels, wandering alone round...
Close

Old Code / New Code

Shows book overlay when book is clicked with the use of it's unique ID.

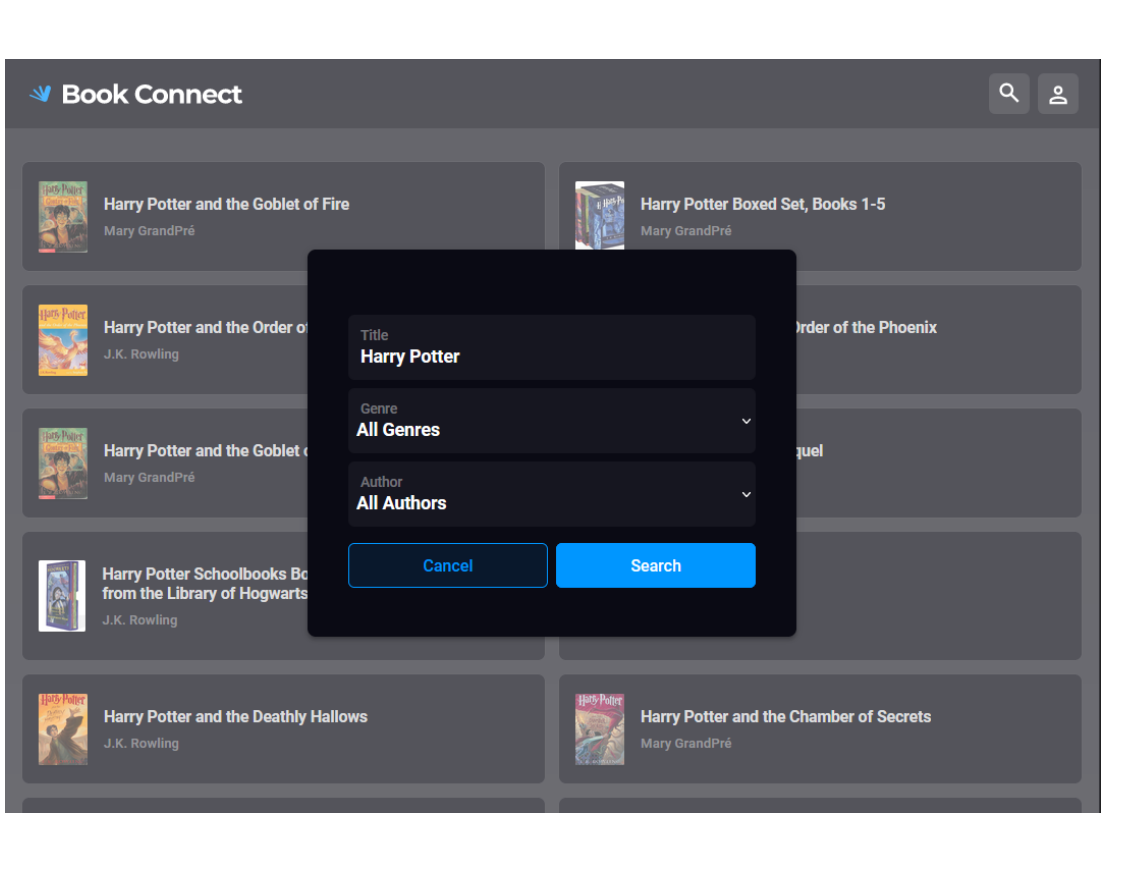


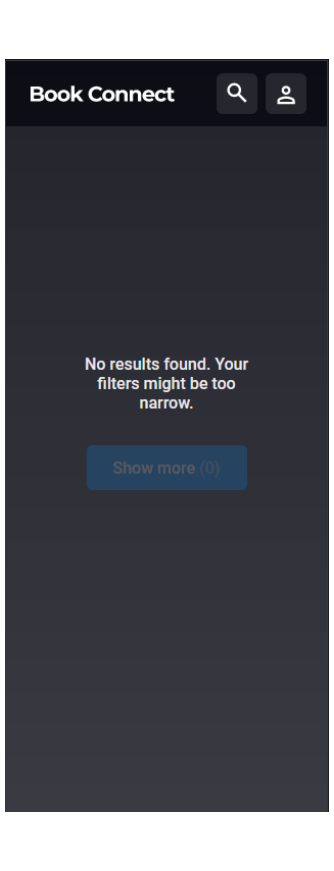
```
data-list-items.click() {
    pathArray = Array.from(event.path || event.composedPath())
    active;

    for (node; pathArray; i++) {
        if active break;
        const previewId = node?.dataset?.preview

        for (const singleBook of books) {
            if (singleBook.id === id) active = singleBook
        }
    }

    if !active return
    data-list-active.open === true
    data-list-blur + data-list-image === active.image
    data-list-title === active.title

    data-list-subtitle === '${authors[active.author]} (${Date(active.published).year})'
    data-list-description === active.description
}
```



```
/**
 * Populates an HTML overlay with book details when a list item representing a book is clicked.
 */
const handleListItemClick = event => {
  // Array of all the elements nodes the event.target will bubble up from.
  const pathArray = Array.from(event.path || event.composedPath());
  /**
   * Stores the active book object.
   * @type {Object|null}
   */
  let active;

  /**
   * Loop through the DOM elements in the pathArray to find the book with matching ID.
   */
  for (const node of pathArray) {
    if (active) break;

    /**
     * Extract the previewId from the element's dataset.
     * @type {string|undefined}
     */
    const previewId = node?.dataset?.preview;

    /**
     * If no previewId is found, skip to the next element in the pathArray.
     */
    if (!previewId) continue;
    /**
     * Search for the book with matching id in the "books" array and assign active = book.
     */
    for (const singleBook of books) {
      if (singleBook.id === previewId) {
        active = singleBook;
        break;
      }
    }
  }

  if (!active) return;

  const { image, title, author, published, description } = active;

  html.list.blur.src = image;
  html.list.image.src = image;
  html.list.title.innerText = title;
  html.list.subtitle.innerText = `${authors[author]} (${new Date(published).getFullYear()})`;
  html.list.description.innerText = description;

  html.list.overlay.open = true;
};

html.list.items.addEventListener("click", handleListItemClick);
```

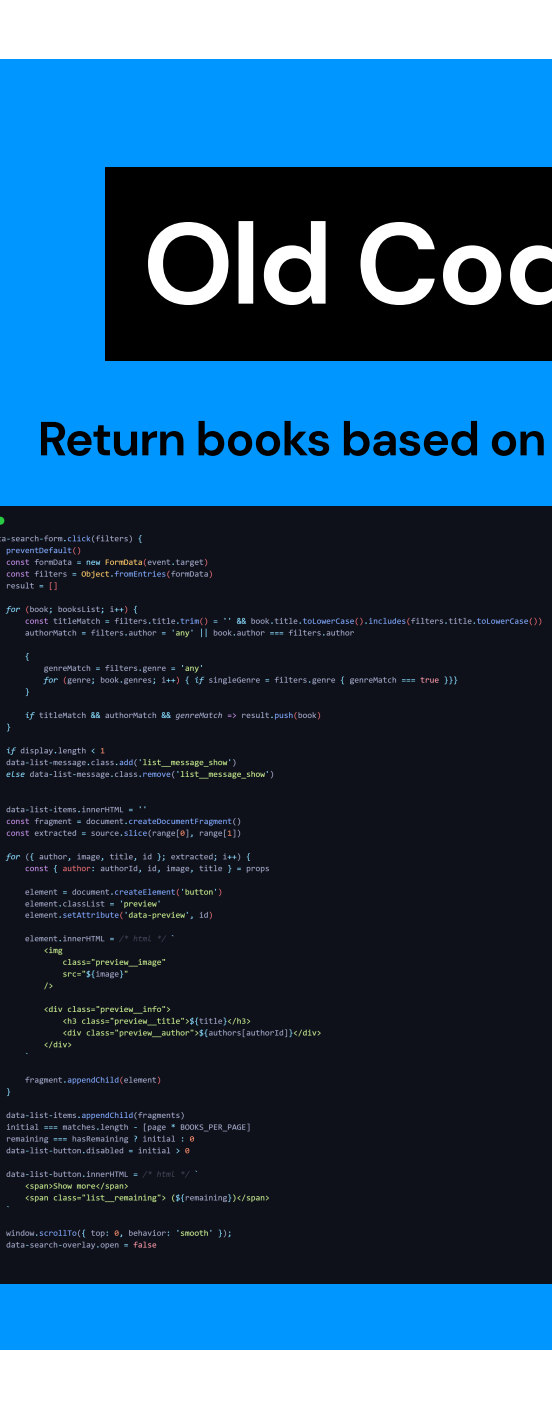
Book Filter Form

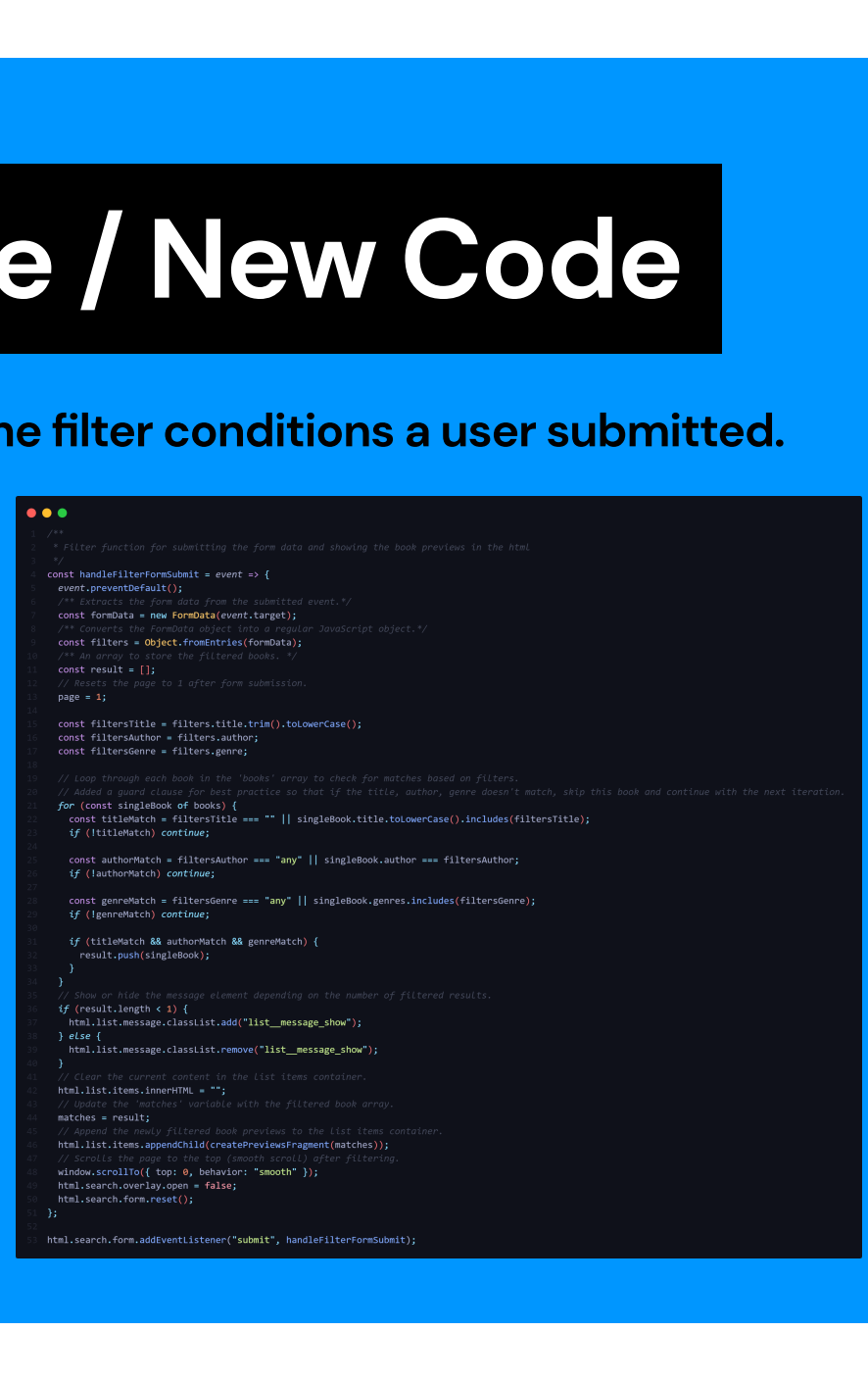

Book Connect
Harry Potter and the Goblet of Fire
Harry Potter Boxed Set, Books 1-5
Harry Potter and the Order of the Phoenix
Harry Potter Schoolbooks Box Set: Two Classic Books from the Library of Hogwarts
Harry Potter and the Deathly Hallows
Harry Potter and the Chamber of Secrets
Title
Harry Potter
Genre
All Genres
Author
All Authors
Cancel
Search


Book Connect
No results found. Your filters might be too narrow.
Show more (0)

Old Code / New Code

Return books based on the filter conditions a user submitted.





```
/**
 * Filter function for submitting the form data and showing the book previews in the html
 */
const handleFilterFormSubmit = event => {
  event.preventDefault();
  /** Extracts the form data from the submitted event.*/
  const formData = new FormData(event.target);
  /** Converts the FormData object into a regular JavaScript object.*/
  const filters = Object.fromEntries(formData);
  /** An array to store the filtered books. */
  const result = [];
  // Resets the page to 1 after form submission.
  page = 1;

  const filtersTitle = filters.title.trim().toLowerCase();
  const filtersAuthor = filters.author;
  const filtersGenre = filters.genre;

  // Loop through each book in the 'books' array to check for matches based on filters.
  // Added a guard clause for best practice so that if the title, author, genre doesn't match, skip this book and continue with the next iteration.
  for (const singleBook of books) {
    const titleMatch = filtersTitle === "" || singleBook.title.toLowerCase().includes(filtersTitle);
    if (!titleMatch) continue;

    const authorMatch = filtersAuthor === "any" || singleBook.author === filtersAuthor;
    if (!authorMatch) continue;

    const genreMatch = filtersGenre === "any" || singleBook.genres.includes(filtersGenre);
    if (!genreMatch) continue;

    if (titleMatch && authorMatch && genreMatch) {
      result.push(singleBook);
    }
  }
  // Show or hide the message element depending on the number of filtered results.
  if (result.length < 1) {
    html.list.message.classList.add("list__message_show");
  } else {
    html.list.message.classList.remove("list__message_show");
  }
  // Clear the current content in the list items container.
  html.list.items.innerHTML = "";
  // Update the 'matches' variable with the filtered book array.
  matches = result;
  // Append the newly filtered book previews to the list items container.
  html.list.items.appendChild(createPreviewsFragment(matches));
  // Scrolls the page to the top (smooth scroll) after filtering.
  window.scrollTo({ top: 0, behavior: "smooth" });
  html.search.overlay.open = false;
  html.search.form.reset();
};

html.search.form.addEventListener("submit", handleFilterFormSubmit);
```

Conclusion

→ Most difficult feature to implement was the Filter Form (complexity)

→ I improved my ability to troubleshoot and improve existing code.

→ I improved my debugging and problem solving skills

THANK YOU