Book Connect

SUMMOH081_FTC2301_GROUPB_SumaiyaMohamed_IWA19

Main changes I made throughout the code include:

1. `{BOOKS_PER_PAGE, authors, genres , books}`

// Imported the BOOKS_PER_PAGE, authors and genres objects as well as the books array from the data.js file

So that they can be accessed in the scripts.js file.

2. `matches = books;`

`page = 1;`

// Declared variables properly throughout my code with const to prevent scope problems.

3. `<script src="./scripts.js"></script>`

`<script src="./data.js"></script>`

// Added the type=”module” attribute to the scripts tags to allow the importing and exporting of variables

// Also added the defer attribute so that the javascript parses after the html has parsed.

```
116     data-list-items.innerHTML = ''
117     const fragment = document.createDocumentFragment()
118     const extracted = source.slice(range[0], range[1])
119
120     for ({ author, image, title, id }; extracted; i++) {
121         const { author: authorId, id, image, title } = props
122
123         element = document.createElement('button')
124         element.classList = 'preview'
125         element.setAttribute('data-preview', id)
126
127         element.innerHTML = /* html */ `
128             <img
129                 class="preview__image"
130                 src="${image}"
131             />
132
133             <div class="preview__info">
134                 <h3 class="preview__title">${title}</h3>
135                 <div class="preview__author">${authors[authorId]}</div>
136             </div>
137         `
138
139         fragment.appendChild(element)
140     }
141
142     data-list-items.appendChild(fragments)
```

This code handles the displaying of the books in the browser (original code).

Changes I made include:

// Sliced the books array that was imported from the data.js file from index 0 to index 35 and is being referenced through a variable named matches.

// Added a function that handles the loop that creates the button element for each book and gives it a class=" preview" as well as a data-preview attribute and changes the innerHtml for each book to its corresponding image, title and author then appends the element to the fragment that was created and appends that fragment to the div element that all the books will appear under.

// Added an eventlistener that fires the function and loads the books when the browser window is opened and also added an eventlistener that tells the browser that the element was added (this code was written for the book overlay that shows the books information when you click on it.

```
const bookList = document.querySelector('[data-list-items]');
const fragment = document.createDocumentFragment();
let currentPage = 1; // add currentPage variable to keep track of the current page


const createBookOverlay = (booksToRender) => { // modify function to receive a parameter of books to render
  for (let i = 0; i < booksToRender.length; i++) {
    const { author: authorId, id, image, title } = booksToRender[i];

    const element = document.createElement('button');
    element.className = 'preview';
    element.setAttribute('data-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>
    `;

    fragment.appendChild(element);


    // Add a click event listener to the button element
    element.addEventListener('click', (event) => {
      const button = event.target;
      console.log(button);
    });
  }
bookList.appendChild(fragment);
};
```

My code with all the changes.

```
    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
}
```

This part of the code enables the user to load more books when the show more button is clicked on

Changes I made include :
// I retrieved the button from the html using its data-list-button attribute then added a event listener to it , so that when its clicked more buttons will load.

// In the function the eventlistener is paresing i declared a variable that gets the number of books that are already loaded which are 36 ,then function adds 36 books every time the button is clicked and a if statement that handles whether the button is usable or not depending on the number of books there are left to add.

```
});

// Code that handles displaying more books.
const loadMoreButton = document.querySelector('[data-list-button]');

loadMoreButton.addEventListener('click', () => {

  const startIndex = currentPage * BOOKS_PER_PAGE; // get the starting index of the books to render
  const endIndex = startIndex + BOOKS_PER_PAGE; // get the ending index of the books to render
  const booksToRender = matches.slice(startIndex, endIndex); // get the books to render from the matches array

  createBookOverlay(booksToRender); // render the books

  currentPage++; // increment the current page


  if (booksToRender.length < BOOKS_PER_PAGE) { // if there are less books than `BOOKS_PER_PAGE`, hide the load more button
    loadMoreButton.style.display = 'none';
  } else {
    loadMoreButton.innerHTML = `Show more (${matches.length - endIndex})`;
  }
});
```

My code with all changes.

This code handles the book overlay and the books information display when the book is clicked on:

Changes I made include :

// Retrieved the book overlay(dialog) and close button from the html using their corresponding data attributes.

// Added an event listener to the close button as well to the booklist.

// Added a for loop that handles the displaying of the book that is selected with its corresponding information (background Image, Image , author , published date and description)

```
const formData = new FormData(event.target)
const filters = Object.fromEntries(formData)
result = []

for (book; booksList; i++) {
    titleMatch = filters.title.trim() = '' && book.title.toLowerCase().includes[filters.title.toLowerCase()]
    authorMatch = filters.author = 'any' || book.author === filters.author

    {
        genreMatch = filters.genre = 'any'
        for (genre; book.genres; i++) { if singleGenre = filters.genre { genreMatch === true }}}
    }

    if titleMatch && authorMatch && genreMatch => result.push(book)
}

if display.length < 1
data-list-message.class.add('list__message_show')
else data-list-message.class.remove('list__message_show')


data-list-items.innerHTML = ''
const fragment = document.createDocumentFragment()
const extracted = source.slice(range[0], range[1])
```

My code with changes part 1

```
// Code for the book summary.
const bookSummaryClose = document.querySelector('[data-list-close]');
const bookDialog = document.querySelector('[data-list-active]');

bookSummaryClose.addEventListener('click', () => {
  bookDialog.close();
});

bookList.addEventListener('click', (event) => {
  let active = null;
  const pathArray = Array.from(event.path || event.composedPath());

  for (let i = 0; i < pathArray.length; i++) {
    const node = pathArray[i];

    if (node?.dataset?.preview) {
      const previewId = node.dataset.preview;

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

  if (!active) {
    return;
  }

  bookDialog.setAttribute('open', true);
```

My code with changes part 2

```
  bookDialog.setAttribute('open', true);

  const bookBlur = document.querySelector('[data-list-blur]');
  const bookImage = document.querySelector('[data-list-image]');
  const bookTitle = document.querySelector('[data-list-title]');
  const bookSubtitle = document.querySelector('[data-list-subtitle]');
  const bookDescription = document.querySelector('[data-list-description]');

  bookBlur.style.backgroundImage = `url(${active.image})`;
  bookImage.src = active.image;
  bookTitle.textContent = active.title;
  bookSubtitle.textContent = `${authors[active.author]} (${new Date(active.published).getFullYear()})`;
  bookDescription.textContent = active.description;
});
```

This code handles the settings form where the user can select their preferred theme

Changes I made include:

// Fetched the submit and close button from the html using their data attributes.

// Changed the day and night object property values to background and color.

// Made two separate functions that handles the option for day and the option for night.

//Made a function named changeThemeColor that fires the function for day or the function for night under a condition.

// Added an eventlistener that listens if there is a change in the select field and then fires changeThemeColor.

//Added a variable with an if statement that stores the selected theme in the localstorage so that the selected theme can be applied to the website.

```
62
63  data-settings-theme.value === window.matchMedia && window.matchMedia('(prefers-color-scheme: dark)').matches ? 'night' : 'day'
64  v = window.matchMedia && window.matchMedia('(prefers-color-scheme: dark)').matches? 'night' | 'day'
65
66  documentElement.style.setProperty('--color-dark', css[v].dark);
67  documentElement.style.setProperty('--color-light', css[v].light);
68  data-list-button = "Show more (books.length - BOOKS_PER_PAGE)"
69
70  data-list-button.disabled = !(matches.length - [page * BOOKS_PER_PAGE] > 0)
71
72  data-list-button.innerHTML = /* html */ [
73      '<span>Show more</span>',
74      '<span class="list__remaining"> (${matches.length - [page * BOOKS_PER_PAGE] > 0 ? matches.length - [page * BOOKS_PER_PAGE] : 0})</span>',
75  ]
76
```

```
// The code below deals with the day and night options and store them.
const day = {
  backgroundColor: '#FFFFFF',
  color: '#333333'
};

const night = {
  backgroundColor:'#212121' ,
  color: '#F0F0F0'
};

const settingsButton = document.querySelector('[data-header-settings]');
const settingsDialog = document.querySelector('[data-settings-overlay]');
const settingsForm = document.querySelector('[data-settings-form]');
const themeSelect = document.querySelector('[data-settings-theme]');
const settingsSubmit = document.querySelector('[data-settings-submit]')
const settingsCancel = document.querySelector('[data-settings-cancel]');

const applyDayTheme = () => {
  document.body.style.backgroundColor = day.backgroundColor;
  document.body.style.color = day.color;
  localStorage.setItem('theme', 'day');
};

const applyNightTheme = () => {
  document.body.style.backgroundColor = night.backgroundColor;
  document.body.style.color = night.color;
  localStorage.setItem('theme', 'night');
};
```

My code with all the changes partl

```
const applyNightTheme = () => {
  document.body.style.backgroundColor = night.backgroundColor;
  document.body.style.color = night.color;
  localStorage.setItem('theme', 'night');
};

const changeThemeColor = (selectedValue) => {
  if (selectedValue === 'night') {
    applyNightTheme();
  } else {
    applyDayTheme();
  }
};

// Listen for changes to the theme select dropdown
if (themeSelect) {
  themeSelect.addEventListener('change', () => {
    const selectedValue = themeSelect.value;
    changeThemeColor(selectedValue);
  });
}

// Retrieve the theme preference from local storage (if available)
const savedTheme = localStorage.getItem('theme');
if (savedTheme) {
  if (savedTheme === 'night') {
    applyNightTheme();
  } else {
    applyDayTheme();
  }
}
```

My code with all the changes part2

This is the code that deals with the search form of authors and genres

Changes I made:

// Retrieved the close and submit buttons from the html using their data attributes.

// Created a variable that makes a new form and a variable that stores the form entries

// Created a empty array and a loop that loops over the books object .

// Created a option element for the genres then appended it to the select field where the genres will appear.

// Used a loop to retrieve all the genres names in the genres object so that they can appear as options inside the select field.

// Created a option element for authors then appended it to the select field where the authors will appear.

// Used a loop to retrieve all the authors names in the authors object so that they can appear as options inside the select field.

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

genres = document.createDocumentFragment()
element = document.createElement('option')
element.value = 'any'
element = 'All Genres'
genres.appendChild(element)

for ([id, name]; Object.entries(genres); i++) {
    document.createElement('option')
    element.value = value
    element.innerText = text
    genres.appendChild(element)
}

data-search-genres.appendChild(genres)

authors = document.createDocumentFragment()
element = document.createElement('option')
element.value = 'any'
element.innerText = 'All Authors'
authors.appendChild(element)

for ([id, name];Object.entries(authors); id++) {
```

```
    genres.appendChild(element)
}

data-search-genres.appendChild(genres)

authors = document.createDocumentFragment()
element = document.createElement('option')
element.value = 'any'
element.innerText = 'All Authors'
authors.appendChild(element)

for ([id, name];Object.entries(authors); id++) {
    document.createElement('option')
    element.value = value
    element = text
    authors.appendChild(element)
}

data-search-authors.appendChild(authors)
```

```
// Code for the search form.
const searchButton = document.querySelector('[data-header-search]')
searchButton.addEventListener('click', () => {
  const searchOverlay = document.querySelector('[data-search-overlay]')
  searchOverlay.setAttribute('open', true)
})

const searchForm = document.querySelector('[data-search-form]')
searchForm.addEventListener('change', (event) => {
  const formData = new FormData(event.target)
  const filters = Object.fromEntries(formData)

  const result = []
  for (let i = 0; i < books.length; i++) {
    const titleMatch = filters.title.trim() === '' || books[i].title.toLowerCase().includes(filters.title.toLowerCase())
    const authorMatch = filters.author === 'All Authors' || books[i].author === filters.author
    const genreMatch = filters.genre === 'All Genres' || books[i].genres.includes(filters.genre)

    if (titleMatch || authorMatch || genreMatch) {
      result.push(books[i])
    }
  }

  console.log(result)
})



const searchCancel = document.querySelector('[ data-search-cancel]');
searchCancel.addEventListener('click', () => {
    const searchDialog = document.querySelector('[ data-search-overlay]')
```

Ln 179, Col 1 Spaces: 2 UTF-8 LF JavaScript Port : 5500 Sp

My code with all changes part 1

```
const searchCancel = document.querySelector('[ data-search-cancel]');
searchCancel.addEventListener('click', () => {
    const searchDialog = document.querySelector('[ data-search-overlay]')
    searchDialog.close() // This code should enable the button to  close the dialog(overlay).
    })


// Create an "All Genres" option
const selectWrapper = document.querySelector('[data-search-genres]');

const allGenresOption = document.createElement("option");
allGenresOption.value = "any"; //What it actually is.
allGenresOption.text = "All Genres"; // what it reads as.
selectWrapper.appendChild(allGenresOption);

// Loop through the genres object and create an option for each genre
for (const id in genres) {
    if (Object.hasOwnProperty.call(genres, id)) { // Getting the genre names that are property values
      const option = document.createElement("option");
      option.value = id;
      option.text = genres[id];
      selectWrapper.appendChild(option);
    }
  }
```

My code with all changes part 2

```
  // Create an "All Authors" option
const selectWrapper2 = document.querySelector('[data-search-authors]'); // Fetching the e

const allAuthorsOption = document.createElement("option");
allAuthorsOption.value = "any"; //What it actually is.
allAuthorsOption.text = "All Authors"; // what it reads as.
selectWrapper2.appendChild(allAuthorsOption);

// Loop through the authors object and create an option for each author
for (const id in authors) {
    if (Object.hasOwnProperty.call(authors, id)) {
      const option = document.createElement("option");
      option.value = id;
      option.text = authors[id];
      selectWrapper2.appendChild(option);
    }
  }


// This code below displays a message if no books were found under the current filters.

const message = document.querySelector('[data-list-message]')
  if (booksOfTheList.length < 1 ){
    message.setAttribute('open',true)
  }else{
    message.setAttribute('open', false)
  }
```

My code with all the changes part3

To conclude

1.Most errors were made with variable declarations so it would be better to use const to declare variables in the future as this will prevent scope problems.

2. Use functions to add functionality to the code e.g if you want a overlay to close it will be better to let the overlay close inside a function (overlay.close() or overlay.setAttribute('open', false)) and then add a eventlistener to the close button and parse the function.