DWA_09.3 Challenge 1

In this challenge, you will continue with your “Book Connect” codebase and further iterate on your abstractions.

Previously, you worked on adding abstraction around the book preview functionality of the project. Next, you must turn the book preview abstraction into a fully-working web component. Then, apply the techniques you’ve learned about this module to the book preview.

---

1. What problems did you encounter converting the book preview to a component?

This is the code

```
//@ts-check

/**
 * Creates a preview element for a book.
 *
 * @param {Object} bookData - The data of the book for which to create the
preview.
 * @param {Object} authors - An object containing author information with
author IDs as keys and author names as values.
 * @returns {HTMLElement} The created preview element.
 */
export function createPreview(bookData, authors) {
    // Destructure bookData
    const { author, id, image, title, genre } = bookData;

    // Create the main preview element
    const preview = document.createElement('div');
    preview.classList.add('preview');
    preview.setAttribute('data-preview', id);

    // Create the book image element
    const imageElement = document.createElement('img');
```

```
    imageElement.classList.add('preview__image');
    imageElement.src = image;

    // Create the book information container
    const info = document.createElement('div');
    info.classList.add('preview__info');

    // Create the book title element
    const titleElement = document.createElement('h3');
    titleElement.classList.add('preview__title');
    titleElement.textContent = title;

    // Create the book genre element
    const genreElement = document.createElement('div');
    genreElement.classList.add('preview__genre');
    genreElement.textContent = genre;

    // Get the author's name from the authors object
    const authorName = authors[author];

    // Create the book author element
    const authorElement = document.createElement('div');
    authorElement.classList.add('preview__author');
    authorElement.textContent = authorName;

    // Append elements to the info container
    info.appendChild(genreElement);
    info.appendChild(titleElement);
    info.appendChild(authorElement);

    // Append elements to the main preview element
    preview.appendChild(imageElement);
    preview.appendChild(info);

    return preview;
}
```

Importing Stuff: In our code, we often need to use functions and data from other files. This is like borrowing tools from a toolbox to build something.

Reusability: Components, like the one we're creating, are meant to be used multiple times. It should work for different books with different information, making it like a reusable building block.

Event Handling: If our component needs to respond to events, such as when someone clicks on it to see more details, we have to think about how to handle these events. We need to make sure the component knows what to do when events occur.

Dynamic Data: Since our component will be used for various books, it needs to handle changing data. The book's title, author, and other details will be different for each book. We need a way to give this data to the component so it can display the correct information.

---

2. What other elements make sense to convert into web components? Why?

As I was going through the code, I noticed several parts that could benefit from being converted into web components. By encapsulating elements like the theme selector, search form handling, book list rendering, settings button functionality, and others. Converting these elements into web components allows me to promote code reusability, encapsulation, and a more modular design. This approach has the potential to simplify the development efforts by reducing duplicated code and making it easier to maintain specific functionalities.

---

3. Why does keeping your HTML, CSS and JavaScript in a single file sometimes make sense?

From what I gather, having everything in one file can simplify things, especially for smaller projects. It's easier to distribute and share. But I'm also aware of the downsides. As projects get bigger, maintaining everything in a single file might become challenging. Separating concerns between HTML, CSS, and JavaScript is a good practice for readability.
