DWA_07.4 Knowledge Check_DWA7

---

1. Which were the three best abstractions, and why?
    - CreatePreviewItems function: This function is responsible for creating a preview element for a book. It is a single responsibility function.

    - PopulatePreviewItems function: This function populates a document fragment with preview elements for a range of books (first 36 books). It is considered a single responsibility.

    - PopulateGenres function: This function populates a document fragment with genre options for filtering. It is a single responsibility function.

---

2. Which were the three worst abstractions, and why?

- Global Imports: using explicit named imports: (import { books, authors, genres, BOOKS_PER_PAGE } from ‘ ./data.js;’) can lead to maintenance issues when imported file changes. As each import statement needs to be updated. It is better to use module or namespace import (import * as data) and accessing the variables using the data namespace.

- Global Variables: The use of global variables (page and matches) can introduce state-related issues and make the code harder to reason about it. It would be better to encapsulate this state within an object or class to manage the page number and book matches.

- Event Listener Setup: The code sets up event listeners for various elements using separate functions. (setSearchCancelButton, setSettingsCancelButton etc). The approach works, but it would be more concise and easier to manage by combining these functions into a single function responsible for setting up all the event listeners.

---

3. How can The three worst abstractions be improved via SOLID principles.

- Event Listener Setup: The code sets up event listeners for various elements using separate functions. (setSearchCancelButton, setSettingsCancelButton etc). The approach works, but it would be more concise and easier to manage by combining these functions into a single function responsible for setting up all the event listeners.

- Mixing HTML and JavaScript. The createPreviewElement function includes HTML markup directly within the JavaScript code, which makes it harder to read and maintain. Separating HTML markup into separate templates or utilizing a template library can improve code readability and separation of concerns.

- Lack of Error Handling: The code does not include proper error handling mechanisms, such as try-catch blocks or error callbacks. Adding error handling logic can prevent unexpected errors from crashing the application and provide a better user experience.
