DWA_07.4 Knowledge Check_DWA7

---

1. Which were the three best abstractions, and why?

- Create Preview Element: The **createPreviewElement function** abstracts the process of creating a preview element for a book. It takes a book object as input and returns an HTML button element representing the preview. This abstraction encapsulates the details of creating the preview structure, including the image, title, and author information. By abstracting this functionality into a separate function, it promotes code reuse and improves code readability.

- Create Option Element: The **createGenreOption** and createAuthorOption functions abstract the creation of option elements for genres and authors, respectively. These functions take the ID and name of a genre/author and return an HTML option element. By abstracting the creation of option elements, the code becomes more modular and reusable. It centralizes the logic for creating option elements and avoids code duplication when generating the genre and author dropdown menus.

- Handle Search Form Submit: The **handleSearchFormSubmit** function abstracts the logic for handling the submit event of the search form. It extracts the form data, applies filters based on user selections, and updates the UI accordingly. This abstraction separates the concerns of handling form submission from the main code logic, making the code easier to understand and maintain. It promotes the single responsibility principle and improves code organization.

---

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

- Search and filtering functionality: The search and filtering functionality in the **handleSearchFormSubmit** function could be abstracted into separate functions. Breaking down the logic into smaller, reusable functions can make the code more modular and easier to understand. It would also allow for easier testing and maintenance.

- Direct DOM Manipulation: The code directly manipulates the DOM by creating elements and appending them to specific containers. This tightly couples the JavaScript code with the HTML structure, making it harder to modify or refactor the HTML layout without impacting the JavaScript logic. Using a modern JavaScript framework or library that supports declarative rendering and component-based architecture, such as React or Vue.js, would provide better separation of concerns and maintainability.

- Event Handling: The code manually attaches event listeners to DOM elements using imperative event handling. While it works, it can lead to potential memory leaks or event listener conflicts if not properly managed. Utilizing a library or framework that provides a more declarative approach to event handling, such as event delegation or reactive event binding, would simplify event management and improve code maintainability.

---

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

3.1 Single Responsibility Principle (SRP): Each function or class should have a single responsibility. Break down the handleSearchFormSubmit function into smaller functions, where each function handles a specific task. For example:

- **getFilteredBooks**: A function that takes the form data and returns the filtered list of books based on the search criteria.

- **updateListItems**: A function that updates the list of book previews based on the filtered books.
- **updateListButton**: A function that updates the "Show more" button based on the remaining books.

3.2 Open/Closed Principle (OCP):

- Identify code that needs to be modified when adding new functionality.
- Encapsulate the varying behavior behind an interface or an abstract class.
- Implement new functionality by extending the interface or abstract class rather than modifying existing code.

3.3 Liskov Substitution Principle (LSP):

- Make sure that subclasses can be used in place of their base classes without affecting the correctness of the program
- Avoid violating the preconditions, postconditions, and invariants of the base class when implementing subclasses.
- Ensure that the behavior of the subclass is consistent with the behavior expected from the base class.
