DWA_07.4 Knowledge Check_DWA7

---

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

    - BookElement Function: The BookElement function abstracts the creation of a book element/button. It takes in the necessary data (author, id, image, title) and generates the HTML structure for displaying a book preview. By encapsulating this logic within a function, it enhances code reusability and makes it easier to generate book elements dynamically.

    - Update List Functions: The updateListButton, updateListItems, and updateListMessage functions abstract different aspects of updating the book list. These functions handle updating the "Show more" button text and disabled state, updating the book elements displayed on the page, and showing or hiding a message based on the number of matches or books, respectively. By separating these concerns into individual functions, the code becomes more modular and easier to understand.

    - Event Handler Functions: The code utilizes several event handler functions like handleListButtonClick, handleSearchCancel, handleSettingsCancel, handleHeaderSearchClick, handleHeaderSettingsClick, handleSettingsFormSubmit, handleSearchFormSubmit, and handleListItemsClick. These functions encapsulate the logic for handling various user interactions, such as button clicks, form submissions, and book item clicks. By abstracting these event handling operations, the code becomes more organized and maintainable.

    These abstractions contribute to code readability, maintainability, and reusability, making it easier to understand and modify different parts of the code without introducing unnecessary complexity.

---

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

- Event Handler Functions: Although the event handler functions help separate the logic for handling user interactions, they could benefit from further modularization. Currently, some of the event handler functions contain multiple responsibilities and could be broken down into smaller, more focused functions. By extracting specific tasks into separate functions, it would enhance the code's readability and make it easier to understand and maintain.

- HandleListItemsClick Function: The handleListItemsClick function includes the event listener setup within the function itself. This approach could lead to potential issues if the function is called multiple times or needs to be reused elsewhere. It would be better to separate the event listener setup into the initialization section or a separate function to improve code organization and avoid potential issues.

- UpdateList Function: The updateList function currently calls the updateListItems, updateListButton, and updateListMessage functions sequentially. While this approach provides a straightforward way to update the book list, it could be improved by implementing a more efficient approach. For example, instead of updating the entire list each time, a more optimized solution could be to only update the elements that have changed or need to be updated. This would result in better performance, especially when dealing with large datasets.

---

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

- Single Responsibility Principle (SRP): Each function or module should have a single responsibility.

To improve the event handler functions and adhere to the SRP:

Break down the event handler functions into smaller, focused functions, each responsible for handling a specific user interaction. For example, separate the logic for canceling search, canceling settings, submitting search form, and so on, into separate functions. This would help in better organization and maintainability.

- Open-Closed Principle (OCP): Entities should be open for extension but closed for modification.

To improve the handleListItemsClick function and adhere to the OCP:

Extract the setup of the event listener into a separate function that can be called during initialization or whenever needed. This way, the function responsible for handling the event does not need to handle the event listener setup, making it easier to extend the behavior without modifying the original function.

- Interface Segregation Principle (ISP): Clients should not be forced to depend on interfaces they do not use.

To improve the updateList function and adhere to the ISP:

Consider introducing an interface or abstraction that defines the contract for updating the book list. This way, the function does not need to directly call the specific update functions like updateListItems, updateListButton, and updateListMessage. Instead, it can rely on the interface and interact with a more generic update mechanism that can be implemented by different classes or functions. This promotes loose coupling and allows for easier modification and extension.
