DWA_07.4 Knowledge Check_DWA7

---

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

1. `createButtonElement`: This function abstracts the creation of a button element representing a book. It encapsulates the logic of creating a button with the necessary HTML structure and content based on the book object. This abstraction enhances code readability and reusability by separating the creation of the button element from the surrounding code.

2. `createOptionElement`: This function abstracts the creation of an option element for a select dropdown. It takes in the value and text parameters and creates an option element with the specified values. By encapsulating this functionality, the code becomes more modular and easier to understand.

3. `handleSearchFormSubmit` and `handleSettingsFormSubmit`: These functions handle the submission of the search and settings forms, respectively. They encapsulate the logic for processing form data and updating the UI based on the submitted values. By abstracting this functionality into separate functions, the code becomes more maintainable and easier to modify.

---

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

1. `initializeList`: Although this function initializes the starting list of book items, it directly manipulates the DOM by creating button elements and appending them to the document fragment. This violates the Single Responsibility Principle (SRP) because it combines the responsibilities of initializing the list and manipulating the DOM.

2. `updateListButton`: This function updates the "Show more" button text and remaining book count. However, it also directly manipulates the DOM by selecting and modifying DOM elements. Similar to the previous point, this violates the SRP. The DOM manipulation should be abstracted into a separate function or component, while `updateListButton` should focus on updating the button text and book count.

3. `handleListItemsClick`: This function handles the click event on the list items and performs a series of DOM manipulations to display the selected book's details. Like the previous points, this violates the SRP. The responsibility of handling the click event should be separated from the DOM manipulation logic. The function should extract the book details from the event and invoke a separate function or component responsible for displaying the book details.

---

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

Improvements via SOLID principles:

1. Single Responsibility Principle (SRP): The functions `initializeList`, `updateListButton`, and `handleListItemsClick` violate the SRP by combining multiple responsibilities. To improve this, the code can be refactored to have separate functions or components for initializing the list, updating the button, and handling list item clicks. This separation of concerns enhances code readability, modularity, and maintainability.

2. Open/Closed Principle (OCP): The code doesn't explicitly violate the OCP. However, by adhering to SOLID principles, the code can be designed in a way that is easier to extend with new features. For example, if new form fields or UI components need to be added, the code should be structured in a modular and flexible manner, allowing for easy extension without modifying existing code.

3. Dependency Inversion Principle (DIP): The code could benefit from applying the DIP by decoupling the dependencies between different components or modules. Currently, the code directly accesses global variables and DOM elements within the functions, making them tightly coupled. By introducing abstractions and dependency injection, the code can be made more modular and testable, enabling easier maintenance and future modifications.
