DWA_07.4 Knowledge Check_DWA7

1. Which were the three best abstractions, and why?
    a) BookApp class: this class encapsulates the functionality of the book app. It abstract away the implementation details of setting up the app, handling search functionality, setting up themes, managing lists, and handling events. It separates the app logic into a class, it promotes modularity , reusability and maintainability.
    b) Book class: It encapsulates the book's properties and provides a clean interface to create a book element with required HTML structure. This abstraction allows for easy addition or modification of book properties and presentation without affecting the overall app logic.
    c) setupListButton() method : By encapsulating this functionality in a separate method, it improves code readability and maintainability.

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

    a) init() method: Although the init() method is responsible for initializing the Book App, it has multiple responsibilities, including setting up preview items, genre options, author options, theme, cancel buttons, search button, settings form, list button, and list items. This violates the Single Responsibility Principle (SRP) because the method should have a single responsibility and not be responsible for multiple tasks.
    b) setupCancelButton() method: This method sets up the functionality for the cancel button in the overlay. However, it takes two selector parameters, which makes it tightly coupled to specific elements in the DOM. It would be better to abstract the functionality of handling the cancel button click event in a more generic way, allowing for reusability and flexibility.
    c) setupGenreOptions() and setupAuthorOptions() methods: These two methods have similar functionality of setting up options in the search form. They share common code for creating option elements based on the data. Instead of

duplicating the code, it would be better to abstract the common functionality into a separate method or helper function to avoid code duplication.

---

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

a) init() method: Refactor the init() method to adhere to the Single Responsibility Principle (SRP). Split the method into smaller, more focused methods, each responsible for a specific initialization task, such as setupPreviewItems(), setupGenreOptions(), setupAuthorOptions(), etc. This will make the code more modular, maintainable, and easier to test and understand.

b) setupCancelButton() method: Instead of directly selecting the elements using specific selector strings, introduce a more generic approach. Create a method, such as setupCancelHandler(), that accepts a cancel button element and an overlay element as parameters. Inside the method, attach the event listener to the cancel button and handle the logic for closing the overlay. This way, the method can be reused for any cancel button and overlay combination, promoting code reuse and flexibility.

c) setupGenreOptions() and setupAuthorOptions() methods: Extract the common code for creating option elements into a separate helper function, such as createOptionElement(). This function can take parameters like the element value and text content and return a created option element. Then, modify the setupGenreOptions() and setupAuthorOptions() methods to use the createOptionElement() function to create the option elements. This reduces code duplication and improves maintainability by having a single place to modify if changes are required in the option creation process.