DWA_07.4 Knowledge Check_DWA7

---

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

1.Book Element Creation Function. In the code, there is a recurring pattern for creating book preview elements with image, title and author details. Abstracting this process into a function can reduce code duplication, make the code short and to the point, and ensure consistency in the appearance of book previews.

2.Find Active Book Function. The findActiveBook function abstracts the process of identifying the active book based on the clicked element's 'data-preview' attribute. This abstraction enhances code readability and separates the book-finding logic from the event listener, making the code easier to understand.

3. Applying a functions within a function as abstraction helps modularize the code and make it easier to understand. Instead of having one function that performs multiple tasks, breaking it down into separate functions for each task makes it easier to focus on each task makes the code more manageable and easier to modify. It also makes it easier to focus on each task independently, without worrying about the details of other tasks.

---

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

1. Search Filtering Logic. The code that handles form submission and book filtering based on search criteria could be improved in terms of abstraction. It combines various filtering conditions in a single loop, which can become harder to maintain as more filtering criteria are added.

2. Form Data Processing Abstraction. While abstracting form data processing is a good idea, the code still contains some repetition in the event listeners for form submissions. The same logic for extracting form data and applying theme changes appears in both the settings and search forms, which could be further abstracted.

3. DOM Manipulation Functions. The code directly manipulates the DOM (e.g… appending elements, setting attributes0, which can make the code less maintainable and harder to test.

---

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

1. We can focus on enhancing the Single Responsibility Principle (SRP):
Separate the responsibilities of UI interaction and search filtering logic in a single event handler function.
Enhancing the Open/Closed Principle (OCP):
Apply the OCP by making the filtering logic open for extension and closed for modification.

2. Single Responsibility Principle (SRP):
Separate the responsibilities of form data processing and DOM manipulation into distinct classes or functions.
Dependency Inversion Principle (DIP):
Apply the DIP by introducing dependency injection to allow for more flexible component composition.

3. Single Responsibility Principle (SRP) and Open/Closed Principle (OCP):
Separate the responsibilities of DOM manipulation and event handling into distinct classes or functions.
Apply the OCP by creating an extensible mechanism for adding event handlers without modifying existing code.
