DWA_07.4 Knowledge Check_DWA7

---

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

- createElement Utility Function
The createElement function abstracts the process of creating DOM elements with attributes and inner HTML. It reduces code duplication and enhances code readability by encapsulating the repetitive task of creating elements. This function is reusable and promotes a consistent structure for creating elements.

- populateSelect Utility Function
The populateSelect function abstracts the process of populating select elements with options based on data. It simplifies the code and enhances maintainability by encapsulating the logic for creating and appending options. It also provides a default option for select elements, improving user experience.

- updateBookList Function
The function abstracts updating the book list based on filtered data. It encapsulates the logic for updating the displayed books, including handling pagination, message display, and enabling/disabling the “Show more” button.

---

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

- Inline Event Listeners
The code uses inline event listeners on HTML elements(e.g. ‘onclick’). This mixes the structure of HTML and JavaScript, making the code less modular and harder to maintain.

- Hard-coded Theme Handling
The theme handling is hard-coded based on a media query, making it less flexible and hard to extend.

- Manual DOM Element Search

The code uses ‘document.querySelector’ multiple times to find and manipulate DOM elements.

---

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

- Move event handling to a separate module or class, adhering to the Single Responsibility Principle.
- Create a ThemeManager class that can be easily extended to support different themes. This allows the code to be open for extension while keeping existing functionality closed for modification.
- Create a View class or module responsible for managing the DOM elements. Encapsulate the DOM manipulation logic within this class, adhering to the SOLID.

By applying SOLID principles, you can create more flexible and maintainable code, making it easier to extend and modify your application as it evolves.
