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 for a book. It takes in a book object and returns a button element with the necessary HTML structure and data attributes. It encapsulates the logic for creating the button, making it reusable and modular.

2. createOptionElement: This function abstracts the creation of an option element for a select dropdown. It takes in a value and text and returns an option element with the appropriate properties set. It provides a clear interface for creating options, improving code readability and maintainability.

3. updateListButton: This function updates the "Show more" button text and remaining book count based on the current page and the number of matches. It abstracts the calculation and updating of the button's content, making it easier to manage and maintain.

---

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

1. handleSearchFormSubmit: This function handles the form submit event for the search form. It performs multiple tasks such as form data extraction, filtering books, updating the UI, and scrolling to the top. This function violates the Single Responsibility Principle (SRP) because it takes on too many responsibilities. It should be refactored to separate concerns and delegate responsibilities to smaller functions or modules.

2. initializeList: This function initializes the starting list of book items by creating button elements and appending them to the DOM. While it's responsible for initialization, it directly manipulates the DOM, making it tightly coupled to the HTML structure. It would be better to abstract the

DOM manipulation into a separate module or utilize a templating library to achieve a cleaner separation of concerns.

3. handleListButtonClick: This function handles the "Show more" button click event. It retrieves additional book items based on the current page and appends them to the list. Similar to initializeList, it directly manipulates the DOM, which can lead to maintenance issues and reduced flexibility. Consider separating the DOM manipulation logic into a dedicated module or adopting a more modular approach.

---

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

1. handleSearchFormSubmit: Refactor this function to adhere to the SRP. Extract separate functions for form data extraction, book filtering, UI updating, and scrolling to the top. Each function should have a single responsibility, making the code more modular and maintainable.

2. initializeList and handleListButtonClick: Separate the DOM manipulation logic from these functions by utilizing a templating library like Handlebars, Mustache, or adopting a more modular approach such as a component-based architecture (e.g., React, Vue.js). By decoupling the DOM manipulation, you can achieve cleaner code with better separation of concerns.
