DWA_07.4 Knowledge Check_DWA7

---

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

Selectors Object: The `selectors` object is a valuable abstraction that encapsulates the selection and manipulation of HTML elements. It centralizes the element selection logic in one place, making it easier to manage and update if you ever need to change the structure of your HTML. This abstraction enhances code readability and maintainability.

Event Handling Functions: You've abstracted event handling into separate functions like `bookButton`, `viewBook`, `settingsEvents`, `themeUpdate`, `moreBooks`, `searchFunctions`, and `createSearchHTML`. These functions encapsulate the behavior associated with various user interactions. This abstraction promotes modularity and readability by separating concerns and making it clear how different events are handled in your application.

HTML Generation with `createPreviewsFragment`: The `createPreviewsFragment` function abstracts the process of generating HTML elements for book previews. It takes care of creating buttons, setting attributes, and populating the elements with data from your `books` array. This abstraction simplifies the code responsible for adding books to the webpage and makes it easier to extend or modify the HTML structure if needed.

---

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

Implicit Global Variables: The code uses several global variables without declaring them explicitly using the `var`, `let`, or `const` keywords. For example, the `page` variable and `frag` variable are declared globally but are not explicitly marked as such. This can lead to unintended consequences or naming conflicts if the codebase grows. It's a good practice to declare variables explicitly within the scope where they are used.

Event Binding Inside Loops: The code binds event listeners inside a loop using `for...of`. For instance, event listeners for the `viewBook` function are added within a loop that iterates over `selectors.oneBook`. While this works, it can be less efficient and might lead to performance issues when dealing with a large number of elements. It's often

better to use event delegation or bind event listeners once at a higher level to improve performance.

Inefficient DOM Manipulation: The code frequently manipulates the DOM by creating and appending elements, which can be inefficient when dealing with a large number of elements. The `createPreviewsFragment` function, for example, creates and appends buttons for book previews. While this is acceptable for a small number of books, it can be slow for a large dataset. Consider optimizing DOM manipulation, such as using Document Fragments for batch appending or virtual DOM libraries if the complexity of your application increases.

It's important to note that these points don't necessarily represent "worst" abstractions but areas where code can be enhanced for better performance, readability, and maintainability. Effective coding practices can help address these concerns and improve the overall quality of your codebase.

---

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