DWA_07.4 Knowledge Check_DWA7

---

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

Function Abstraction: The preview function abstracts the creation and rendering of a book preview. It takes in parameters such as author, id, image, and title, and generates an HTML button element with corresponding data and inner content. This function encapsulates the logic for creating a preview and improves code readability and reusability.

DOM Manipulation Abstraction: The code uses several DOM manipulation methods such as document.createElement, element.setAttribute, element.innerHTML, document.querySelector, etc., to create and modify HTML elements. These methods abstract the underlying DOM operations and provide a higher-level interface for interacting with the HTML document.

Data Generation Abstraction: The createOptions function abstracts the generation of select options for genres and authors. It takes an option label and a corresponding data object as parameters and generates HTML option elements based on the provided data. This abstraction simplifies the process of generating dynamic select options and can be reused for different data sets.

---

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

Lack of Modularization:
The code seems to be written in a single file without proper modularization or separation of concerns. It contains a mix of functionalities, including importing data, creating previews, generating options, setting themes, updating UI elements, adding event listeners, and more. This lack of modularization makes the code harder to understand, maintain, and test. It would be better to split the code into smaller, reusable functions or modules that handle specific tasks.

Mixing HTML Manipulation with JavaScript Logic:

The code directly manipulates HTML elements using document.querySelector, setAttribute, innerHTML, and other similar methods. This approach tightly couples the JavaScript logic with the HTML structure, making it harder to change the HTML layout or reuse the JavaScript code. A better approach would be to use a templating engine or a component-based framework that separates the HTML and JavaScript code, providing better maintainability and reusability.

Inline CSS Style Manipulation:
The code directly manipulates CSS styles by setting properties on the document.documentElement.style object. This approach mixes the presentation logic with the JavaScript code, making it harder to manage and update the styles. It would be better to define CSS classes with appropriate styles and use JavaScript to add or remove those classes dynamically.

---

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

Open/Closed Principle (OCP):

Design the code in a way that allows for easy extension without modifying the existing code.
Use interfaces or abstract classes to define the behavior of different components, such as the preview generator, option generator, theme setter, and event listener manager. This way, new implementations can be added without modifying the existing code.

Interface Segregation Principle (ISP):

Split the original code into smaller, more focused classes or modules that provide specific functionality. Each class or module should expose only the necessary methods and properties, avoiding a large and bloated interface.

Single Responsibility Principle (SRP):

Extract the logic for setting the theme based on user preference into a separate class or module. This class should handle the theme setting based on the user's preference.

Create a separate class or module to handle event listeners and their associated logic. This class should be responsible for attaching event listeners to the appropriate elements and executing the necessary actions when events occur.
