DWA_07.4 Knowledge Check_DWA7

---

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

createBookItem function
The function has a well defined purpose. It is designed to create a book item element based on a provided match object. The use of document fragment for efficient DOM manipulation, especially when multiple DOM updates are involved.

populateDropdownWithOptions function
The abstraction has a clear descriptive purpose which clearly communicates the purpose of its behavior.
The code extracts data from the ‘data’ object using ‘Object.entries’ which is good for processing key-value pairs.

applyThemeSettings function
The logic based on the ‘isDarkMode’ variable allows for dynamic theme switching based on the users preferences for light or dark mode. The Element selection is efficiently used to relevant HTML elements to apply the settings, such as the theme element and the root style. The function also has a well defined purpose.

---

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

getFilteredBooks function
Readability can be improved; the code uses more nested loops and conditions, making it less readable. Breaking down the logic into smaller well named helper functions can be beneficial.The function contains complex logic to filter books based on genre, tittle and author.

setThemesColors function
Hard coded values: the function uses hard coded values (‘255, 255, 255’) directly within the function.It would be better to make those values configurable.
Lack of clear commenting.

displayBooks function
Inefficient DOM manipulation; the function clears the inner HTML of an elements and appends new items but a more efficient approach might be to append new items directly without clearing the existing content.

---

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

Single responsibility principle (SRP) the function is responsible for filtering books based on specific criteria. Splitting the function into single responsibility function may be impactful.
Open-closed principle designing the function to be open for extension but closed for modification.This allows for future changes without altering the existing code.

setThemeColors function
Single responsibility principle (SRP) ensure that the function has a single responsibility which is to set the theme colors.
Dependency injection: instead of hard coding color values, inject these values as parameters. This allows for greater flexibility and configability.

```
const setThemeColors = (theme, darkColor, lightColor) => {
   const rootStyle = document.documentElement.style;
   rootStyle.setProperty('--color-dark', darkColor);
   rootStyle.setProperty('--color-light', lightColor);
}
```

displayBooks function
Dependency Inversion principle (DIP) to improve the efficiency and readability of the function dependency injection could be used to provide a way to interact with the DOM.
Interface segregation principle (ISP): if the function interacts with a variety of the DOM elements, we can consider splitting the function into smaller functions each responsible for interacting with a specific element. Thus making the code more modular.
