DWA_07.4 Knowledge Check_DWA7

---

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

- Having **separate modules for different functionalities of the app**.
  This is one of the best abstractions because it meets the first SOLID principle: Single Responsibility Principle - *create separate modules or classes that handle specific tasks or responsibilities.*

- **The html references module**. This is one of the best abstractions as it reduces the use of document.querySelector() in the mainscript and thus reducing making the code easier to follow. The references module also makes for easier access of the DOM elements.

- The **genreAndAuthorsOptions module**. In this module I placed all the code responsible for creating the first options and all other options for both the genres and authors into one function. This was my attempt to reduce repetition of code as the code for genre does the same thing as the code for loading the author options.

---

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

- All my functions were **not reusable eg, genreAndAuthorOption module, that function is not reusable**. This is one of the worst abstractions because to maybe add more functionality to my code I would have to directly modify my code instead of extending from the current code. This goes against the SOLID principle: Open-Closed-Principle (OCP) - *This principle states that software entities (classes, modules, functions, etc.) should be open for extension but closed for modification.*

- Instead of creating functions to reduce some other repetitive code, I split a lot of my code into modules. Even though the modules have single functionality, there was still some repetition between them.

---

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

- I should write functions that are more reusable, functions that are pure with no side effects. For example, for the function in the display.js module I could have created a function that takes in the books as a parameter. This would ensure the reusability of that function in the search.js module's first function from line 46. This solution also applies to my second worst abstraction. Applying this solution would be inline with the Open-Closed-Principle (OCP) - *This principle states that software entities (classes, modules, functions, etc.) should be open for extension but closed for modification.*
