DWA_07.4 Knowledge Check_DWA7

SOLID Principle:

1. **Single responsibility principle** - functions responsible for a single function
2. **Open-closed principle** - functions should be open for extension, closed for modification
3. **Liskov substitution principle** - subtypes must be substitutable for their base types without altering the correctness of the program.
4. **Interface segregation principle** - only create functions that the user will need, avoid unnecessary ones they won't use.
5. **Dependency inversion principle** - high-level modules should not depend on low-level modules. Instead, both should depend on abstractions.

---

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

1. **Function filterBooks**: I made a function that filters books based on the search criteria, either title, author, or genre. Originally this was split into 3 different blocks of similar code, which I combined to create a single function, in order to reduce duplication of code. It's my best abstraction because it reduced the code and made it overall easier to understand.
2. **Function closeOverlay**: Previously these were two separate code blocks for handling the closing of the search and settings overlay, which I consolidated into a single function that handles the closing of overlays.
3. **Function updateTheme**: I consolidated the functionality to update the themes into one function. The code previously was separated into different code blocks. This new function makes the code much more understandable.

---

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

1. **Problem 1 - Descriptive naming**: Some of the variable and function names are too vague. For example, "page" is too generic. It could definitely do with more descriptive names.
2. **Problem 2**: This part of the code is quite confusing and should have been put in a function such as "updateListDisplay'.

```
  if (active) {
    document.querySelector('[data-list-active]').open = true;
    document.querySelector('[data-list-blur]').src = active.image;
    document.querySelector('[data-list-image]').src = active.image;
    document.querySelector('[data-list-title]').innerText = active.title;
    document.querySelector('[data-list-subtitle]').innerText = `${authors[active.author]} (${new
Date(active.published).getFullYear()})`;
    document.querySelector('[data-list-description]').innerText = active.description;
  }
});
```

3. **Problem 3**: While the filterBooks function is overall a good function as it combines the filter logic of title, author, genre. If this website were to be expanded on it would be harder to reuse this code as it's a single function, especially if you only need to target a specific filter such as author.

---

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

1. Applying SOLID to problem 1:: Descriptive naming is not related to the SOLID principles, however it is a form of abstraction (it abstracts away the implementation details and conveys the intention) that improves the readability of the code.

2. Applying SOLID to problem 2: This violates the SRP rule as the code block is responsible for multiple tasks, updating multiple elements on the DOM, and would be improved by enforcing the SRP.

3. Applying SOLID to problem 3: It could be improved by refactoring the code into 3 separate functions. The SRP states that functions should be responsible for a single function. In this way the function would be split into 3 separate functions. But as it is now the function is responsible for 3 different functions. This would in turn align with the OCP as it makes the functions more open for extension if they are separated. This aligns with the ISP as these functions are necessary. This does align with the DIP as these modules would all rely on abstractions.
