DWA_07.4 Knowledge Check_DWA7

---

1. Which were the three best abstractions, and why?
-Grouping all genres and authors on the same modular
-Configuring functions to create a task
-How to export and import declarations

implementation of methods is made easier. I could declare a method in my theme class called theme select so that I can be able to choose which theme(day or night mode) I want to select . I also declared genreFragment and authorFragment in the same modular since they are fragments.

---

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

-Grouping elements on the same modular
-When to throw unhandled errors
-repeated elements

They keep giving error on my code if am running it, especially queryselectors because I have many of them.

---

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

- **Open-Closed Principle (OCP):** This principle states that software entities (classes, modules, functions, etc.) should be open for extension but closed for modification. In JavaScript, you can apply this principle by creating abstract classes or interfaces that define a set of methods or properties. Concrete classes can then inherit from these abstract classes or interfaces and implement their methods. This way, you can add new functionalities without modifying existing code.

- **Liskov Substitution Principle (LSP):** This principle states that objects of a superclass should be replaceable with objects of a subclass without affecting the correctness of the program.

- **Interface Segregation Principle (ISP):** This principle states that a client should not be forced to depend on methods it does not use. In JavaScript, you can apply ISP by creating small, focused interfaces that define only the methods that a client needs. This way, you can avoid creating large, bloated interfaces that are difficult to maintain and understand.
