DWA_07.4 Knowledge Check_DWA7

---

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

1.1 Applying functions within a function as abstraction allows these functionalities into separate functions, the code becomes more organized, modular, and easier to understand. It abstracts away the details of checking the color scheme and calling the necessary functions to set the theme and theme colors.

For example (I've set separate functions for each preferred color scheme within a function called setThemeBasedOnColorScheme)

1.2 Adding an object as an abstraction, where you can organize related code into the same local scope. This improves code readability, and separate concerns. It also allows you to reuse the object's methods throughout your codebase, making it easier to maintain and modify the theme-related functionality.

---

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

2.1 Abstractions that hide important details , if they hide important details that are hard to understand or debug, can lead to problems with finding and solving the problem for anyone working on the code .

2.2 Lack of documentation or further explanations of the code can also prevent the usage and maintenance of abstractions.

2.3 Abstractions with poor naming or documentation ,clear and descriptive naming is very important for understanding the purpose and behavior of an abstraction.Poor naming and documentation can lead to misinterpretation and make the code difficult to understand and work on especially for collaboration with other developers.

---

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

3.1 Interface Segregation Principle (ISP): Design interfaces that expose only the necessary and relevant information to clients, hiding implementation details that are not needed. This promotes a clearer separation between the client and the implementation.

3.2 Single Responsibility Principle (SRP): By applying the SRP, each abstraction will have a clear and well-defined purpose. This clarity allows for more accurate and descriptive naming. When an abstraction has a single responsibility, it becomes easier to choose meaningful names that accurately represent its purpose and behavior.
