DWA_07.4 Knowledge Check_DWA7

---

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

1. **“createPreviewButton” Function:**
   By passing in a book object as a parameter, it allows for dynamic creation of preview buttons for different books without duplicating code

2. **“createOptionsDocumentFragment” Function:**
   By separating the creation of options into a reusable function, it promotes code reuse and improves readability

3. **“appendToDropdown” Function:**
   It encapsulates the DOM manipulation involved in adding options to dropdown menus, improving code organization and readability. (For Example, the code wraps up all the steps needed to add choices to a dropdown menu in a tidy package).

---

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

1. **“setTheme” Function:**
   The "setTheme" function decides how to change the colors based on your preferred theme, but it also directly changes how things look on the webpage. This connection between how things work and how they look could make it tricky to fix or test the code later on.

2. **“toggleOverlay” Function:**
   The "toggleOverlay" function helps to show or hide a pop-up window, but it does this by directly changing parts of the webpage. This connection between how the pop-up works and how it looks could make it tricky to fix or test the code later on.

3. **“handlePreviewItemClick” Function:**
   The handlePreviewItemClick function deals with what happens when you click on a preview of a book. It looks for certain parts on the webpage and changes what they show. But, it's very connected to exactly how the webpage is set up.

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

1. **setTheme Function:**
   I will implement the Open/Closed Principle (OCP) to ensure that the setTheme function can be extended to accommodate different theme application strategies, such as using classes or functions, without needing to modify the existing setTheme function.
2. **“toggleOverlay” Function:**
   I will implement the Dependency Inversion Principle (DIP) to separate the toggleOverlay function from particular DOM elements. This will involve either accepting them as parameters or depending on abstractions/interfaces, ensuring that the function relies on high-level concepts like an OverlayManager interface instead of focusing on low-level details like specific DOM elements.
3. **“handlePreviewItemClick” Function:**
   We can use the Liskov Substitution Principle (LSP) to ensure that if we swap out any subclasses or versions used by the handlePreviewItemClick function, it won't mess up how it works. This helps us handle different types of previews more easily.