DWA_07.4 Knowledge Check_DWA7

---

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

The three best abstractions for me are :

- HTML Elements Object Literal
- Event Listeners and Handlers
- createPreview() Function

The use of abstractions such as the HTML Elements Object Literal, the createPreview() function, and event listeners and handlers provide several benefits. They enhance code organization by centralizing element references and encapsulating functionality into modular, reusable components. This improves code readability by providing meaningful names and promoting a clear understanding of the purpose of each element and function. Overall, these abstractions contribute to easier code maintenance, reduced redundancy, and improved code quality in web applications.

---

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

The three worst abstractions for me are :

- Lack of Modularization
- Inline Event Handlers
- Global Variables

Addressing potential issues in a codebase leads to improved code quality, maintainability, and collaboration. By writing clean and well-abstracted code, the software becomes more maintainable, scalable, and adaptable to future changes. This contributes to efficient development, reduces the likelihood of errors, and enhances overall software quality. Clear, understandable code fosters collaboration among developers, making it easier to work together on the project. Ultimately, prioritizing these practices results in a codebase that is easier to manage, understand, and extend.

---

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

- Code Duplication
- Unnecessary Abstraction
- Misleading Function Names

Applying SOLID principles to your codebase can lead to improved design and maintainability. By adhering to these principles, you can reduce duplication, enhance modularity, and make your code more adaptable to future changes. The SOLID principles promote a modular and flexible code structure, making it easier to understand and modify individual components without impacting the entire system. This approach ultimately results in more maintainable, scalable, and high-quality code.
