DWA_01.3 Knowledge Check_DWA1

1. Why is it important to manage complexity in Software?

a) Maintainability: Managing complexity makes the codebase more maintainable, this allows more straightforward bug fixes, updates, and enhancements.
b) Readability: Developers can have difficulty understanding complex structures if they are not properly managed. Complexity management improves code readability which results in easy-to-understand logic and structures.
c) Debugging and Testing: Managing complex software reduces the debugging and testing effort required to ensure the software functions correctly.
d) Collaboration: If and when the codebase is well managed, meaning it is maintainable and readable, it proves to be advantageous for team-based environments. It becomes easier for multiple developers to work on it efficiently.
e) Scalable: it also allows the software to be more adaptable and scalable to accommodate future growth and changes in requirements.

2. What are the factors that create complexity in Software?

Managing complexity in software is crucial for various reasons. The complexity of software can arise from factors such as the **desired functionality**, involving **sophisticated algorithms**, **intricate business rules**, and **complex interactions between components**. Additionally, **technical dependencies on libraries, frameworks, and external systems** can further contribute to complexity. A lack of modularity and poorly abstracted and encapsulated code make it difficult to manage complexity effectively. **Inheriting or dealing with legacy codebases adds to complexity due to outdated technologies, undocumented logic, and misalignment with modern development practices**. Furthermore, **frequent changes in requirements** can introduce complexity as developers must adapt existing code or introduce new features while maintaining overall coherence. By addressing these factors through techniques like modularization, abstraction, design patterns, code organization, testing, documentation, and code reviews, developers can effectively manage complexity and improve the maintainability, readability, scalability, and collaboration of software projects.

---

3. What are ways in which complexity can be managed in JavaScript?

a) Modularization: Breaking down the code into smaller, reusable modules helps manage complexity. Using module systems like CommonJS or ECMAScript modules (ES modules) allows for better organization and encapsulation of code.

g) Documentation: Providing clear and comprehensive documentation, including inline comments, API documentation, and architectural overviews, helps developers understand the codebase and manage its complexity effectively.

b) Abstraction and Encapsulation: Employing principles like abstraction and encapsulation helps hide internal complexities and expose only the necessary interfaces. This improves code readability and maintainability.

---

4. Are there implications of not managing complexity on a small scale?

Yes, there are implications of not managing complexity on a small scale. Even in a small software project, unmanaged complexity can lead to several problems

---

5. List a couple of codified style guide rules, and explain them in detail.
   a) Descriptive names for variables and functions: This helps give other developers reading the code an easier time understanding it. Giving names that describe the value of a variable, what the function does, or even what it returns helps in creating a more readable structure
   b) Do not use abbreviations: Never use abbreviations that are not widely recognized because this can cause a lot of confusion even for you.

---

6. To date, what bug has taken you the longest to fix - why did it take so long?

In the IWA module, the express kitchen web app, the drag feature bug gave me a hard time and still is not clear on it. The main reason for this is a lack of understanding of the core topic, this caused an overwhelming feeling which caused me to get stuck and think that this was not my thing.