DWA_01.3 Knowledge Check_DWA1

---

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

Managing complexity in software is essential for maintainability, readability, debugging, scalability, performance, and risk reduction. It enables developers to create robust, efficient, and adaptable software that is easier to understand, modify, and maintain throughout its lifecycle.

---

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

Non-functional Requirements: Non-functional requirements such as performance, scalability, security, and reliability can add complexity to software development. Ensuring that these requirements are met often involves additional layers of complexity in the design and implementation.
Documentation and Communication: Insufficient or inadequate documentation, as well as poor communication within development teams, can contribute to complexity. When developers lack clear and comprehensive documentation or struggle to communicate effectively, it becomes harder to understand and maintain the software.
Time and Resource Constraints: Tight deadlines and limited resources can lead to shortcuts, technical debt, and rushed decision-making. These factors can introduce complexity that accumulates over time if not properly managed.

---

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

Modularization: Breaking down the JavaScript code into smaller, independent modules can help manage complexity. By organizing related functionality into separate modules, developers can isolate concerns, improve code readability, and make it easier to reason about the system.
Abstraction and Encapsulation: Utilizing abstraction and encapsulation principles can reduce complexity. By hiding implementation details and providing clear, well-defined

interfaces, developers can create higher-level abstractions that simplify usage and understanding of complex functionalities.

---

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

Without managing complexity, even a small codebase can become difficult to understand and maintain over time. This can result in increased time and effort required for debugging, making changes, or adding new features. It may also lead to a higher likelihood of introducing bugs or unintended consequences when making modifications.

---

5. List a couple of codified style guide rules, and explain them in detail.

Naming Conventions:

Rule: Use descriptive and meaningful names for variables, functions, and classes.
Explanation: This rule emphasizes the importance of choosing clear and expressive names for code elements. By using descriptive names, other developers can quickly understand the purpose and functionality of different components, improving code readability and maintainability. For example, using a variable name like totalSales instead of ts makes it easier to comprehend the purpose of the variable.
Indentation and Bracing:

Rule: Use consistent indentation and brace placement for code blocks.
Explanation: Consistent indentation and brace placement enhance code readability and maintainability. This rule ensures that code blocks, such as loops or conditional statements, are properly indented and braces are placed consistently (e.g., on the same line or a new line). Consistent indentation helps visualize the code structure and facilitates understanding. For example:

---

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

Missing a capital letter of a variable name, It took me too long because I kept missing the the variable name until I realised I used camelCase.
