DWA_01.3 Knowledge Check_DWA1

---

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

*Maintainability: Complex software systems are often difficult to understand and modify. By managing complexity, you can make the codebase more maintainable, allowing developers to easily comprehend and make changes to the software over time.

*Debugging and troubleshooting: Complexity can make it harder to identify and fix bugs or issues in the code. By managing complexity, you can reduce the likelihood of introducing bugs and make it easier to identify and resolve them when they do occur.

*Collaboration: Software development is often a collaborative effort involving multiple developers. Managing complexity ensures that the codebase is organized and comprehensible, enabling effective collaboration among team members.

---

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

*Size and scope: The larger and more extensive the software system, the greater the complexity involved in managing it.

*Dependencies: When software relies on numerous external libraries, frameworks, or modules, it can introduce additional complexity in terms of integration, versioning, and compatibility.

*Business rules and requirements: Complex business rules and requirements can add layers of complexity to the software design and implementation.

*Lack of documentation: Insufficient or outdated documentation can make it challenging for developers to understand and work with the software effectively.

---

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

*Modularization: Breaking down the codebase into smaller, modular components helps manage complexity by providing clear boundaries and encapsulation. It allows for easier maintenance, testing, and reuse of code.

*Abstraction and encapsulation: Using abstraction techniques like functions, classes, and modules allows for hiding implementation details and exposing only relevant interfaces. Encapsulation helps manage complexity by reducing dependencies and providing a clear separation of concerns.

*Code organization and structure: Employing consistent and logical code organization practices, such as following a modular directory structure, using meaningful variable and function names, and adhering to design patterns, can improve code readability and maintainability.

*Documentation and comments: Documenting code, writing clear comments, and providing API documentation can help others understand the codebase and reduce complexity.

*Testing and code reviews: Implementing automated tests and conducting regular code reviews can identify and address potential complexity issues before they become more significant problems.

---

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

*Difficulty in understanding and maintaining code: Without managing complexity, the codebase can become convoluted and challenging to comprehend. This leads to difficulties in making changes, fixing bugs, or adding new features.

*Increased risk of bugs and errors: Complex code is prone to introducing bugs, which can be difficult to track down without proper management. This can result in software instability and unreliable behavior.

*Slow development pace: Unmanaged complexity can slow down development as developers spend more time deciphering code and working around convoluted structures. It hampers productivity and efficiency.

---

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

*Rule: "Use meaningful and descriptive variable and function names."

This rule emphasizes the importance of using descriptive names that accurately convey the purpose and functionality of variables and functions. Meaningful names enhance code readability and understanding, making it easier for other developers to work with the codebase.

*Rule: "Avoid deeply nested code structures."

This rule discourages excessive nesting of code blocks, especially in scenarios involving asynchronous operations and callbacks. Deeply nested code can be difficult to follow and understand. It can lead to maintenance issues, code duplication, and potential bugs. Using techniques like Promises, async/await, or modularizing code can help avoid callback hell and improve code readability.

---

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

When I was working on my final IWA19, I had to learn a lot of things about debugging the code in JavaScript, so this is what I can say in short. Debugging a code that does not consol