DWA_01.3 Knowledge Check_DWA1

---

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

Managing complexity allows for a better understanding of the software, making it easier to maintain, debug, and enhance in the future. It also facilitates collaboration among team members by providing a clear and structured system because as software systems become more complex, it becomes increasingly challenging for developers to comprehend the entire system.

---

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

1. Requirements: Complex software often arises from complex or ambiguous requirements. When requirements are unclear or constantly changing, it becomes challenging to design and implement a robust and cohesive solution.

2. Size and Scope: Larger software systems tend to be more complex due to the sheer volume of code, modules, and interactions between components. As the size and scope of a software project increase, so does the complexity of managing and coordinating all the different elements.

3. Interdependencies: When software components or modules depend on each other, changes in one component can have ripple effects throughout the system. The more interdependencies there are, the more complex it becomes to understand, modify, and maintain the software.

4. Technology Stack: The choice of technology stack and frameworks can introduce complexity. Different technologies may have varying levels of maturity, compatibility issues, or steep learning curves. Integrating diverse technologies can also add complexity, especially when they have different programming paradigms or require complex configurations.

5. Integration: Software often needs to integrate with external systems or libraries. This can involve working with different data formats, protocols, or APIs, leading to complexity in handling data transformations, ensuring compatibility, and managing error handling.

---

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

1. Modularization: Break your code into smaller modules or functions that have specific responsibilities. This helps in organizing code and makes it easier to understand and maintain. Modules can be imported and exported using the ES modules syntax or CommonJS modules in Node.js.

2. Abstraction: Use abstraction to hide complex implementation details and expose simpler interfaces. This can be achieved through object-oriented programming concepts like classes, encapsulation, and inheritance. Abstraction helps in reducing the cognitive load by focusing on high-level concepts rather than low-level details.

3. Separation of Concerns: Follow the principle of separating concerns, where each part of your code should have a single responsibility. For example, separate user interface (UI) logic from data manipulation or business logic. This makes it easier to understand and modify specific parts of the code without affecting others.

4. Code Organization: Adopt a consistent code organization approach, such as using meaningful variable and function names, following naming conventions, and organizing files and directories logically. This improves code readability and allows developers to quickly locate and understand different parts of the codebase.

5. Documentation: Document your code using comments, JSDoc annotations, or dedicated documentation tools. Well-documented code helps developers

understand its purpose, usage, and any important considerations. It also enables easier collaboration and maintenance.

6. Testing: Implement automated tests to verify the correctness of your code and catch potential issues early on. Unit tests, integration tests, and end-to-end tests help in identifying and fixing bugs, improving code quality, and providing confidence during refactoring or adding new features.

7. Code Review: Encourage code reviews within your development team. Peer reviews can help identify potential issues, improve code quality, and ensure adherence to best practices. They also provide opportunities for knowledge sharing and learning from each other.

8. Use Design Patterns: Familiarize yourself with common design patterns in JavaScript, such as the Module pattern, Singleton pattern, Observer pattern, and others. Design patterns provide reusable solutions to common programming problems and can help in reducing complexity.

9. Dependency Management: Utilize package managers like npm or Yarn to manage your project dependencies. This ensures that your project has the required libraries and frameworks in a consistent and maintainable manner. Additionally, package managers often provide versioning and update mechanisms to handle dependency conflicts and security vulnerabilities.

10. Refactoring: Regularly review and refactor your codebase to improve its structure, readability, and performance. Refactoring involves restructuring code without changing its external behavior. It helps in simplifying complex code, eliminating duplication, and improving overall code quality.

---

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

Yes, there can be implications of not managing complexity on a small scale. While complexity itself is a natural and inevitable aspect of many systems, failure to effectively manage it can lead to various negative consequences.

---

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

1. Consistent Naming Conventions:

 -Rule: Use consistent naming conventions for variables, functions, classes, and other code entities.
Explanation: Consistent naming conventions enhance code readability and maintainability. When code entities are named consistently, it becomes easier for developers to understand their purpose and usage. It also promotes a unified coding style across the codebase, making collaboration and maintenance more efficient. For example, using camel case (e.g., myVariable) for variable names or Pascal case (e.g., MyFunction) for function names are common naming conventions in many programming languages.

2. Indentation and Formatting:

Rule: Apply consistent indentation and formatting rules to improve code readability.
Explanation: Proper indentation and formatting make code more readable and understandable. By using consistent indentation (e.g., four spaces or a tab), the structure of the code becomes visually clear, making it easier to follow the control flow and identify nested blocks of code. Similarly, consistent formatting guidelines for spacing, line breaks, and brackets help maintain a clean and organized codebase. Following a style guide's indentation and formatting rules also ensures that multiple developers working on the same codebase have a consistent appearance, making collaboration smoother.

---

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

-My own scratch game that I created last year, the game itself is complicated and it always has some glitches and unlike the VSC app, I can’t console via the Scatch Web and get to see the problems.