DWA_01.3 Knowledge Check_DWA1

---

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

It is essential to manage complexity in software because as the complexity increases, so does the difficulty of understanding, maintaining, and debugging the code. Complex software can become harder to modify, extend, and test, leading to higher development costs, increased risk of introducing bugs, and reduced overall productivity. By managing complexity, developers can improve code maintainability, readability, and reusability, resulting in more efficient and robust software systems.

---

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

Size and functionality,
Interdependencies and interactions
Evolving requirements
Concurrency and parallelism
Legacy systems and technical debt
External dependencies

---

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

In JavaScript, complexity can be managed through various techniques:

a. Abstraction and modularization: Breaking down complex systems into smaller, more manageable modules and defining clear abstractions can help reduce overall complexity.

b. Composition over inheritance: By favoring composition, developers can build complex functionality by combining simpler components, resulting in more flexible and maintainable code.

c. Clear code organization: Structuring code in a logical and consistent manner, with proper naming conventions and meaningful comments, enhances readability and reduces cognitive load.

d. Code style and guidelines: Following a standardized code style guide, such as JavaScript Standard Style or Airbnb Styleguide, helps enforce consistent practices across the codebase and improves code clarity and maintainability.

e. Testing and automation: Implementing thorough unit tests, integration tests, and automated build processes can catch issues early on and provide confidence when making changes, reducing the likelihood of introducing complexity.

---

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

Yes, there are implications of not managing complexity even on a small scale. Failure to manage complexity can result in code that is difficult to understand and maintain, leading to increased development time and higher chances of introducing bugs. This can hinder collaboration within the development team and result in decreased overall productivity. Additionally, unmanaged complexity can make it harder to adapt to evolving requirements and can lead to a lack of scalability and flexibility in the software system.

---

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

Codified style guide rules:

a. Variable naming conventions: It is important to use descriptive and meaningful names for variables. Variables should visually look different at a glance, and extra information can be included in the names to provide context and clarity.

b. Object property notation: When accessing nested values within an object, using dot-notation rather than deep nesting improves code readability. It helps to avoid excessively long and cumbersome code expressions.

c. JSDoc comments: JSDoc is a popular way to document JavaScript code. It allows developers to describe the shape of objects, behavior of functions, and provide additional context for better understanding. Using JSDoc comments helps improve code documentation and maintainability.

d. Grouping related code: It is important to group related code together, such as placing related variables, functions, or classes in close proximity. This improves code organization and makes it easier for developers to find and understand related logic.

---

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

I encountered a performance issue on a website that took months to fix. Slow server response times were caused by an unnecessary dependency increasing server load overhead. Initially, I suspected caching, CSS, or JavaScript issues. However, after extensive investigation, I discovered the issue was in the system architecture. By making architectural adjustments, I resolved the problem, improving server response times. This experience emphasized the importance of considering architecture and the need for patience and persistence in fixing complex bugs.
