DWA_01.3 Knowledge Check_DWA1

---

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

Managing complexity in software is essential for ensuring long-term success, maintainability, and sustainability of software projects. It allows developers to build robust, scalable, and reliable software that can adapt to changing requirements and technological advancements.

---

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

Size: The larger a software system is, the more complex it may be. This is because there are more components and features to keep track of, and more interaction between those components.

Functionality: The more functionality a software system has, the more complex it may be. This is because there are more ways for users to interact with the system, and more potential for bugs and errors.

Integration with other systems: A software system that integrates with other systems may be more complex than one that stands alone. This is because there are more ways for the system to interact with other systems, and more potential for bugs and errors.

Quality of the code: Poorly written code can make a software system more complex, as it may be difficult to understand and maintain.

Security: The more security features a software system has, the more complex it may be. This is because there are more ways for the system to be attacked and more ways to protect against those attacks.

Scalability: A software system that is designed to be scalable may be more complex than one that is not. This is because there are more ways for the system to handle increased loads and more ways to ensure that the system can handle those loads.

---

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

Code Reviews: Conduct regular code reviews to identify complex areas and encourage best practices. Fresh perspectives from other developers can often spot potential issues and suggest improvements.

Consistent Coding Style: Enforce a consistent coding style across your project using linters and coding guidelines. This helps make the codebase more readable and maintainable

Refactoring: Regularly refactor your code to improve its structure and maintainability. Refactoring helps remove duplication, simplify logic, and reduce complexity.

By employing these techniques and best practices, developers can effectively manage complexity in JavaScript projects and create more maintainable and efficient codebases.

---

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

To address these implications, it's essential to proactively manage complexity, even on small-scale projects. Applying best practices, following coding standards, refactoring regularly, and collaborating effectively within the team can go a long way in reducing complexity and its associated challenges. By managing complexity early on, developers can build a solid foundation that will support the project's growth and long-term success

---

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

Rule 1: Use Single Quotes for Strings Explanation: In JavaScript, strings can be declared using either single quotes (') or double quotes (""). A common style guide rule is to use single quotes consistently for string declarations.

For example:
// Good
const message = 'Hello, world!';
// Avoid
const message = "Hello, world!";
Using single quotes consistently improves code consistency and readability

Rule 2: Use CamelCase for Variable and Function Names Explanation: CamelCase is a naming convention in which each word in a compound name is capitalized except for the first one, and there are no spaces between words.

For example:
// Good
const firstName = 'John'; const lastName = 'Doe'; function calculateTotalPrice() {
// ...
}

// Avoid
const first_name = 'John'; const last_name = 'Doe'; function calculate_total_price() {
// …
}

Using CamelCase for variable and function names is a widely adopted convention in JavaScript and many other programming languages. It enhances code readability and consistency by providing a clear and standardized way to name variables and functions..

---

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

I spent a week trying to figure out if I was doing something wrong or if I had found a genuine bug in the demo project we were tasked to fix. My gosh it almost broke my spirit. I don't think it would have been possible to fix without further research and investigations of my own, but it ended all working and the problem was fixed.