DWA_01.3 Knowledge Check_DWA1

---

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

- **Maintainability**: Software that is highly complex becomes challenging to understand, modify, and maintain over time. Managing complexity helps keep the codebase clean, organized, and easier to work with, reducing the likelihood of dealing with bugs and allowing other developers to work on your code.

- **Readability**: Complex code can be difficult to read and understand, even for the original author. By managing complexity, developers can try for code that is readable, self-explanatory, and easier to understand by others, improving collaboration and reducing the time required for future development or debugging.

- **Debugging and troubleshooting**: When software systems become overly complex, identifying and resolving bugs or issues becomes more difficult. Simplifying complexity makes it easier to isolate problems, debug code, and improve overall system stability.
-
- **Scalability and performance:** When software becomes too complex, it becomes harder to make it work well and fast. Managing complexity means simplifying the software design so that it's easier to understand and work with. By doing this, we can make the software easier to scale (handle more users or data), improve its performance (make it faster), and use resources like memory and processing power more efficiently. Essentially, managing complexity helps us build software that can grow smoothly, work faster, and use fewer resources.

---

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

- **Poor naming and documentation**: Lack of clear and descriptive naming conventions, as well as insufficient documentation, can add to the complexity by making it difficult to understand the purpose and behavior of different code elements.

- **Lack of modularization**: =When we don't break down the different tasks in the software into smaller parts that can be used again, it can cause problems. These problems include having the same code repeated multiple times, making the software more complicated, and making it hard to understand and update the code.

- **Lack of testing and quality assurance:** Inadequate testing practices and poor quality assurance processes can result in complex, error-prone code that is hard to understand and maintain.

---

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

- **Modularization**: Modularization means breaking down a software system into smaller, independent modules or components. Each module focuses on a specific functionality or task, and these modules can be reused or combined to build the overall software solution. It helps in organizing and structuring the codebase, making it easier to understand, maintain, and update. Additionally, modularization promotes code reuse, reduces code duplication, and improves overall software quality.

- **Code Reviews and Pair Programming**: Encourage code reviews and pair programming sessions with other developers. This allows for collaboration, knowledge sharing, and catching potential complexity issues early on. Fresh eyes and different perspectives can often identify areas of improvement.

- **Code organization and structure**: Follow consistent and logical code organization practices. Use meaningful variable and function names, break down complex logic into smaller functions, and keep related code close together.

- **Testing and automation**: Implement comprehensive testing practices, including unit tests, integration tests, and end-to-end tests. Automated testing helps catch issues early, improves code quality, and reduces the chances of introducing complexity during development.

---

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:

- **Time and effort**: Without managing complexity, development, and maintenance tasks take longer. It becomes challenging to understand and modify the code, leading to increased development time, debugging efforts, and higher chances of introducing new bugs.

- **Code quality and reliability**: Unmanaged complexity can result in poor code quality, making the software less reliable and more prone to bugs

---

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

- **Always include comments for clarity and documentation**: Using comments allows you to have clarity on your code's functionality, intention, or any important details. Comments should be concise, clear, and focused on conveying relevant information.

- **Avoid using global variables**: Try to have local variables instead to avoid naming conflicts which can make debugging difficult.

- **Limit the length of lines and files**: Long lines of code can be challenging to read and understand, especially when they extend beyond the width of the screen. By limiting line length, you improve code readability and make it easier for developers to comprehend the code at a glance.

---

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

In the IWA19 challenge. The search button gave me difficulties when it had to display more books. The code was all over the place and my variables were not properly placed, because of that, I would often make the mistake of declaring the same variable twice.
