Name: Keotshepile_Maje

DWA_01.3 Knowledge Check_DWA1

---

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

Managing complexity in Software makes the work easier for everyone working on the code or project. It creates better code quality, improves maintainability, and enhanced teamwork making the work better for everyone. And instead of wasting time on figuring out the code you take more on the actual solving of the problem.

Readability and Maintainance:

- By managing the complexity of software, the codebase becomes more readable and easier to understand, making it easy to maintain.
    - Clear code structure, modularization, and well-defined interfaces allow developers to easily see the purpose of the code making it easier to debug and fix errors.

Debugging and Troubleshooting:

- The bigger the codebase the more challenging it is to identify and fix issues. When the codebase is well-organized and follows the best practices the better and more efficient it is to debug the code.

Scalability and Performance:

- Complex software may need to handle large-scale data, heavy workloads, or increasing user demands. Proper management of complexity allows for scalable designs and efficient performance.

Collaboration and Teamwork:

- When complexity is managed well the team members can understand the codebase better and collaborate effectively and efficiently. Reduce conflicts and the time it takes to get the work done, making effective teams.

Testing and Quality Assurance:

- By managing complexity, developers can create modular, testable components that are easier to verify. Isolating functionality simplifies the testing process and increases the chances of detecting and fixing defects before they impact the overall system.

Adaptability and Extensibility:

- When the codebase is well-organised it is easier to make changes to the codebase as new requirements and improvements are made available.

User Experience:

- Managing complex help create an intuitive and user-friendly interface. Enhancing user satisfaction and adaption in software.

---

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

- Architecture and design
    - Poorly designed or overly complex architecture of software can make a system hard to understand, maintain and modify.
- Scalability and performance
    - Software that can handle large-scale usage and performance is often very complex.
- Collaboration/team
    - Communication challenges, conflicting priorities and coordination issues can impact the overall complexity of development. This can also be caused by lack of planning and not agreeing in one specific coding style.

---

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

- Split the code into smaller manageable parts and pieces, and according to functionality.
- Take repetitive code or functionality patterns, that look similar and try to abstract them out reducing the total code size.
- When working in a team, agree on the following; styling guidelines, design patterns, and the technology stack so it’s easy to understand each other's code and work.

---

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

The implication of not managing complexity on a small scale can be very problematic in the future as more features get added to the project. This is called Technical Debt. Where poor design, shortcuts, and poor code quality makes the code hard to understand, modify and maintain. And this has the potential to cause bugs in the future and even made the program to crash.

---

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

Style guide for JavaScript using 'Javascript Standard Style'

- Indentation and spacing
    - Use two spaces for each level of indentation => Proper indentation improves code readability, making it easier to understand the structure and flow of code.
- Use of quotes
    - Use single quotes(' ') for strings unless escaping is necessary.
- Writing variables, functions, object properties
    - Use camelCase
- Equality comparison
    - Use strict equality operator (=== and/ !==) instead of loose equality operators (== and !=)
- Naming convection
    - Avoid using single-letter variable names or overly abbreaviatec names
    - Use descriptive and meaningful names.
    - Follow a consistency naming conventions: camelCase.

---

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

When I was working on IWA19 the Final project which was a book library. I had to fix the button that was supposed to bring/display more books when clicked. And what the button click did was add more of the books that are already displayed. My problem took me so much time to solve but eventually got solved.