DWA_01.3 Knowledge Check_DWA1

---

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

Managing complexity in software is important because it makes the software easier and cheaper to maintain, scale, and understand. It reduces the chances of errors, helps new developers get up to speed faster, and allows the software to adapt quickly to changes. In essence, it saves time, money, and effort while improving the overall quality of the software.

---

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

1. Requirements: If we're not sure what the software needs to do, or if those needs keep changing, it gets complicated.

2. Architecture: If the way the software is built is messy or too fancy, it becomes hard to understand.

3. Technology: Using too many different tools and gadgets without understanding them makes things more confusing.

4. Dependencies: When different parts of the software rely too much on each other, it's tough to change one thing without affecting everything else.

5. Code: If the instructions (code) are messy, long, or not clear, it's hard for people to figure out what's going on.

6. Business Rules: Complex rules about how the software should work make it harder to use and change.

7. Data: If the information the software handles is too complicated or messy, it adds to the difficulty.

8. Performance: Trying too hard to make the software run faster can make things more complicated than they need to be.

9. Legacy Systems: Old, outdated parts of the software can make everything else more confusing.

10. Team Issues: Problems within the group working on the software, like not talking enough or not having the right skills, can make things more complicated.

---

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

1. Break into smaller parts: Divide your code into smaller pieces that do specific tasks.

2. Keep it clean: Use clear names for variables and functions, and organize your files neatly.

3. Handle dependencies: Use tools to manage the extra bits your code relies on and keep them up-to-date.

4. Test it: Check your code regularly to catch mistakes early.

5. Write clear instructions: Make sure other people (or even future you!) can understand what your code does.

6. Clean up regularly: Tidy up your code by removing anything unnecessary or making it simpler.

7. Learn from others: Understand common ways to solve problems in JavaScript and use those methods in your own code.

---

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

1. Confusion: Code becomes hard to understand, slowing down work.
2. More Bugs: Complex code tends to have more mistakes.
3. Slower Progress: Development takes longer due to complexity.
4. Costs Increase: More time spent means higher expenses.
5. Technical Debt: Complexity builds up, needing future fixes.
6. Harder Onboarding: Newcomers struggle to understand complex code.
7. Poor User Experience: Complexity can lead to slower and buggier software, upsetting users.

---

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

JavaScript style guides provide a set of conventions and best practices for writing JavaScript code. They cover various aspects such as naming conventions, indentation, spacing, line length, variable declarations, function declarations, and more. Two popular JavaScript style guides are the JavaScript Standard Style and the Airbnb JavaScript Style Guide.

1. JavaScript Standard Style:

- **Principle**: JavaScript Standard Style focuses on simplicity and minimalism. It aims to provide a single, consistent style across all JavaScript projects.

- **Key Features**:
  - No semicolons: JavaScript Standard Style advocates for omitting semicolons at the end of statements unless necessary. It relies on automatic semicolon insertion rules provided by JavaScript engines.
  - Two spaces indentation: It recommends using two spaces for indentation.

    - Single quotes for strings: JavaScript Standard Style prefers using single quotes for string literals.
    - No unused variables: It discourages declaring variables that are not used.

  - **Example**:
    ```javascript
    function greet(name) {
      return 'Hello, ' + name;
    }
    ```

2. Airbnb JavaScript Style Guide:

  - **Principle**: The Airbnb JavaScript Style Guide is a more comprehensive and opinionated style guide developed by Airbnb. It aims to promote code consistency, readability, and maintainability across large codebases.

  - **Key Features**:
    - Semicolons: Airbnb recommends using semicolons at the end of statements.
    - Four spaces indentation: It prefers using four spaces for indentation.
    - Double quotes for strings: Airbnb suggests using double quotes for string literals.
    - Consistent function and variable declarations: Airbnb provides specific guidelines for declaring functions and variables to ensure consistency.
    - ES6 features: Airbnb encourages using ES6 features where appropriate, such as arrow functions, template literals, and destructuring.

  - **Example**:
    ```javascript
    function greet(name) {
      return `Hello, ${name}`;
    }
    ```

Both style guides have their strengths and are widely used in the JavaScript community. The choice between them often depends on the preferences of the development team and the specific requirements of the project. It's essential to adopt a consistent style across the project to improve code readability and maintainability

---

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

According to my final project, the favorite section took long also the theme code and i can not even say where i did go wrong.
