# DWA_01.3 Knowledge Check_DWA1

---

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

Managing complexity in software is crucial to ensure that the code is readable and maintainable. If your code is not structured well it will be increasingly difficult to debug when problems inevitably arise. When your project grows beyond its original scope, it can become an unmanageable nightmare and very difficult to fix and grow.

---

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

A project where the original requirements are evolving can cause complexity since developers need to adapt and modify the software. Under pressure to meet deadlines, quick fixes are implemented over perfect code and this leads to complexity and will be harder to manage if it's not dealt with. Additionally, scaling, which becomes necessary when an application gains popularity can introduce further complexity.

---

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

One approach to managing complexity is to enhance readability by adding JsDoc comments and documentation. Modularising your code into smaller, reusable modules and functions combat complexity. Use clear, descriptive variable names following industry conventions for better understanding. Following style guides like JavaScript Standard or Airbnb ensures clean, universally readable code. Finally, simplifying your code using abstraction can further tackle complexity.

---

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

Neglecting to manage complexity, even in small-scale software projects, can have notable consequences. Readability and maintainability suffer when code lacks proper structure, making debugging a challenging task. As projects grow or evolve, unmanaged complexity can transform into unmanageable nightmares, hindering development, and potentially causing scalability issues. Collaboration may become problematic, and developers might spend more time deciphering and working around poorly structured code. Technical debt accumulates, slowing down development, introducing bugs, and ultimately affecting code quality. It's crucial to address complexity, even in smaller codebases to avoid this.

---

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

The following rules are from the Airbnb style guide:

**Rule 1: Always use 'const' or 'let' to declare variables**

This rule discourages the use of 'var' because var does not have a block-based scope. When a variable declared with 'var' exists outside of a block scope, like an object, it becomes a global variable that is accessible from anywhere in the program. This can be problematic because it can lead to unexpected behaviour, name conflicts and it can complicate the debugging process.

**Rule 2: Avoid single-letter names. Be descriptive with your naming**

Using single-letter variable names can significantly hinder readability and understanding. Developers may waste valuable time trying to decipher the meaning and purpose of the variable, especially during debugging. By using clear and descriptive names it makes your code read like plain language, helping with maintainability over time. Additionally, it can reduce the likelihood of unintended misuse and ensure that fewer errors occur.

---

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

The most challenging bug I've encountered was in IWA15. It involved extracting the last values from multiple arrays and moving them to another array. The problem occurred when one of the arrays ran out of values, causing the function to break. Initially, I didn't take the time to assess the problem and rushed into a solution, which turned out to be overly complex. Eventually, I realised that I had to start from scratch and plan things properly.

---