DWA_01.3 Knowledge Check_DWA1

---

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

Complexity in software can be a real headache. It makes code hard to understand, maintain, and extend. Plus, it invites bugs, decreases productivity, and increases development time and costs. On the flip side, well-managed code is easier to debug, test, and modify. It's more reliable and efficient in the long run.

To sum it up, complexity is a challenge, but by keeping things in check, we can save ourselves from a lot of trouble.

---

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

- Size and Scope: Large projects with numerous components and features can become inherently complex.
- Poor Architecture: A lack of well-defined architectural patterns or adherence to best practices can lead to tangled and complicated code.
- Changing Requirements: Frequent changes in project requirements can introduce uncertainty and make code more complex.
- Legacy Code: Old, poorly documented, or poorly structured code can be difficult to integrate with or modify.
- Inefficient Algorithms: Complex or poorly optimized algorithms can increase code complexity and decrease performance.
- Lack of Documentation: Insufficient comments and documentation can obscure code intent and functionality.
- Inconsistent Coding Styles: Varying coding styles among team members can make code harder to read and understand.

---

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

- Documentation: Clearly documenting code, including comments and API documentation, aids understanding.
- Code Reviews: Conducting code reviews among team members ensures code quality and identifies potential complexity.

---

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

Yes, even on a small scale, neglecting complexity management can have negative consequences. Small-scale projects can grow in complexity over time, making them harder to maintain and extend.

---

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

- Always use const:
  This rule encourages the use of the const keyword to declare variables when their values won't change. By using const, you signal that a variable's value should remain constant, making your code more predictable and reducing the chance of accidental reassignment or mutation.

- Use named function expressions instead of function declarations:
  This rule suggests using named function expressions for defining functions rather than function declarations. Named function expressions make it easier to identify functions in stack traces and improve code organization. They also prevent polluting the global scope with function names, reducing the chance of naming conflicts.

---

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

My biggest bug so far in my coding journey probably in my previous challenge in the IWA for the kitchen express was appending certain css properties or data types to the screen i knew that i had to use append child or something like that but regarding javascript and how incredibly complex it is it’ll take me a while to get a hang of it but in summary it took me about 2 - 3 days to get the append child property bug in my javascript to work i understand its functionality completely just can't remember the syntax or logic