DWA_01.3 Knowledge Check_DWA1

---

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

Technical challenges, Software projects can be complex and difficult due to the technical challenges involved. This can include complex algorithms, database design, and system integration, which can be difficult to manage and test effectively.

---

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

Software engineering and software project management can be complex due to various factors, such as the dynamic nature of software development, changing requirements, technical challenges, team management, budget constraints, and timeline pressures.

---

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

Can be managed by being:
Responsible for drawing the page, which itself is a complex, multi-step process involving calculating styles, updating and compositing layers, and painting to screen.

Responsible for listening to and reacting to events, including events such as scrolling.
Responsible for loading and unloading the page.
Managing media, security, and identity. That's all before any code you write can even execute on that thread, such as:
Manipulating the DOM.
Accessing sensitive APIs, for example, device capabilities, or media/security/identity.

---

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

Yes, complexity will reduce stress and make you feel better. Your value in the job market will rise. You will be harder to replace. You will gain more influence, without it you’re nothing.

---

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

Follow the Style Guide

Every programming language has a style guide that tells you in great detail how to indent your code, where to put spaces and braces, how to name stuff, how to comment—all the good and bad practices.

Create Descriptive Names

Constrained by slow, clunky teletypes, programmers in the past used to contract the names of their variables and routines to save time, keystrokes, ink, and paper. This culture persists in some communities, in the name of backward compatibility; consider C's tongue-twisting wcscspn (wide character string complement span) function. But there's no excuse for this practice in modern code.

Comment and Document

Start every routine you write (function or method) with a comment outlining what the routine does, its parameters, and what it returns, as well as possible errors and exceptions. Summarize in a comment the role of each file and class, the contents of each class field, and the major steps of complex code. Write the comments as you develop the code; if you think you'll add them later, you're kidding yourself

Don't Repeat Yourself

Never copy-and-paste code. Instead, abstract the common parts into a routine or class (or macro, if you must), and use it with appropriate parameters. More broadly, avoid duplicate instances of similar data or code. Keep a definitive version in one place, and let that version drive all other uses.

Check for Errors and Respond to Them

Routines can return with an error indication, or they can raise an exception. Deal with it. Don't assume that a disk will never fill up, your configuration file will always be there, your application will run with the required permissions, memory-allocation requests will always succeed, or that a connection will never time out. Yes, good error-handling is hard to write, and it makes the code longer and less readable. But ignoring errors and exceptions simply sweeps the problem under the carpet, where an unsuspecting end user will inevitably find it one day.

---

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

A whole night!

I was working on Logisim, a software for designing and simulating digital logic circuits, for an assignment in our Computer Architecture course.

We had to implement a 5-stage CPU Pipeline. Well, had it built in a few hours but it wasn’t giving the expected output. Everything was in order. I stayed up all night in the library searching for any wrong connection in the circuit but no luck. Finally, I saw that one of the

black boxes had its name starting with the letter ‘i’ instead of ‘I’. Cursed myself, corrected the error and slept in the library itself!
