DWA_01.3 Knowledge Check_DWA1

---

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

- If one does not it reduces the quality of the software as your code is more bug prone.
- If code is not structured in a concise way that is easy to read, errors can be made by misunderstanding code. Some of these errors can be catastrophic.
- Mid-level bugs can be much harder to find and fix
- Different people can work on one project over a long period of time. If code is not commented or written self explanatory, things can be interpreted incorrectly and errors can be made

---

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

- Programming inherently comes with complexity
- Requirements of projects can change overtime which can cause complexity (Requirements Evolving)
- Technical Debt can occur where a client needs a quick fix to a bug and the developer forgets to go back to the quick fix to document it properly. Technical debt can incur ‘interest’ if not tended to.
- Scaling. Code needs to change as the project grows.
-

---

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

- Taking a look at code styles and using Style Guides
- Adding documentation to your code
- Build the code Modular, things like global constants, functional programming and object oriented programming.
- Use abstraction (build little pieces of software and join them together)
-

---

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

- Yes.
- If the project was to evolve, the complexity that was not managed on a small scale can pile up and cause bugs and errors.
- It can make it more difficult to maintain and update software over time
- It can also lead to poor performance and reliability, which can negatively impact the user experience
- Managing complexity on a small scale requires careful planning and execution as well as ongoing monitoring and maintenance that would develop high quality software that meets the needs of users.

---

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

1) Code Style and Style guides
    - Having the properties of objects on a new line
    - Having indentation as things get nested in a property
    - Using brackets if an if statement is more than a single line
    - Writing global constants in upper snake case
2) Adding Documentation
    - Having comments describing what the code does
    - Having comments describing the code types and shapes
    - Having comments about what the code returns
3) Build the code Modular
    - Make the code easy to reuse (using global constants or functions)
    - Keep related things close together as functions or objects
    - Using Functional Programming (FP)
    - Using Object Oriented Programming (OPP)
4) The use of Abstraction
    - Build little pieces of software then join them together (eg. using ‘export’ and ‘import’)
    - Only allowing the interface to be visible. Eg. Making it easier for the next developer not to need to understand the background work like coercion that goes into a calculation.

---

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

- Probably a spelling error. I overestimated the bug. I didn’t think that my error could be caused by a ‘simple’ spelling error. That made me over analyze my code to look for a more ‘complex’ error.
