DWA_01.3 Knowledge Check_DWA1

---

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

Software/programming can be catastrophic. If something isn't right, it fails, there is no in-between (like domino's). Software crashes more than it works. That's why they say software developers have an unusually high tolerance for catastrophes. Every few minutes, you are likely to be told by the computer that something is wrong.

- **Collaboration purposes**: your team members need to be able to understand the codebase. Clear code enables team members to comprehend each others code.
- **Readability purposes**: complex code can make it hard for developers to comprehend and understand, leading to confusion.
- **Complexity purposes**: software is complex enough, there is no need to make it even more complex. Developers manage complexity of code by breaking down complex problems into smaller manageable components.

---

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

- Programming is complex, even small projects
- Requirements are **always evolving** (as customers use it, the objective is always changing direction)
- **Technical debt**: sometimes you just need to do a quick fix if user needs to use it now and don't have time to comment it and write it out well, to address customers in a timely manner, overtime can lead to pile technical debt
- **Scaling**: code needs to change as product grows (always refactoring code, improving performance)
- Poorly designed software
- Architecture
- Inadequate documentation/ comments

---

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

- Have a consistent code style and formatting

- Use common code style conventions (such as the Airbnb JavaScript Style Guide or the one recommended by your company). This helps especially when collaborating with other developers
- Use clear, descriptive, specific names for functions/variables etc, even if you have to make it long (e.g., const ‘timeAsSeconds’ instead of simply ‘time’)
- Include regular checks: utilize the ‘ throw new error’ statement to regularly check and handle errors in your code
- Enhance code readability by providing descriptive comments using ‘@param’ tags which allow you to specific information about the parameters accepted by your functions
- Comment your code to provide context and explanations for code blocks
- Keep as much hidden from the person using the code. The more info/code you give, the more complex it becomes. For example for parseInt, you just need to know what it does, not how it does it
- Document comments
- Document what the code returns
- Build it more modular (keep things together in modules): keep related code close, it’s easier to reuse.
- Abstraction: build small pieces of software and compose it together

---

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

- Increased errors if its too complex
- If its not managed property it can lead to redundancy of code
- Can be misunderstood easily by coworkers
- Can be costly in the long run since it will require more time to try and understand the code and rework the code

---

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

- Declare variables using const and let
- Have properties of an object be on a new line
- Include indents as things get deeper nested in a property
- Use brackets if a ‘if’ statement is more than a single line
- Global constants: write in upper snake case

---

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

I spent a long time trying to get the books to display on the web page in our last project.
