DWA_01.3 Knowledge Check_DWA1

---

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

It is important to manage complexity to prevent instances where for example a developer goes back to the code they wrote a week ago and finds it hard to understand anything which becomes complicated that it's hard to even know what they were thinking when they wrote it. Another reason why it is important to manage complexity can be seen when other developers start looking at your code which can be a daunting task to understand if the codebase is complex. It is best to code and comment in such a way that other developers learn and understand your own work. Managing complexity can improve the understandability, maintainability, scalability and error detection of the codebase

---

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

Some factors that can create complexity in software are:

- Programming in general is complex. It's not an easy task to program software. It can be complicated and hard to understand.
- Requirements are usually evolving. Most of the time you may not know everything upfront and so as a result of changing direction as you are building the software, it can get complex overtime.
- Technical debt which is described as when developers delay or remove work for the sake of speed. If the problems are not fixed then, it will affect them in the future and will cause existing work to get even worse. The longer the technical debt is present, the more complex the code is going to be.
- Scaling. Code is going to change or be updated as the software scales. As software grows, the size of the codebase will grow too and the larger it becomes, the more complex it becomes to understand, navigate and maintain it.

---

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

Ways in which the complexity can be managed is:

- Making use of a code style and style guides. This can include making variables look visually different, including extra information in variable names, avoiding vague names, using uppercase for global constants.
- Code organization. This means to keep a clear and consistent structure of the code such as grouping related data together or following good naming conventions. This will assist developers in locating code faster.
- Making use of JS Documentation and code commenting. Include comments that explain complex logic or tricky parts of the code. Documenting the types and shapes of the functions can provide a reference to developers and facilitate an understanding.
- Building the code in a modular way. This means to break the code into smaller modules that focus on specific functionality which improves the organization and reusability of the code.
- Making use of abstraction. This can mean building little small pieces of software and then composing them together. Another aspect can be keeping as much code hidden from the user that is using it about what goes on internally to prevent more complexity.

---

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

Yes there are implications in not managing complexity on a small scale which is:

- Difficulty for developers to understand the system which leads to difficulties in maintaining and debugging the software which will result in developers struggling to make changes or updates to the codebase.
- The software will be more prone to errors and bugs as it will be challenging to identify the errors and fix them.
- The maintenance of the software overtime will be challenging and costly. It will be time-consuming and will likely experience more bugs which affects the longevity of the software.

---

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

Some rules regarding variables in code is:

- Always use const or let to declare variables otherwise it will result in global variables. So as to avoid polluting the global namespace.

Example:

Const timeInSeconds = 10;

- Use const or let declaration per variable or assignment. Reason being it will be easier to add new variables and never having to worry about adding a semi-colon for a comma.

Example:
Instead of:
Const time = 20,
PI = 3.14;
fullName = “Jenny Ortega”

Do this:
const time = 20;
const PI = 3.14;
const fullName = “Jenny Ortega”

- Grouping all the const’s and all the let’s. It will be helpful when needing to assign a variable depending on one of the previously assigned variables.

Example:
Instead of:
Let i = 0;
Const time = 50;
Let name = “”;
Const PI = 3.14

Do this:
Let i = 0;
Let name = “”;

Const time = 50;
Const PI = 3.14

- Do not chain variable assignments as this can create implicit global variables.

Example:

Instead of this:
Let a = b = c = 9

Do this:
Let a = 9;
Let b = a;
Let c = a;

- Avoid the use of unary increments and decrements (++, - -). Reason being is the increment and decrement statements are subject to automatic semicolon insertions and can cause silent errors with incrementing or decrementing values within the code and it will also prevent pre-incrementing or pre-decrementing values unintentionally which can cause unexpected behavior in the code.

Example:

Instead of:
Let num = 9;
num++;

Do this:
Let num = 9;
num += 1

---

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

Most of the time the bug that continues to slow me down is syntax errors. These are errors that violate the language syntax rules which can be missing parentheses or semicolons or incorrect variable names. My issue is it can sometimes just be a spelling error where I missed a letter of the variable name and it's just hidden in plain sight. To resolve this I don't skim through code but rather read the code line by line until I find the error which can be a time-consuming process if the codebase is large in size.
