DWA_01.3 Knowledge Check_DWA1

---

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

Code is complex and cumulative by design. It builds upon itself, and because of this, it is impossible to keep track of every moving part. By organising and structuring code well we can make our code more resilient against both its own weight and the weight of what it supports in a larger structure. You can't use or augment something you don't understand, nor would it be easy for someone to do the same to your own work when it inevitably needs to be viewed in relation to another part of the whole. Thus, we develop methods to maintain stability over a tentative structure for as long as we can, while being as well equipped as we can to continue improving and developing our work.

---

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

Programming is simply complex by nature. Our very starting point is to create and navigate a construction that must always evolve and that evolution will cause strain. As things grow in scope more pieces are required for more moving parts leading to more errors, more interplay, and more technicality. This inevitably leaves us with a growing technical debt that if ignored comes to collect at the most inconvenient and damning of times.

---

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

We can manage complexity in JavaScript in several ways.

- Using clearly defined coding styles (commonly the JavaScript standard, or AIRbnb's formatting which was also recommended)
- Adding useful documentation through comments, and through describing in detail what parameters can be taken and where (namely through JS docs).
- Keeping related things together whether that be all global constants at the top of a file or all functions pertaining to a certain task (FP or OOP which is said to be coming later in our learning).
- Build code as modular to make it reusable.

- Use abstraction. We build small sections of software, have them act as self-contained structures, and lend them to larger pieces of software in the hands of coders who don't need to know the intrinsic details of how that smaller piece of code works.

---

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

Poorly managed complexity on a small scale will accrue technical debt that will escalate as you try and expand the project. At a minor level things might not be noticeable, but a disorganised creation will be more prone to errors both from yourself and from anyone else who has the misfortune of needing to use it.

Alternatively, failing to maintain control of your software complexity on a small scale may be indicative of poor performance at a later date once those same problems are scaled up phenomenally in larger projects.

---

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

Use upper snake case for global constants

- It helps improve code readability by giving a clear visual indicator that certain variables are a constant value. It reminds us that these are meant to be accessible from any part of the program and that they are immutable.

Naming variables in obviously different ways

- There are sometimes many variables within a similar concept that need to be defined individually. In those cases it is better to name things in a visually distinct way, or to use longer variable names to ensure clarity. This way neither you nor someone else looking at your work will get confused about what exactly is going on in any particular section of code.

From standardjs.com

- "Use array literals instead of array constructors" - Using [ ] makes it obvious that you're using an array and cleans up unnecessary text.
- "Use a single import statement per module" - Removes unnecessary line clutter that comes from repeatedly referencing an external module for importing. You can import multiple things from the same module in one line.

---

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

I can't remember the longest, but I can remember a relatively frustrating one. I spent a few hours trying to get one of the previous JS projects to work, moved and re-wrote things multiple times, went to the internet, to stack overflow, to GPT, only to discover that I hadn't imported my javascript file correctly the entire time. I then had to figure out what the hell I did in the first place because now that that one obvious problem was solved I had unwittingly created others. I connect my JS files correctly now to say the least. We do learn *some* things.
