DWA_01.3 Knowledge Check_DWA1

---

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

Maintainability - It's one thing to write code that works, but writing code that you or someone else can come back to and use without having to start from scratch in terms of understanding it, that requires managing complexity.

Bugs - When working with poorly structured or styled code, bugs and errors become far more difficult to spot.

---

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

- Software in itself is complex.
- More often than not, during development additional requirements are realized that need to be addressed in order to reach the end goal, and thus the requirements are constantly evolving and adding complexity.
- Unresolved technical debt can result in a lot of unnecessary complexity.
- The larger the scale of a project, the more complex it becomes.

---

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

- Code style - Using common and industry recommended style guides which are aimed at keeping code simple and readable.
- Documentation - Well documented code is far easier to manage and understand when coming back to it after some time.
- Abstraction - Keeping code modular and reusable, and grouping code that serves a similar function together in a separate file helps manage complexity.

---

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

Even if you are the only person working on a project, failing to manage the complexity will result in wasting time going over old code each time you return.

---

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

From the airbnb Javascript style guide:

1. Use dot notation when accessing properties. Dot notation is often preferred because it is easier to read, less verbose, and works better with aggressive JavaScript minimizers.

   ```
   const luke = {
     jedi: true,
     age: 28,
   };

   // bad
   const isJedi = luke['jedi'];

   // good
   const isJedi = luke.jedi;
   ```

2. Group all your consts and then group all your lets. This is helpful when later on you might need to assign a variable depending on one of the previously assigned variables.

   ```
   // bad
   let i, len, dragonball,
       items = getItems(),
       goSportsTeam = true;
   ```

```
// bad
let i;
const items = getItems();
let dragonball;
const goSportsTeam = true;
let len;

// good
const goSportsTeam = true;
const items = getItems();
let dragonball;
let i;
let length;
```

---

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

Whilst working on a side project involving ReactJs, my first attempt at using it, I wanted to use routing to change the contents of a page without needing to load and refresh the entire page each time. I used React Router Library to achieve this, however I immediately encountered a bug where the page URL would update to the correct page URL, but the content would not load unless I manually refresh it. This behavior is the opposite of what React Router is supposed to do, and so I spent the next two days trying to fix the issue. After working through countless forum posts and google searches, I finally decided to look at the React Router Library github issues page, which is where I found a thread discussing the very issue I had encountered as well as a temporary solution that worked. The lesson here being: Always check the official github issues tab first!
