DWA_01.3 Knowledge Check_DWA1

---

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

1. Maintainability, readability and understandability.
2. It is also crucial for debugging purposes.
3. Managing complexity allows for better scalability, because it is then easier to identify and isolate components that need to be scaled.
4. Also because coding is most of the time a collaborative effort, managing the complexity makes it easier to work together and enables a smoother integration of code changes.
5. Code should also be adaptable to change and managing the complexity can benefit so that minimal disruption is caused and managing changes can be done without extensive work.
6. Managing complexity can help code to be sustainable in the long run and provide a solid foundation for future development.

---

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

1. Changes in style.
2. Changes in the requirements/Requirements evolving. Code is dynamic and requires ongoing changes.
3. Frequent changes may lead to lack of consistency.
4. Variables/concepts that look the same or are poorly defined.
5. The scale and size of a system, as the project grows, components, interactions and dependencies increase.
6. Lack of documentation.
7. Technical debt.
8. Poor code quality makes it harder to understand and maintain.

---

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

1. Modularization. Keeping code in small and focused modules that encapsulate specific functionality.

2. Use of classes and Object Oriented Programming OOP.
3. Functional Programming. Pure functions and higher-order functions.
4. Error handling. Catch errors at appropriate levels and logging errors.
5. Documentation. Describe, add types and shapes of usage.
6. Clear and comprehensive documentation of code.
7. Keep code modular by keeping it reusable. Keep related code close to each other.
8. Put code in a “box” as an object or function.
9. Using abstraction to keep little pieces then compose together.

---

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

1. As complexity increases code becomes harder to understand and modify, which reduces maintainability.
2. Increases possibility of bugs.
3. Not managing complexity on the small scale will decrease the reusability of the code.
4. Makes code harder to understand and slows down development.
5. Unmanaged complexity contributes to technical debt.
6. Makes code difficult to understand for other developers that must work on it.
7. If it is not managed scalability is limited as it becomes more challenging to add new features or changing requirements.
8. Not managing complexity can make collaboration ineffective.

---

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

1. Use’ const’ and ‘let’. The use of ‘const’ and ‘let’ is essential for declaring variables. Using ‘const’ makes variables not reassignable. ‘Let’ makes variables reassignable.
2. Prefer arrow functions. Encourages the use of arrow functions for short functions to enhance readability and maintainability.
3. Use 2 spaces for indentation. Defines a consistent indentation style for better alignment and readability.
4. Use semicolons at the end of statements. Ensure predictability and avoid automatic semicolon insertion.
5. Use shorthand syntax for object property assignment when possible. Promotes concise and readable object property assignments.

6. Write meaningful comments and avoid unnecessary comments. Encourages the use of comments for explaining complex code and discourages comments that restate code.

---

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

A week, it was a spelling mistake.
