DWA_04.3 Knowledge Check_DWA4

1. Select three rules from the Airbnb Style Guide that you find **useful** and explain why.

1. **Use const for All of Your References; Avoid var**

Using `const` for all references ensures that the variables cannot be reassigned, which makes the code more predictable and helps prevent bugs that can arise from unintended variable reassignments. When a variable needs to be reassigned, using `let` explicitly indicates that intention. Avoiding `var` eliminates issues related to variable hoisting and scope leakage, leading to clearer and more maintainable code.

2. **Use Arrow Functions Whenever Possible**

Arrow functions provide a more concise syntax and lexically bind the `this` value, which is especially useful in scenarios involving callbacks or nested functions. This avoids common pitfalls with the `this` keyword in traditional function expressions and helps make the code more readable and less error-prone. Additionally, arrow functions improve the consistency of the code by reducing the boilerplate required in function expressions.

3. **Use Template Literals Instead of String Concatenation**

Template literals enhance readability and maintainability by allowing for easier embedding of variables and expressions within strings. This approach not only reduces the risk of errors associated with string concatenation (such as missing spaces or incorrect operator precedence) but also supports multi-line strings without the need for concatenation. Using template literals makes the code cleaner and more intuitive.

---

2. Select three rules from the Airbnb Style Guide that you find **confusing** and explain why.

1. Rule: "Use const for all of your references; avoid using var."
    - Confusion: This rule promotes the use of `const` for all variable declarations and discourages the use of `var`. While it's clear that `const` is preferred for its immutability and block scoping benefits, the rule doesn't explicitly address when to use `let`. This might confuse developers about the scenarios in which `let` is appropriate, particularly when variable reassignment is needed.
2. Rule: "Use object method shorthand."
    - Confusion: The rule suggests using the method shorthand for defining functions in object literals (e.g., `const obj = { method() {} };`). However, it might be confusing for developers to understand when this shorthand should be used over traditional function expressions, especially in cases where the `this` context might be affected or when defining methods on prototypes or classes.
3. Rule: "Use named function expressions instead of function declarations."
    - Confusion: This rule encourages the use of named function expressions rather than function declarations, which can be confusing for several reasons. Firstly, named function expressions are not hoisted, unlike function declarations, which can lead to issues if the function is called

before its definition in the code. Secondly, the naming of function expressions might seem redundant when the function name is not used elsewhere in the code, leading developers to question the practicality of this rule.