DWA_04.3 Knowledge Check_DWA4

---

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

- **Use === and !== over == and !=.**

This rule encourages the use of strict equality operators (=== and !==) instead of loose equality operators (== and !=) for equality comparisons in JavaScript. The use of strict equality operators is useful because it makes your code more explicit and self-explanatory. When you use === and !==, it clearly indicates that you are comparing both value and type, leaving no room for uncertainty . This improves code readability and makes the intent of the comparison more apparent to other developers who read your code.

- **Always use const or let to declare variables. Not doing so will result in global variables. To avoid polluting the global namespace.**

This rule is useful because it helps prevent the unintentional creation of global variables and minimizes the risk of polluting the global namespace. When variables are declared without the const or let keywords, they become implicitly assigned to the global object (e.g., window in browsers or global in Node.js). By using **const** or **let** variables are scoped within their respective blocks or functions, reducing the chance of unintentional global variable declarations.

- **Use single quotes for strings unless needing double quotes**

This rule promotes consistency in string representation. Using single quotes for strings is generally preferred in JavaScript, unless double quotes are necessary within the string itself. It is useful because of its clarity and differentiation. Differentiating between strings and code can be easier when using single quotes. When you use double quotes for strings, it can be less obvious when a quote character is used for string delimiters or when it's part of the string itself. Single quotes provide a clearer visual indication of the string's boundaries.

---

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

- **Never use the Function constructor to create a new function.**

This rule is confusing because of being unfamiliar or unclear as I am not familiar with the potential risks and implications associated with using the Function constructor. The rule itself does not provide an explicit explanation of the security risks or drawbacks associated with using the Function constructor. Without a clear explanation, developers may struggle to understand the rationale behind the rule and its potential impact on their code.

- **Disallow unused variables**

This rule speaks on disabling the declaration of variables that are never used. While eliminating unused variables can help improve code clarity and reduce potential bugs, there are cases where variables might be intentionally declared but not used immediately. For example, variables might be placeholders for future functionality or serve as a reference for debugging purposes. It's essential to strike a balance between adhering to this rule and considering the specific context and requirements of the codebase.

- **Avoid using unary increments and decrements (++, --).**

This rule suggests avoiding the use of the increment (++) and decrement (--) operators and encourages the use of explicit assignment (+= 1 or -= 1) instead. I find this rule confusing because the increment and decrement operators are common and concise ways to modify variables by one. While there can be readability concerns when used excessively or in complex expressions, wise use of ++ and -- can sometimes improve code readability and conciseness.
