DWA_04.3 Knowledge Check_DWA4

---

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

Semicolons: This rule recommends the consistent use of semicolons to terminate statements… because it emphasizes a simple, consistent style in JavaScript code.

Using `let` and `const` instead of `var` in JavaScript makes your code more predictable and easier to read. It prevents common issues, like hoisting bugs, and gives you clear rules for changing values.
`const` makes sure values don't change
`let` allows changes in a controlled way within specific parts of your code.

Instead of using `new Array()` to create an array in JavaScript, it's better to use `[]`. This is because:

Easier to Understand: Using `[]` is shorter and easier to grasp.
Use `[]`, so it keeps your code consistent.
To keep your code clear and prevent potential confusion, stick with `[]` for creating arrays.

---

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

Use `===` and `!==` over `==` and `!=`.**
This rule suggests using strict equality (`===`) and inequality (`!==`) operators instead of loose equality (`==`) and inequality (`!=`) operators.

While strict equality is generally preferred to avoid unexpected type coercion, there are situations where loose equality can be useful. Some developers may find it confusing when to use strict versus loose equality, especially when dealing with values that may have different types.

Use braces with all multi line blocks.
This rule suggests using braces with all multi line blocks. However, in JavaScript, single statements in `if` or loop blocks don't require braces.

Some developers argue that it's a good practice for consistency and to prevent potential issues when adding more statements to a block later. However, others argue that it can make the code less concise and harder to read, especially when the block contains only a few statements.

. Use `// FIXME:` to annotate problems and `// TODO:` to annotate solutions to problems.**

While the intent behind these annotations is clear, some developers may find it challenging to distinguish between problems (`FIXME`) and their proposed solutions (`TODO`). It can be subjective to decide when a comment should be marked as a problem or a solution, and some developers might prefer more descriptive comments for clarity.