DWA_04.3 Knowledge Check_DWA4

---

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

*Rule: "Semicolons" (semi):

This rule enforces the use of semicolons at the end of statements.
Why it's useful: While JavaScript allows you to omit semicolons in some cases, using them consistently helps prevent potential issues with automatic semicolon insertion. Enforcing semicolons ensures code clarity and can prevent subtle bugs.

*Rule: "Variable Declaration" (one-var):

This rule encourages declaring multiple variables with a single var, let, or const statement, rather than separate statements for each variable.
Why it's useful: This rule helps maintain code readability by reducing the number of variable declaration statements. It can make code more concise and easier to manage, particularly when declaring multiple variables of the same type.

*Rule: "Space Before Block Statements" (space-before-blocks):

This rule enforces a space before opening curly braces in block statements (e.g., if statements, loops, and function definitions).
Why it's useful: Adding a space before a block statement improves code readability by making it clear where the block begins. It's a common practice to use this space to visually separate the control statement from the block it controls.
These rules continue to focus on code style and consistency, making your code more readable and maintainable without introducing complex coding decisions.

---

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

*Rule: "Default Exports" (import/export):

This rule suggests using named exports instead of default exports.
Explanation of Confusion: While named exports are generally more explicit and encourage better code organization, there are cases where default exports may be more appropriate, such as exporting a single value or class from a module. The rule's recommendation to avoid default exports altogether can be confusing when there's a legitimate use case for them.

*Rule: "Function Parentheses Style" (arrow-parens):

This rule enforces a consistent style for arrow function parameters, recommending either always using parentheses around parameters or omitting them when there's only one parameter.
Explanation of Confusion: The confusion arises from differing opinions on whether to use parentheses with single parameters in arrow functions. Some developers prefer consistency and always use parentheses, while others appreciate the brevity of omitting them when there's a single parameter. The rule's strictness can lead to debates about which style is better.

*Rule: "No Param Reassignment" (no-param-reassign):

This rule discourages reassigning function parameters.
Explanation of Confusion: While reassigning function parameters can introduce unexpected behavior and make code harder to reason about, there are cases where it may be necessary or clearer. For example, modifying an object property that's passed as a parameter is a common practice. The rule's strict prohibition of parameter reassignment may not always align with real-world scenarios.
