DWA_04.3 Knowledge Check_DWA4

---

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

First Rule - Using const for all of your references; avoid using var.

Explanation - This rule encourages the use of const for variables that do not reassign their value and to avoid using var because using const provides clear intent that a variable is not meant to be reassigned, making the code more readable and reducing the risk of accidental reassignment.

Second Rule - Always use braces for block statements in control structures.

Explanation - Using braces with control structures like if, else, for, while, etc., ensures clarity and helps prevent bugs due to unintended scoping issues. Even though JavaScript allows single-line statements without braces, it is a best practice to include them for readability and maintainability.

Third Rule - Use template literals for string concatenation and formatting.

Explanation - Template literals (enclosed in backticks ``) provide a cleaner and more readable way to concatenate strings and embed expressions. They allow multi-line strings without the need for concatenation or escaping characters. Template literals also make it easier to interpolate variables directly into the string, leading to more maintainable and organized code.

---

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

Rule 1 - I would avoid using the `++` or `--` operators.

Explanation - The Airbnb Style Guide recommends avoiding the use of the increment (`++`) and decrement (`--`) operators, and instead suggests using `+= 1` or `- = 1`. While this rule aims to promote clarity and discourage potential side effects, experienced developers might find the `++` and `--` operators perfectly reasonable and concise in certain situations, especially when working with simple numeric increments or decrements.

Rule 2 - Disallow the use of alert, confirm, and prompt.

Explanation - This rule advises against using alert, confirm, and prompt for user interaction, suggesting that developers use more modern and customizable alternatives, such as modals. While these methods can be considered outdated and may have limitations, they are still functional and straightforward for basic use cases, especially in small-scale applications or quick prototyping

Rule 3 - Disallow the use of Object.prototype builtins.

Explanation: This rule discourages using methods directly from Object.prototype, such as hasOwnProperty, isPrototypeOf, etc., and recommends using safer alternatives like Object.hasOwnProperty. While this rule aims to avoid potential conflicts in case the prototype is modified, some developers may find the direct use of Object.prototype methods more convenient in certain situations

---