DWA_04.3 Knowledge Check_DWA4

---

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

- Prefer const over let and let over var: This rule encourages the use of const for declaring variables by default, followed by let when variable reassignment is required, and discourages the use of var. This practice promotes immutability and reduces the risk of unintentional variable reassignments, leading to more predictable and maintainable code. Using const also signals the intent that a variable's value will not change after initialization, making the code easier to reason about and understand.

- Use template literals instead of concatenation: This rule encourages the use of template literals (also known as template strings) over string concatenation when constructing dynamic strings. Template literals make code more readable and maintainable by allowing you to embed expressions directly within the string using ${expression} syntax. This eliminates the need for clunky concatenation operators and makes it easier to visualize the final string output. Additionally, template literals automatically handle escaping and line breaks, further improving code readability.

- Use the literal syntax for array creation. eslint: no-array-constructor
  ```
  // bad
  const items = new Array();
  // good
  const items = [];
  ```

  Use Array#push instead of direct assignment to add items to an array.
  ```
  const someStack = [];
  // bad
  someStack[someStack.length] = 'abracadabra';
  // good
  someStack.push('abracadabra');
  ```

---

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

- "No-param-reassign" Rule: This rule suggests not to reassign function parameters, as it can lead to unintended side effects. While the intention is to prevent developers from accidentally modifying function arguments, it can be confusing for those who are used to modifying parameters as part of the function's intended behavior. In some cases, parameter reassignment might be necessary for in-place modifications, and adhering to this rule could result in awkward workarounds or less efficient code.

- "No-else-return" Rule: This rule advises against using an else block after a return statement in a function. Instead, it suggests using early returns to reduce nesting. While early returns can lead to cleaner code and easier-to-read functions, some developers might find this rule confusing or challenging to follow in complex scenarios. There could be instances where using an else block after a return provides more clarity and readability, especially when dealing with multiple conditions.

- "Function-paren-newline" Rule: This rule enforces a specific newline style for function arguments, requiring either all arguments to be on the same line or each argument to be on a new line. While consistent code formatting is essential, this rule's strict enforcement might lead to disagreements among developers who have different preferences for function argument formatting. Some developers might prefer keeping all arguments on one line to minimize vertical space, while others might prefer placing each argument on a new line for improved readability.
