

# DWA\_04.3 Knowledge Check\_DWA4

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

1.

- [15.1](#) Use `===` and `!==` over `==` and `!=`. eslint: [eqeqeq](#)

- I find this rule useful because “===” and “!==” are considered to be type-safe/strict equality operators, so we can ensure that all equality expressions are evaluated to more predictable/accurate outcomes.

- [15.6](#) Ternaries should not be nested and generally be single line expressions. eslint: [no-nested-ternary](#)

```
// bad
const foo = maybe1 > maybe2
  ? "bar"
  : value1 > value2 ? "baz" : null;

// split into 2 separated ternary expressions
const maybeNull = value1 > value2 ? 'baz' : null;

// better
const foo = maybe1 > maybe2
  ? 'bar'
  : maybeNull;

// best
const foo = maybe1 > maybe2 ? 'bar' : maybeNull;
```

2.

- This rule is useful because it improves code readability and makes the code a lot easier to understand. Nested ternaries can also increase the risk of errors if not used carefully.

- [11.1](#) Don't use iterators. Prefer JavaScript's higher-order functions instead of loops like `for-in` or `for-of`. eslint: [no-iterator](#) [no-restricted-syntax](#)

Why? This enforces our immutable rule. Dealing with pure functions that return values is easier to reason about than side effects.

Use `map()` / `every()` / `filter()` / `find()` / `findIndex()` / `reduce()` / `some()` / ... to iterate over arrays, and `Object.keys()` / `Object.values()` / `Object.entries()` to produce arrays so you can iterate over objects.

3.

- Higher-order functions like `map()`, `filter()`, and `reduce()` are more expressive and easier to understand at a glance. They clearly convey the operation being performed on the array.

- These functions return a new array and do not mutate the original array. This is important in functional programming and can prevent bugs related to data mutation.
  - They abstract away the process of iteration, allowing you to focus on the operation being performed on each element.
- 

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

- [8.6](#) Enforce the location of arrow function bodies with implicit returns. eslint: [implicit-arrow-linebreak](#)

```
// bad
(foo) =>
  bar;

(foo) =>
  (bar);

// good
(foo) => bar;
(foo) => (bar);
(foo) => (
  bar
)
```

1.

- I find this confusing because based on the examples given, I can not really see the major difference.
-