# DWA\_04.3 Knowledge Check\_DWA4

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

1. Using 'const' for constants and 'let' for variables helps to avoid accidental reassignments. It makes it clear which variables are intended to be immutable and which are meant to change. This practice enhances readability and maintainability by signaling the intent of each variable at a glance. It helps prevent bugs by ensuring that values meant to remain constant are not accidentally changed.
2. Arrow functions provide a more concise syntax compared to traditional function expressions. They also inherit the 'this' value from their surrounding scope, which can prevent common bugs related to the 'this' keyword. Arrow functions reduce boilerplate code, making functions easier to write and read.
3. Template literals provide a cleaner and more readable way to include variables and expressions within strings. This reduces the risk of errors that can occur with complex concatenations.  By using template literals, the code becomes more readable and easier to understand, especially when constructing strings that include multiple variables or expressions. Template literals also allow for easy creation of multiline strings without the need for escape characters.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

1. Using the unary increment (++) and decrement (--) operators leads to me being confused and having readability issues.
2. Understanding the difference between default and named exports can be challenging.
3. Functions like 'map()', 'forEach()', 'filter)', and 'reduce()', this is why I stick with the 'for' loop.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_