DWA_04.3 Knowledge Check_DWA4

---

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

- **Variables** - Use **const** and **let** for declaring variables instead of using **var**, **const** and **let** are block-scoped and **var** is function-scoped. This is useful because by using **const** and **let**, you have block-scoped variables which help prevent accidental reassignments and enhance code clarity.

- **Arrow Functions -** use arrow functions as they provide a more concise syntax for writing functions in JavaScript. They automatically bind the **'this'** context, avoid the need for a **'return'** statement and **'function'** keyword, and allow for the implicit return of single expressions. Using arrow functions can lead to cleaner and more readable code.

- **Naming Convention** - Use descriptive variable and function names. Clear and concise naming enhances code readability and makes it easier for other developers to understand the purpose and functionality of different components in the codebase.

---

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

- **Destructuring** - Use destructuring consistently, this rule encourages using destructuring assignment when extracting values from objects or arrays. While destructuring can improve code readability and expressiveness, the rule is confusing when it comes to nested structures or when dealing with a single value extraction. It requires additional lines of code that might be complex destructuring patterns, which makes the code less straightforward and harder to understand.

- **Modules** - Use ES modules (‘**import’** and ‘**export’**) over commonJS modules (‘**require’** and ‘**module.exports’**). Using complex module structures is confusing for me because as the projects grow larger and more complex, managing the dependencies and structure of modules can become challenging. Also understanding how to effectively structure modules in real-world scenarios with these multiple levels of dependencies is overwhelming.

- **Unused Variables** (‘**no-unused-vars’**) - Variables that are declared and not used anywhere in the code are most likely an error due to incomplete refactoring. Such variables take up space in the code and can lead to confusion by readers. This helps identify potential issues and promotes cleaner code, it can be confusing in certain situations, for example when defining functions that are used as callbacks or when implementing placeholder variables for future use, this rule may incorrectly mark them as unused variables.
