DWA_04.3 Knowledge Check_DWA4

---

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

#Rule 1: Avoid using var and prefer using const or let
Explanation: In JavaScript, it's recommended to use 'const' for variables whose value doesn't change,
and 'let' for variables whose value can change. This helps improve code readability and avoids accidental reassignments.

```
const PI = 3.14159; // Use 'const' for constants
let radius = 5; // Use 'let' for variables that may change
let area = PI * radius * radius;
```

Rule 2: Prefer template literals over concatenation
Explanation: Template literals (also known as template strings) provide a more readable and convenient way
to create strings that include variables or expressions. It avoids the clutter of concatenation with '+'.

```
 let name = "John";
let age = 30;
let message = `My name is ${name} and I am ${age} years old.`;
```

 # Rule 3: Use arrow functions instead of function expressions
Explanation: Arrow functions offer a concise syntax and preserve the context of 'this'.
They are especially useful for defining short, one-line functions and avoiding issues with 'this' scoping.

```
 // Function expression function add(a, b) { return a + b; }

// Arrow function const add = (a, b) => a + b;
```

---

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

Rule 1: Avoid using the 'arguments' object
Explanation: The Airbnb Style Guide advises against using the 'arguments' object in favor of using rest parameters (ES6).
However, it might be confusing for some developers who are accustomed to using 'arguments' in traditional function declarations.

```
// Using 'arguments' object (confusing)
function sum() {
let total = 0;
for (let i = 0; i < arguments.length; i++) {
total += arguments[i];
   }
return total;
}

 // Using rest parameters (preferred)
function sum(...numbers) {
return numbers.reduce((acc, val) => acc + val, 0);
}
```

Rule 2: Avoid using the 'bind' method for performance reasons
Explanation: The Airbnb Style Guide recommends avoiding the use of 'bind' to create a new function with a specific 'this' context.
The main concern is performance, as creating a bound function has a slight overhead compared to other alternatives like arrow functions.

```
// Using 'bind' (confusing)
const person = {
name: "Alice",
greet: function () {
console.log(`Hello, my name is ${this.name}.`);
  },
 };

const boundGreet = person.greet.bind(person);
boundGreet();

// Using arrow function (preferred)
const person = {
```

```
name: "Alice",
greet: function () {
console.log(`Hello, my name is ${this.name}.`);
  },
};

const arrowGreet = () => person.greet();
arrowGreet();

# Rule 3: Avoid using the 'with' statement
# Explanation: The 'with' statement is discouraged in the Airbnb Style Guide due to its potential to create ambiguous code.
# It introduces a new scope that can lead to variable collisions and make it hard to determine where certain variables are defined.

// Using 'with' (confusing)
const person = {
name: "Alice",
age: 30,
};

function printPerson() {
with (person) {
console.log(name, age);
  }
}

printPerson();

// Without 'with' (preferred)
const person = {
name: "Alice",
age: 30,
};

function printPerson() {
    console.log(person.name, person.age);
}
```

```
printPerson();
```