DWA_04.3 Knowledge Check_DWA4

---

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

1. **RULE**

Types

- 1.1 Primitives: When you access a primitive type you work directly on its value.

- object
- array
- function

```
const foo = [1, 2];
const bar = foo;

bar[0] = 9;

console.log(foo[0], bar[0]); // => 9, 9
```

This helps me to know how to mutate a variable's value even if I declared it using a const.

---

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

1.2 TYPES: Complex: When you access a complex type you work on a reference to its value.

- object

- array
- function

```
const foo = [1, 2];
const bar = foo;
bar[0] = 9;
console.log(foo[0], bar[0]); // => 9, 9
```

What is the difference between a primitive type and a complex type?

---

2. RULE

const item = new Object() // bad because it is a constructor function

Const item = {} // Good

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

3.3 Use object method shorthand.

```
// bad why is this bad
const atom = {
  value: 1,

  addValue: function (value) {
    return atom.value + value;
  },
```

```
};

const atom = {
  // Good

const atom = {
  value: 1,

  addValue(value) {
    return atom.value + value;
  },
};
```

3. RULE

4.3 Use array spreads ... to copy arrays.

```
// bad
const len = items.length;
const itemsCopy = [];
let i;

for (i = 0; i < len; i += 1) {
  itemsCopy[i] = items[i];
}

// good
const itemsCopy = [...items];
```

Because it provides simplicity and readability.

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

4.7 Use return statements in array method callbacks.
It's ok to omit the return if the function body

consists of a single statement returning an expression without side effects, following

```
// Good
[1, 2, 3].map((x) => x + 1);

// bad - no returned value means `acc` becomes undefined after the first iteration
[[0, 1], [2, 3], [4, 5]].reduce((acc, item, index) => {
  const flatten = acc.concat(item);
});
```