DWA_04.3 Knowledge Check_DWA4

---

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

1. **Objects (3.6) Only quote properties that are invalid identifiers**

   Example:

```
// bad
const bad = {
  'foo': 3,
  'bar': 4,
  'data-blah': 5,
};

// good
const good = {
  foo: 3,
  bar: 4,
  'data-blah': 5,
};
```

Why is it useful?

Single worded properties will still be valid with quotes, but it makes more sense to only have object properties with more complex naming (like ones that have dashes) be put in quotes for readability.

2. **Functions (7.1) Use named function expressions instead of function declarations**

   Example:

```
// bad
function foo() {
  // ...
}

// bad
const foo = function () {
  // ...
};

// good
// lexical name distinguished from the variable-referenced invocation(s)
const short = function longUniqueMoreDescriptiveLexicalFoo() {
  // ...
};
```

Why is it useful?

With the use of function declarations, one is allowed to call the function before it's defined. This makes the code hard to maintain because the logic and flow is affected. To avoid this, functions should rather be placed in their own variable expression to better manage possible complexity.

3. **Modules (10.5) Do not export mutable bindings**

   Example:

```
// bad
let foo = 3;
export { foo };

// good
const foo = 3;
export { foo };
```

Why is it useful?

If references mutate, or are allowed to mutate in one file, it affects how the values are consumed in the file it is being exported to. Therefore, if it can be avoided, only constant references are recommended to be exported.

---

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

1. **Arrays (4.3) Use array spreads […] to copy arrays.**
   Example**:**

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

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

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

Why is it confusing?

Why is using the fully written code not better if it allows the reader to better understand the transfer of the content of one array to another? Using array spreads are less detailed.

2. **Hoisting (14.2) Anonymous function expressions house their variable name, but not the function assignment**

   Example**:**

```
function example() {
  console.log(anonymous); // => undefined

  anonymous(); // => TypeError anonymous is not a function

  var anonymous = function () {
    console.log('anonymous function expression');
  };
}
```

Why is it confusing?

- Why is this allowed? The result of the function defined below returns ‘is not a function’ when it is called above.
- Is the code above just used to define a point?
- Is it returning an error because it’s technically a variable holding a function (and didn’t start out as an initialised function) or is it simply because the callback is before the anon function?

3. **Functions (7.2) Wrap immediately invoked function expressions in parentheses**

   Example:

```
(function () {
  console.log('Welcome to the Internet. Please follow me.');
}());
```

Why is it confusing?

Additional brackets affect readability and may be unnecessary, since all functions technically run immediately when they are invoked

---