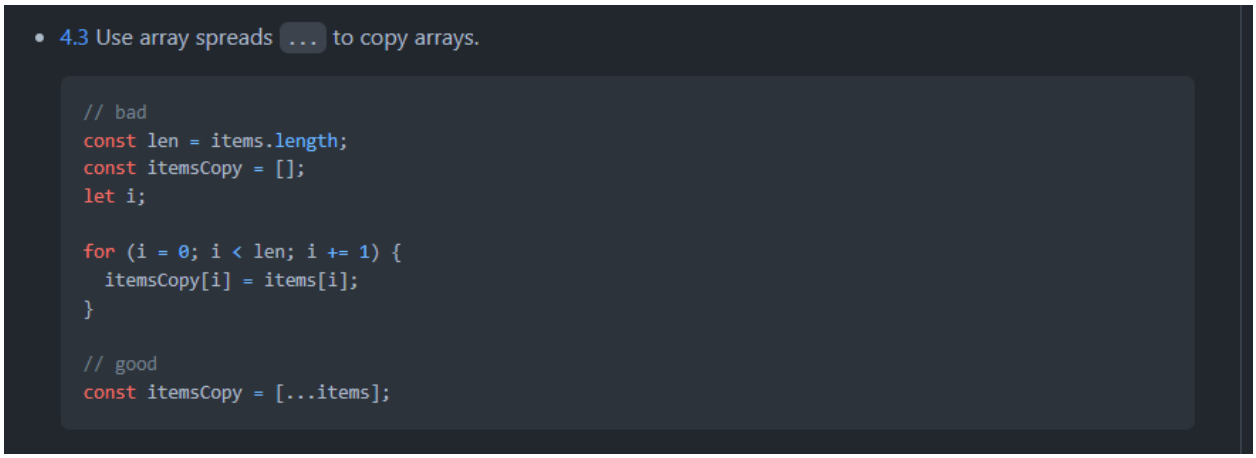
DWA_04.3 Knowledge Check_DWA4

---

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

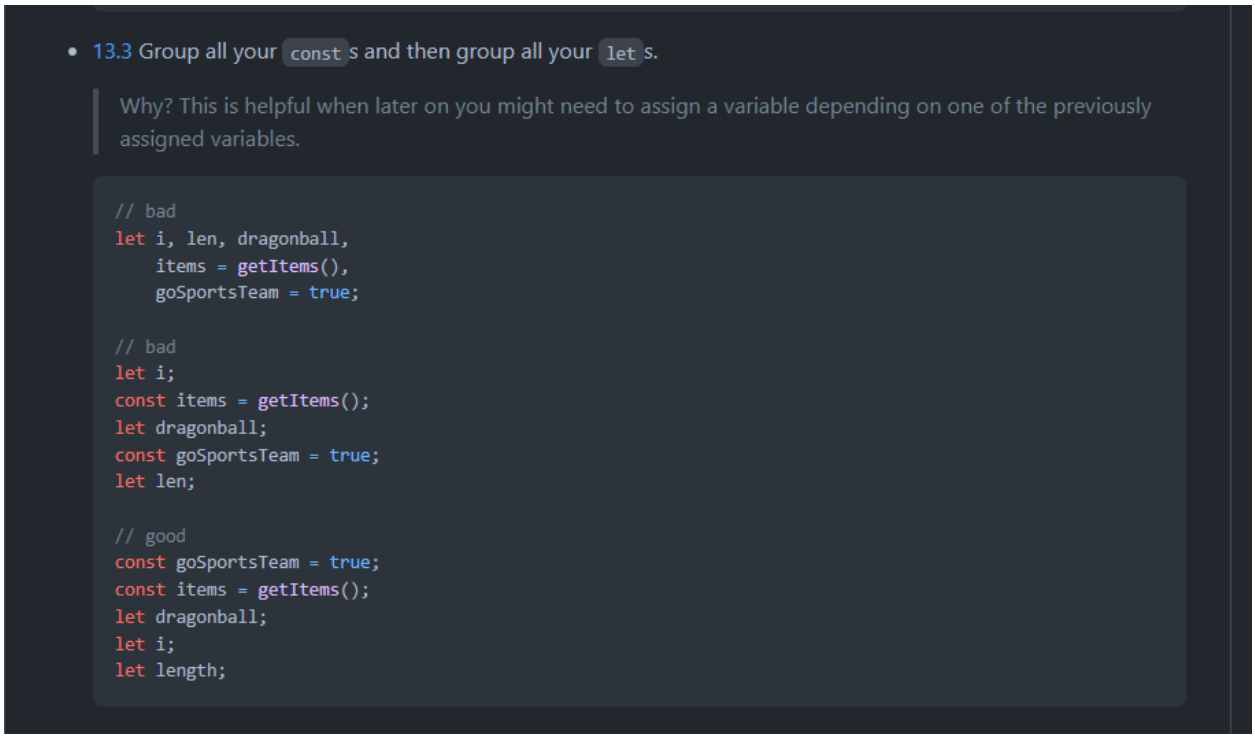
    1. I find this rule to be useful because it shortens the code needed to create a deep copy of an array or object.



- 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];
```

    2.

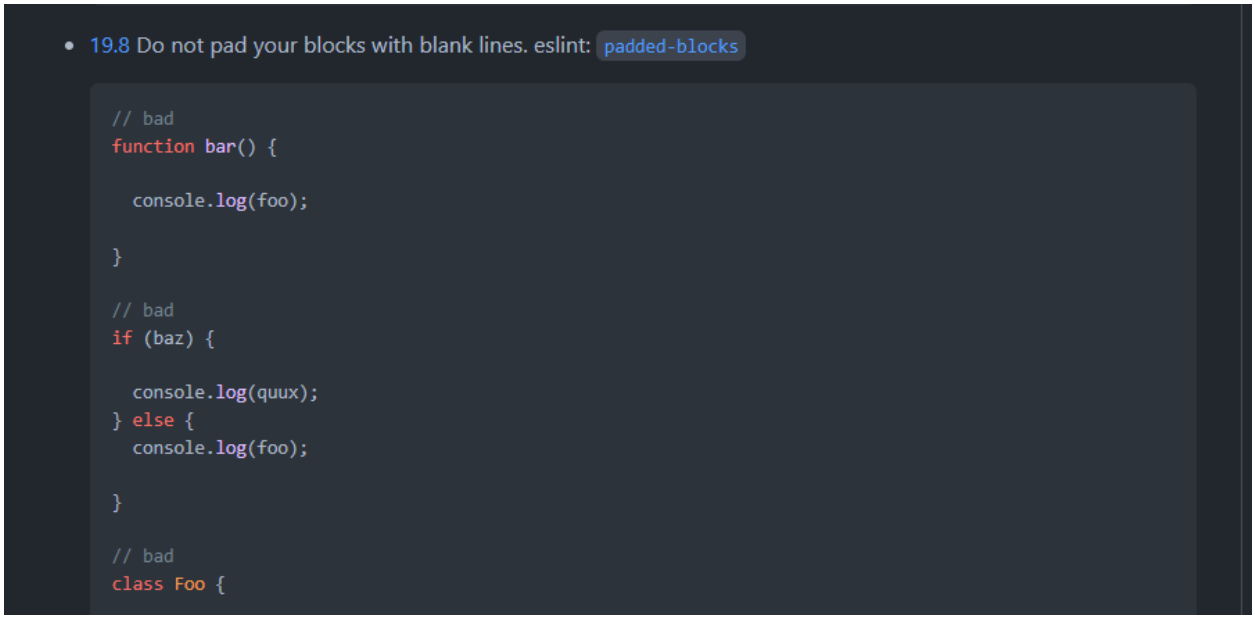


- 13.3 Group all your `const`s and then group all your `let`s.

> Why? This is helpful when later on you might need to assign a variable depending on one of the previously assigned variables.

```
// bad
let i, len, dragonball,
    items = getItems(),
    goSportsTeam = true;

// bad
let i;
const items = getItems();
let dragonball;
const goSportsTeam = true;
let len;

// good
const goSportsTeam = true;
const items = getItems();
let dragonball;
let i;
let length;
```

3. kfjjkfg



- 19.8 Do not pad your blocks with blank lines. eslint: `padded-blocks`

```
// bad
function bar() {

  console.log(foo);

}

// bad
if (baz) {

  console.log(quux);
} else {
  console.log(foo);

}

// bad
class Foo {
```

---

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

1. jfhjfhdl

README.md

```
// good
const foo = [1, 2, 3];
console.log(foo[0]);
```

- 19.12 Add spaces inside curly braces. eslint: `object-curly-spacing`

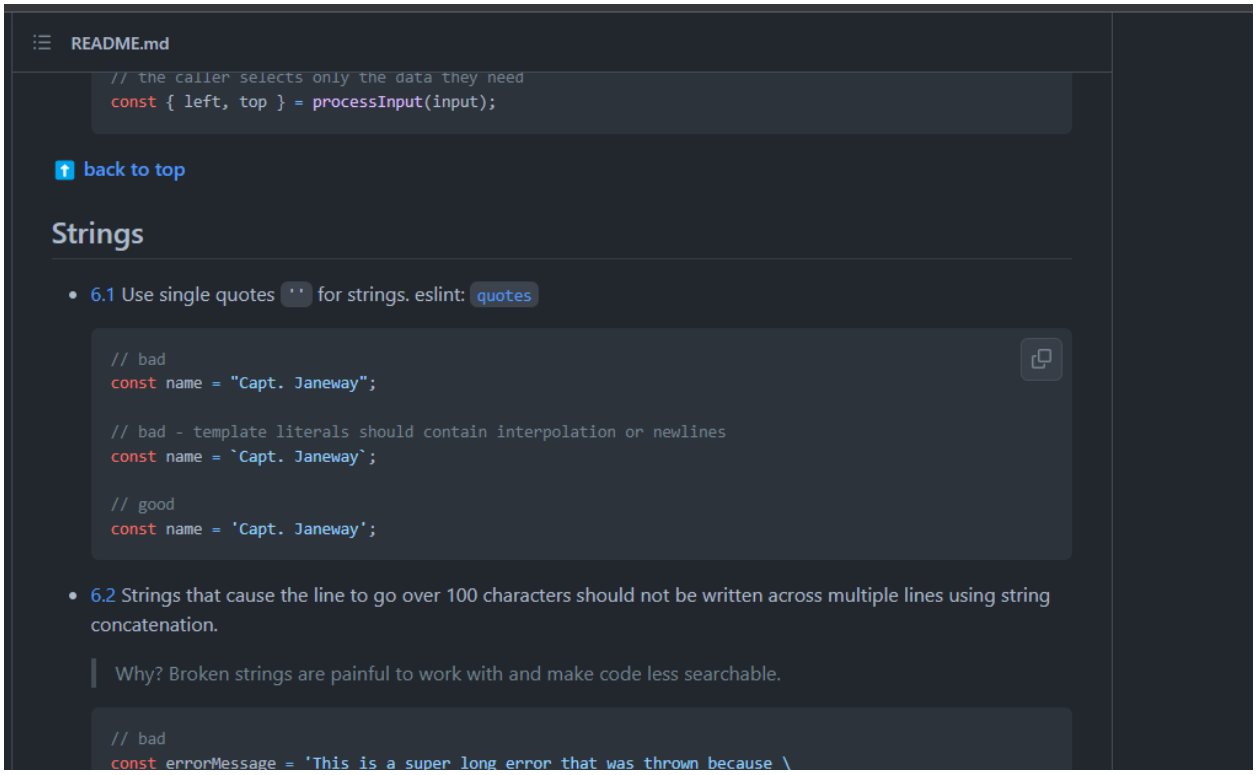
```
// bad
const foo = {clark: 'kent'};

// good
const foo = { clark: 'kent' };
```

- 19.13 Avoid having lines of code that are longer than 100 characters (including whitespace). Note: per above, long strings are exempt from this rule, and should not be broken up. eslint: `max-len`

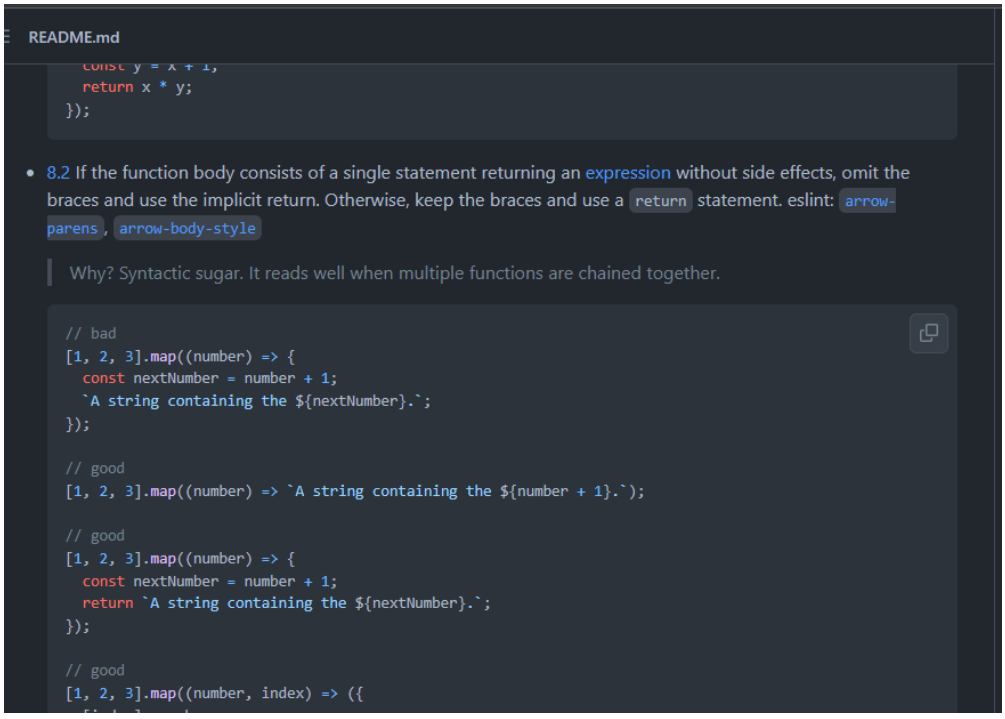
> Why? This ensures readability and maintainability.

```
// bad
const foo = jsonData && jsonData.foo && jsonData.foo.bar && jsonData.foo.bar.baz && jsonData.foo.bar.baz.

// bad
$.ajax({ method: 'POST', url: 'https://airbnb.com/', data: { name: 'John' } }).done(() => console.log('Co

// good
const foo = jsonData
```

2. fghfhdg


README.md
// the caller selects only the data they need
const { left, top } = processInput(input);
back to top
Strings
6.1 Use single quotes '' for strings. eslint: quotes
// bad
const name = "Capt. Janeway";
// bad - template literals should contain interpolation or newlines
const name = `Capt. Janeway`;
// good
const name = 'Capt. Janeway';
6.2 Strings that cause the line to go over 100 characters should not be written across multiple lines using string concatenation.
Why? Broken strings are painful to work with and make code less searchable.
// bad
const errorMessage = 'This is a super long error that was thrown because \

3. gjyfjj


README.md
return x * y;
});
8.2 If the function body consists of a single statement returning an expression without side effects, omit the braces and use the implicit return. Otherwise, keep the braces and use a return statement. eslint: arrow-parens, arrow-body-style
Why? Syntactic sugar. It reads well when multiple functions are chained together.
// bad
[1, 2, 3].map((number) => {
const nextNumber = number + 1;
`A string containing the ${nextNumber}.`;
});
// good
[1, 2, 3].map((number) => `A string containing the ${number + 1}.`);
// good
[1, 2, 3].map((number) => {
const nextNumber = number + 1;
return `A string containing the ${nextNumber}.`;
});
// good
[1, 2, 3].map((number, index) => ({