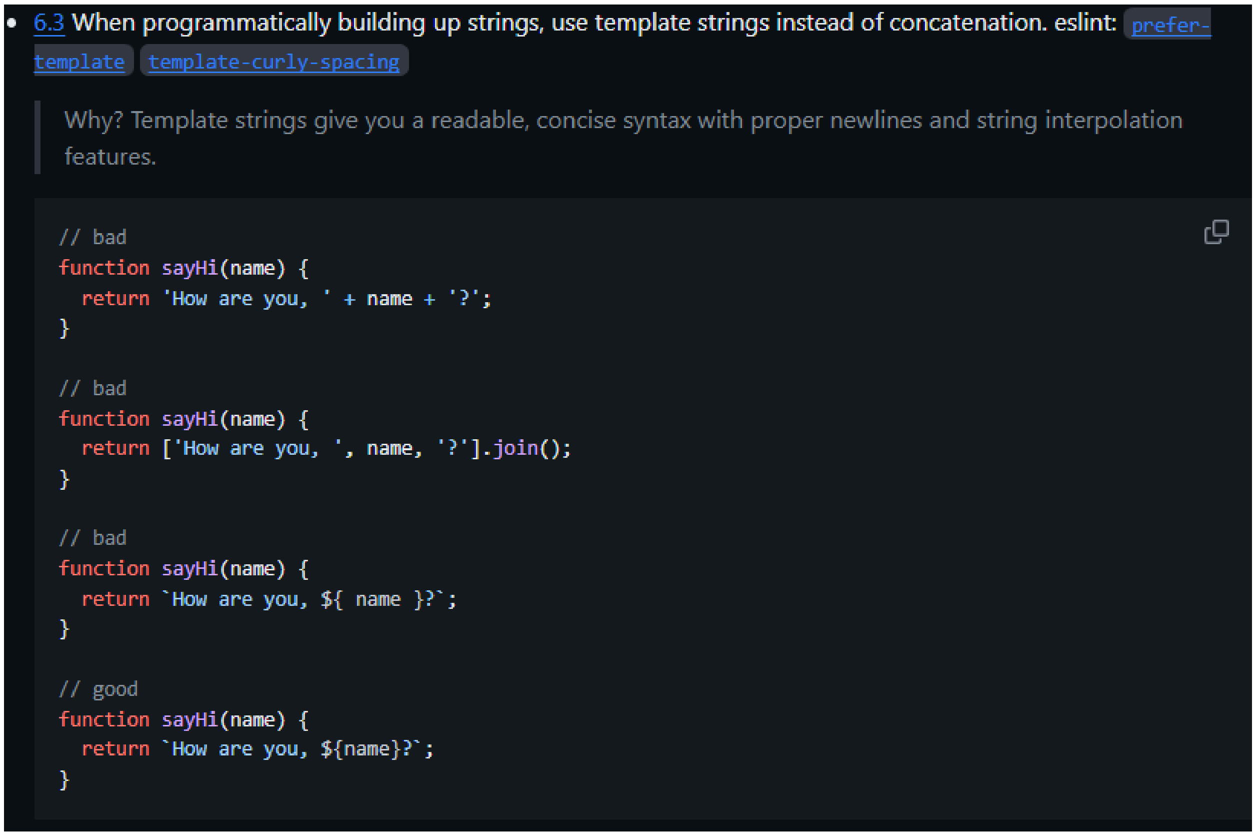
DWA_04.3 Knowledge Check_DWA4

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

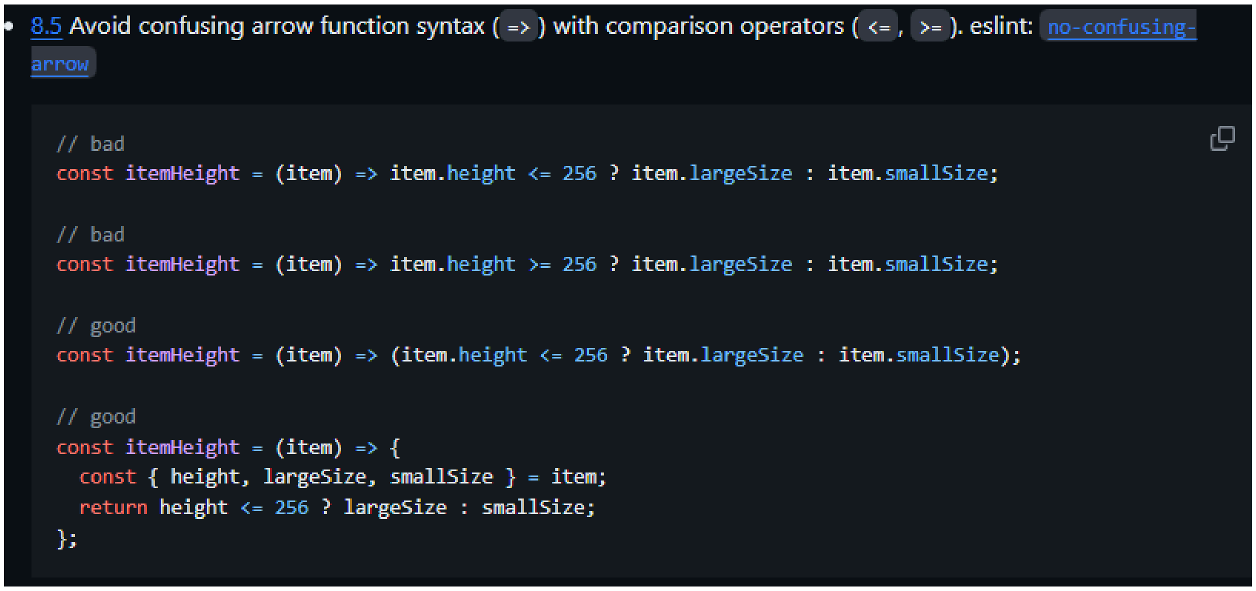


- 6.3 When programmatically building up strings, use template strings instead of concatenation. eslint: `prefer-template` `template-curly-spacing`

  > Why? Template strings give you a readable, concise syntax with proper newlines and string interpolation features.

```
// bad
function sayHi(name) {
  return 'How are you, ' + name + '?';
}

// bad
function sayHi(name) {
  return ['How are you, ', name, '?'].join();
}

// bad
function sayHi(name) {
  return `How are you, ${ name }?`;
}

// good
function sayHi(name) {
  return `How are you, ${name}?`;
}
```

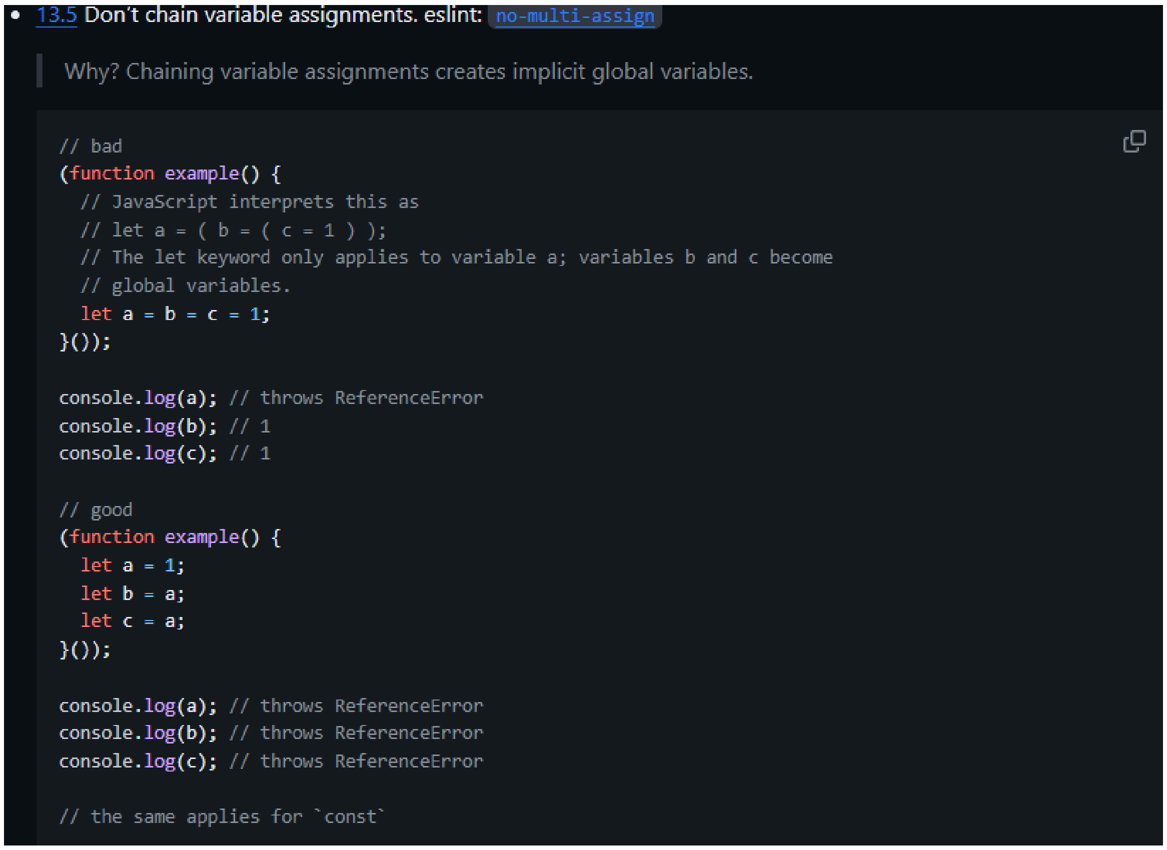
Concatenating strings can be tiresome to type out and are difficult to read. Template literals do a great job of keeping long strings readable.



- 8.5 Avoid confusing arrow function syntax (`=>`) with comparison operators (`<=`, `>=`). eslint: `no-confusing-arrow`

```
// bad
const itemHeight = (item) => item.height <= 256 ? item.largeSize : item.smallSize;

// bad
const itemHeight = (item) => item.height >= 256 ? item.largeSize : item.smallSize;

// good
const itemHeight = (item) => (item.height <= 256 ? item.largeSize : item.smallSize);

// good
const itemHeight = (item) => {
  const { height, largeSize, smallSize } = item;
  return height <= 256 ? largeSize : smallSize;
};
```

Useful once again for readability and easy understanding.



- 13.5 Don't chain variable assignments. eslint: `no-multi-assign`

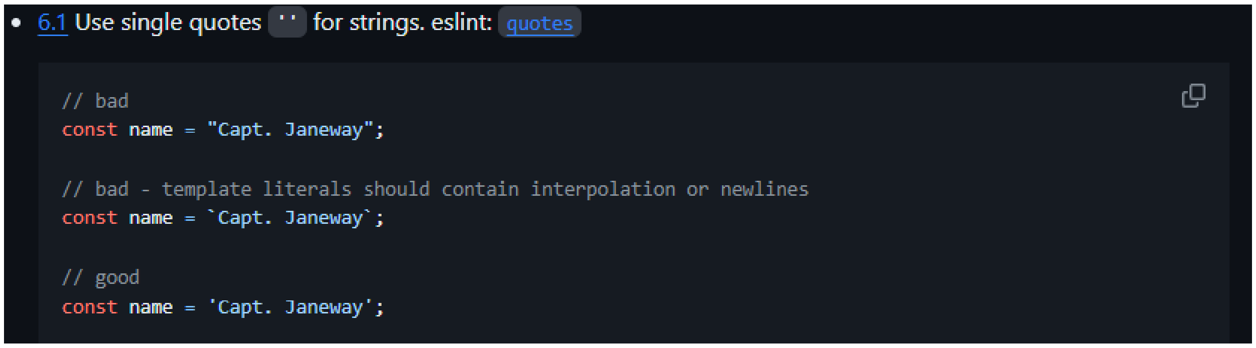
> Why? Chaining variable assignments creates implicit global variables.

```
// bad
(function example() {
  // JavaScript interprets this as
  // let a = ( b = ( c = 1 ) );
  // The let keyword only applies to variable a; variables b and c become
  // global variables.
  let a = b = c = 1;
}());

console.log(a); // throws ReferenceError
console.log(b); // 1
console.log(c); // 1

// good
(function example() {
  let a = 1;
  let b = a;
  let c = a;
}());

console.log(a); // throws ReferenceError
console.log(b); // throws ReferenceError
console.log(c); // throws ReferenceError

// the same applies for `const`
```

This is an excellent rule to avoid accidentally creating global variables, as well as improves readability a great deal.

---

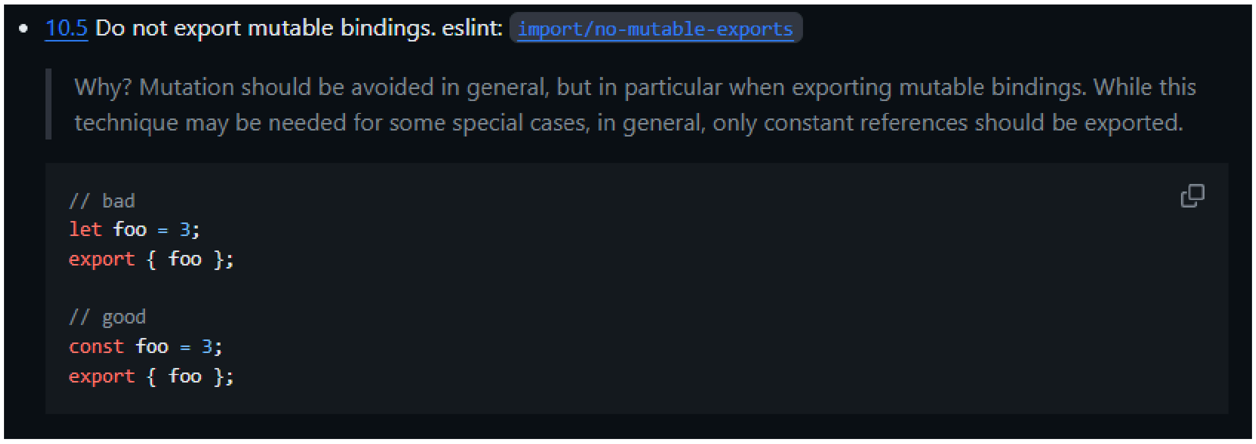
2. Select three rules from the Airbnb Style Guide that you find **confusing** and explain why.



- 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';
```

I haven't really come across a good reason to use single quotes over double, other than personal preference. As long as one is consistent I see no reason why it matters.

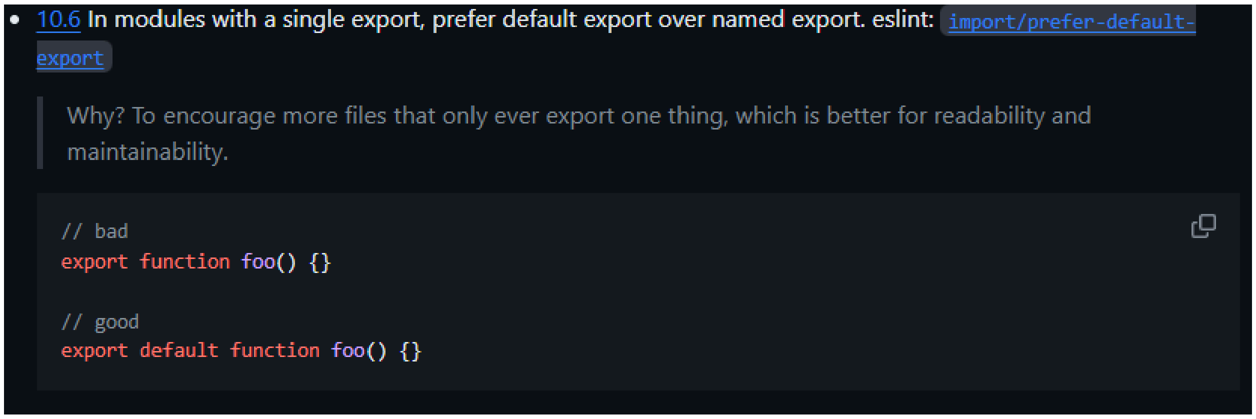


- 10.5 Do not export mutable bindings. eslint: `import/no-mutable-exports`

> Why? Mutation should be avoided in general, but in particular when exporting mutable bindings. While this technique may be needed for some special cases, in general, only constant references should be exported.

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

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

While I understand the reasoning behind this rule, I've had a hard time implementing it as I've had two recent cases where it was absolutely necessary to export a mutable variable. I feel that if the variable is well documented/commented as being mutable, it should be fine to export without causing too much confusion.



- 10.6 In modules with a single export, prefer default export over named export. eslint: `import/prefer-default-export`

> Why? To encourage more files that only ever export one thing, which is better for readability and maintainability.

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

// good
export default function foo() {}
```

I found this rule a bit arbitrary as it has no direct influence on functionality or readability from what I can see.