JavaScript Coding Standards

When comparing use === instead of ==

This is important due to JavaScript being a dynamic language so using == might give you unexpected results due to it allowing the type to be different.

Never use var use let instead

A simple one, using let will help with any scoping issues var causes in JavaScript.

Use const instead of let

This stops developers trying to changing things they shouldn't and really helps with readability.

Always use semicolons (;)

Although this is optional in JavaScript as semicolons are not needed as statement terminators like other languages. Using a ;really helps to keep the code consistent and a great for statement separators.

Naming conventions in JavaScript

- let should be camelCase
- const if at the top of a file use upper case snake case. MY_CONST . If not at the top of the file use camelCase
- class should be PascalCasing. MyClass
- functions should be camelCase . myFunction

Use template literals when contacting strings

Template literals are string literals allowing embedded expressions.

Fail:

```
let fullName = firstName + " " + lastName;
```

Pass:

```
let fullName = `${firstName} ${lastName}`;
```

Use ES6 arrow functions where possible

Arrow functions are a more concise syntax for a writing function expression. They’re are anonymous and change the way this binds in functions.

Fail(s):

```
var multiply = function(a, b) {
return a* b;
};
```

Pass:

```
const multiply = (a, b) => { return a * b};
```

Always use curly braces around control structures

Braces are required for all control structures (i.e. if, else, for, do, while, as well as any others)
This can divide opinion with one line if statements but the number of times this has caused mistakes when not using a curly brace. This can cause developers to add other statements underneath thinking the conditional statement is surrounding it.

Try and reduce nesting

An if within if can get messy and very hard to read, very quickly. Sometimes you may not be able to get around but always have a look at the structure of your JavaScript to see if you can change it around.

Lowercase file names

MyFile.js should be myFile.js

Use default parameters where possible

In JavaScript, if you don't pass in a value into a parameter when calling a function it will be undefined

Switch statements should use break and have default

I usually try and discourage the use of switch statements but you really want to use them make sure you break each condition and use a default .

Use named exports

Do not use default exports. Importing modules must give a name to these values, which can lead to inconsistencies in naming across modules.

Use shortcuts for booleans

Fail:

```
if (isValid === true)
if (isValid === false)
```

Pass:

```
if (isValid)
if (!isValid)
```

Try and avoid unneeded ternary statements

Fail:

```
const boo = a ? a : b;
```

Pass:

```
const boo = a || b;
```

Only use one variable per declaration

This one can split opinion again but I feel declaring more than one at a time, they can get missed at times.

Fail:

```
let a = 1, b = 2;
```

Pass:

```
let a = 1;
let b = 2;
```

Check here for the full standards.