DWA_04.3 Knowledge Check_DWA4

---

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

a) 10.4 Only import from a path in one place. eslint: `no-duplicate-imports`
Why? Having multiple lines that import from the same path can make code harder to maintain.

```
// bad
import foo from 'foo';
// … some other imports … //
import { named1, named2 } from 'foo';

// good
import foo, { named1, named2 } from 'foo';

// good
import foo, {
  named1,
  named2,
} from 'foo';
```

It is important to avoid many links especially if you are importing from one file and I remember I was made this mistake and I really felt like it was a real eye opener when coach made me aware of this mistake.

b) 7.11 Spacing in a function signature. eslint: `space-before-function-paren` `space-before-blocks`
Why? Consistency is good, and you shouldn't have to add or remove a space when adding or removing a name.

```
// bad
const f = function(){};
const g = function (){};
const h = function() {};

// good
const x = function () {};
const y = function a() {};
```

When formatting a function, whitespace is allowed between the function name or `function` keyword and the opening paren. Named functions also require a space

between the `function` keyword and the function name, but anonymous functions require no whitespace. For example:

```
function withoutSpace(x) {
    // ...
}


function withSpace (x) {
    // ...
}

var anonymousWithoutSpace = function() {};

var anonymousWithSpace = function () {};
```

With regards to spacing it is important to add space not only for the sake of readability but even for code to look neat.

c) 8.3 In case the expression spans over multiple lines, wrap it in parentheses for better readability.

Why? It shows clearly where the function starts and ends.

```
// bad
['get', 'post', 'put'].map((httpMethod) =>
Object.prototype.hasOwnProperty.call(
    httpMagicObjectWithAVeryLongName,
    httpMethod,
  )
);

// good
['get', 'post', 'put'].map((httpMethod) => (
  Object.prototype.hasOwnProperty.call(
    httpMagicObjectWithAVeryLongName,
    httpMethod,
  )
));
```

I like this one hence i normally use word wrap on Visual Studio Code, but now that I have learned that I can use parenthesis instead, I will use it instead of the other option I used to use.

---

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

a) 6.4 Never use `eval()` on a string, it opens too many vulnerabilities. eslint: `no-eval`

> JavaScript's `eval()` function is potentially dangerous and is often misused. Using `eval()` on untrusted code can open a program up to several different injection attacks. The use of `eval()` in most contexts can be substituted for a better, alternative approach to a problem.

```
var obj = { x: "foo" },
    key = "x",
    value = eval("obj." + key);
```

I really don't understand this and i tried to look it up but it is still confusing

b) 7.9 Always put default parameters last. eslint: `default-param-last`

```
// bad

function handleThings(opts = {}, name) {
  // ...
}
```

```
// good

function handleThings(name, opts = {}) {

  // ...

}
```

Putting default parameter at last allows function calls to omit optional tail arguments.

```
// Correct: optional argument can be omitted

function createUser(id, isAdmin = false) {}

createUser("tabby")



// Incorrect: optional argument can **not** be omitted

function createUser(isAdmin = false, id) {}

createUser(undefined, "tabby")
```

This one is a little unclear as well, and feel I need to look it up

c) 9.3 Methods can return `this` to help with method chaining.

```
// bad

Jedi.prototype.jump = function () {

  this.jumping = true;

  return true;

};
```

```
Jedi.prototype.setHeight = function (height) {
  this.height = height;
};

const luke = new Jedi();
luke.jump(); // => true
luke.setHeight(20); // => undefined

// good
class Jedi {
  jump() {
    this.jumping = true;
    return this;
  }

  setHeight(height) {
    this.height = height;
    return this;
  }
}

const luke = new Jedi();
```

```
luke.jump()

  .setHeight(20);
```

This is confusing and I really can’t explain why, but I need to research more

---