DWA_04.3 Knowledge Check_DWA4

---

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

Naming conventions: Avoid single-letter names. Be descriptive with your naming. eslint: id-length

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

// good
function query() {
  // ...
}
```

White Spacing: Do not pad your blocks with blank lines. eslint: padded-blocks

The code can be seen as one block of code and it's easier to read. It also prevents the accidental deletion of curtain codes that can be mistaken for not belonging anywhere.

```
// bad
function bar() {
```

```
  console.log(foo);

}

// good
if (baz) {
  console.log(quux);
} else {
  console.log(foo);
}
```

A base filename should exactly match the name of its default export.

```
// file 1 content
class CheckBox {
  // ...
}
export default CheckBox;

// file 2 contents
export default function fortyTwo() { return 42; }

// file 3 contents
export default function insideDirectory() {}

// in some other file

// bad
```

```
import CheckBox from './checkBox'; // PascalCase import/export, camelCase filename
import FortyTwo from './FortyTwo'; // PascalCase import/filename, camelCase export
import InsideDirectory from './InsideDirectory'; // PascalCase import/filename, camelCase export
```

.1 Use === and !== over == and !=.

This rule suggests using strict equality (===) instead of loose equality (==) when comparing values.

- Strict equality compares both the value and the type of the operands. This ensures that the comparison is more explicit and less prone to unexpected behavior due to type coercion.
- Loose equality performs type coercion if needed.

---

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

14.2 Anonymous function expressions hoist their variable name, but not the function assignment.

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

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

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

14.3 Named function expressions hoist the variable name, not the function name or the function body.

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

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

  superPower(); // => ReferenceError superPower is not defined

  var named = function superPower() {
    console.log('Flying');
  };
}

// the same is true when the function name
// is the same as the variable name.

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

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

  var named = function named() {
```

```
    console.log('named');
  };
}
```

14.4 Function declarations hoist their name and the function body.

```
function example() {
  superPower(); // => Flying

  function superPower() {
    console.log('Flying');
  }
}
```

"Disallow dangling underscores at the end of names"

The rule suggests avoiding the use of trailing underscores in variable or function names. However, some developers may find this rule confusing or may not fully understand the reasoning behind it. While consistency in naming conventions is important, the specific prohibition of trailing underscores may not always be clear or necessary in every context.

"Do not use iterators like forEach or for...in"

This rule advises against using certain iteration methods like forEach or for...in, and instead recommends using alternative

constructs like for...of or Array.from. While this guideline can help avoid potential issues related to iteration and scoping, it may also be perceived as overly restrictive or less familiar to developers who are accustomed to using forEach or for...in in their code.
