DWA_04.3 Knowledge Check_DWA4

---

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

18.2 Use `//` for single line comments. Place single line comments on a newline above the subject of the comment. Put an empty line before the comment unless it's on the first line of a block.

bad
```
const active = true;  // is current tab
```

good
```
// is current tab
const active = true;
```

bad
```
function getType() {
  console.log('fetching type...');
  // set the default type to 'no type'
  const type = this.type || 'no type';

  return type;
}
```

good
```
function getType() {
  console.log('fetching type...');

  // set the default type to 'no type'
  const type = this.type || 'no type';

  return type;
}
```

also good
```
function getType() {
  // set the default type to 'no type'
  const type = this.type || 'no type';

  return type;
}
```

19.1 Use soft tabs (space character) set to 2 spaces. eslint: `indent`

bad
```
function foo() {
····let name;
```

```
}
```

bad

```
function bar() {
·let name;
}
```

good

```
function baz() {
··let name;
}
```

13.8 Disallow unused variables. eslint: `no-unused-vars`

Variables that are declared and not used anywhere in the code are most likely an error due to incomplete refactoring. Such variables take up space in the code and can lead to confusion by readers.

---

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

1. **Arrays (4.3) Use array spreads […] to copy arrays.**
   Example**:**

```
// bad
const len = items.length;
const itemsCopy = [];
let i;

for (i = 0; i < len; i += 1) {
  itemsCopy[i] = items[i];
}

// good
const itemsCopy = [...items];
```

Why is it confusing?

Why is using the fully written code not better if it allows the reader to better understand the transfer of the content of one array to another? Using array spreads are less detailed.

2. **Hoisting (14.2) Anonymous function expressions house their variable name, but not the function assignment**

   Example**:**

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

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

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

Why is it confusing?

- Why is this allowed? The result of the function defined below returns ‘is not a function’ when it is called above.
- Is the code above just used to define a point?
- Is it returning an error because it’s technically a variable holding a function (and didn’t start out as an initialised function) or is it simply because the callback is before the anon function?

3. **Functions (7.2) Wrap immediately invoked function expressions in parentheses**

   Example:

```
(function () {
  console.log('Welcome to the Internet. Please follow me.');
}());
```

Why is it confusing?

Additional brackets affect readability and may be unnecessary, since all functions technically run immediately when they are invoked