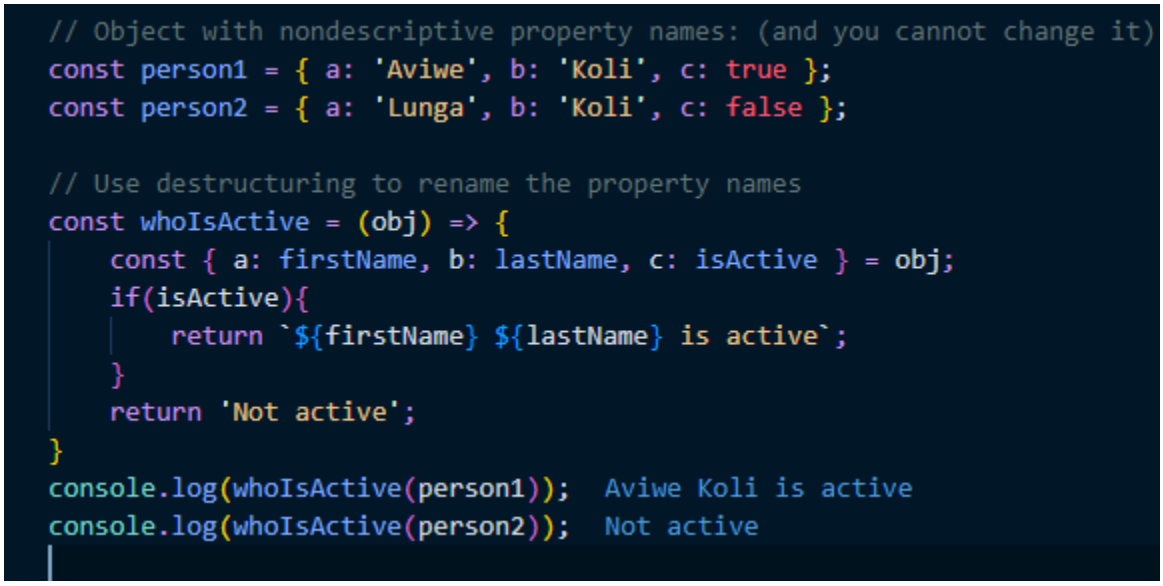
DWA_04.3 Knowledge Check_DWA4

---

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

   a) In modules, do not use wildcard imports:
   Explicit imports come with benefits such as having a clear and maintainable codebase. Wildcard exports make it hard to see which modules or functions are intended to be used, causing confusion. Also, using explicit imports may prevent naming collisions, for example, if two modules have the same export name, this may lead to conflict if you're using wildcard exports as it becomes unclear which export module you're using.

   b) Use destructuring when accessing and using multiple properties of an object:
   This is useful in so many ways in modern JavaScript development, as mentioned in the Airbnb Style Guide, it makes your code more readable and prevents you as the developer from writing repetitive code. One other thing that stands out for me is the fact that you can rename variables during destructuring, this is very helpful when dealing with objects that have property names that you do not control, or property names that are not descriptive. An example of this is shown below:



```
// Object with nondescriptive property names: (and you cannot change it)
const person1 = { a: 'Aviwe', b: 'Koli', c: true };
const person2 = { a: 'Lunga', b: 'Koli', c: false };

// Use destructuring to rename the property names
const whoIsActive = (obj) => {
    const { a: firstName, b: lastName, c: isActive } = obj;
    if(isActive){
        return `${firstName} ${lastName} is active`;
    }
    return 'Not active';
}
console.log(whoIsActive(person1));  Aviwe Koli is active
console.log(whoIsActive(person2));  Not active
```

c) Prefer spread operator syntax over “Object.assign” when making shallow-copies of objects. And also for Arrays as well:
Using the spread operator syntax when creating a copy of an object or array that has additional properties or items is very useful in development as it prevents mutation in both objects and arrays. To clearly demonstrate this, the example below shows an original object that has properties name and surname, a copy is created using both Object.assign() and the spread operator, and an additional property isActive is added to the copy of the array.

ws.js

```
//Using the Object.assign to make a copy
const original = {
    name : 'Aviwe',
    surname: 'Koli',
}
const copy = Object.assign(original, {isActive: true})
console.log(original);  { name: 'Aviwe', surname: 'Koli', isActive: true }
console.log(copy);  { name: 'Aviwe', surname: 'Koli', isActive: true }
```

jectsPlayground.js

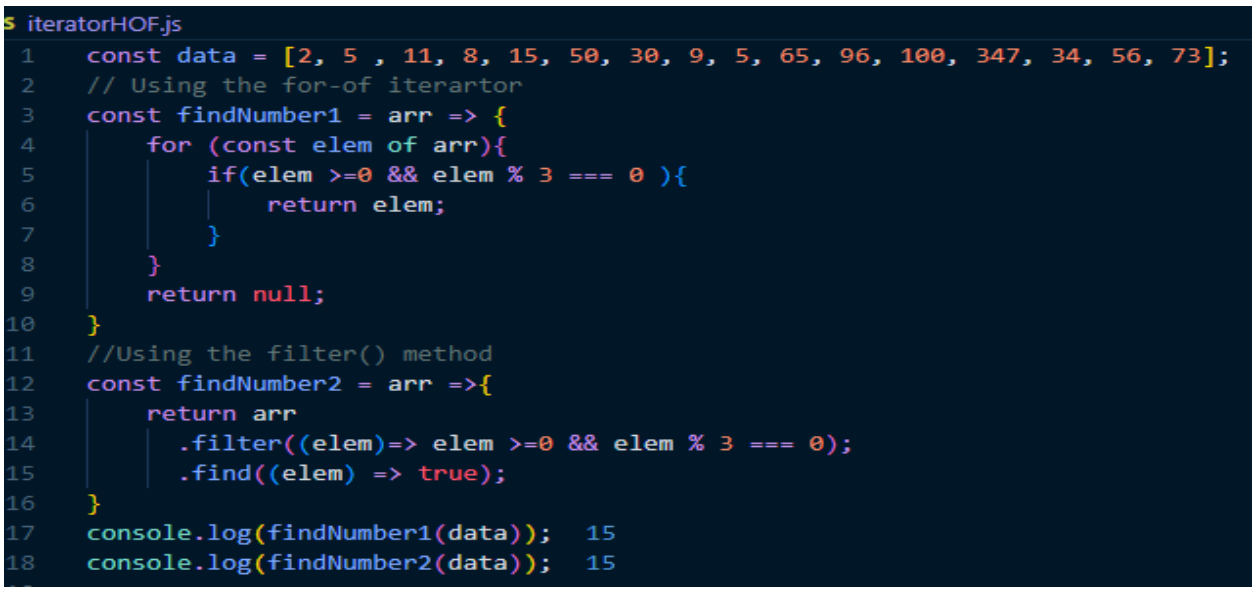
```
//Using the spread operator syntax to make a copy
const original = {
    name : 'Aviwe',
    surname: 'Koli',
}
const copy = {...original, isActive: true}
console.log(original);  { name: 'Aviwe', surname: 'Koli' }
console.log(copy);  { name: 'Aviwe', surname: 'Koli', isActive: true }
```

---

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

a) Do not use iterators(for-in, for-of, etc). Prefer Javascript’s HOF (map(), filter()):
As much as this makes sense in terms of managing complexity (Functional Programming practices that avoid mutation), iterators give us fine-grain control over HOF. For example, suppose you have a large set of numbers (maybe in an array), and you want to return the first number that is greater than 10 and divisible by 3. Using a for-of loop in this case is efficient as the for-of loop will terminate the loop as soon the condition is met, while on the other hand, using a HOF, say the filter() and find() methods, it would achieve the same results but it will still

iterate through the entire array even after finding the desired number, this may be less efficient in large codebases.



b) Do not include Javascript filename extensions when importing:
Airbnb Style guide explains this based on the fact that adding filename JavaScript extensions prevents refactoring, yes refactoring can lead to codebases that are easy to maintain, extend and debug, but what if you're working in a larger codebase that imports various file types and some of which have the same names (index.js, index.jsx, index.ts), wouldn't this result in confusion and bugs?
Including filename extensions when importing makes it clear to developers the type of files they're working with. Also, using filename extensions when importing can also be seen as some form of documentation for your codebase, making it easy to communicate to other developers as to what type of modules they can expect when using the imports.

c) In Accessor: Airbnb Style guide says that
(24.2 Do not use JavaScript setters/ getters) and then went on to say
(24.4 It's okay to use get() and set(), but be consistent)
Which is which exactly?
Not only is this confusing to me, but accessor functions allow us as developers to control access to object properties, and to enforce constraints or restrictions, for example, you can use accessors to validate inputs, enforce read-only properties, or trigger certain actions when values change.
Also, accessor functions are very useful when working with classes.