What's the JavaScript Nullish Coalescing operator?

Written by Seth Corker on Benevolent Bytes

What is it?

The bullish coalescing operator is a long name for such a small and unassuming `??`. Its job is to take two values, the left-hand side and right-hand side. If the left-hand side is `null` or `undefined` it will return the right-hand side. To better explain it, let's have a look at an example.

A coffee ordering example

```
function orderCoffee(options = {}) {
  const sugarAmount = options.addSugar ?? "ask customer for sugar";

  return `Coffee (${Number.isInteger(sugarAmount) ? `${sugarAmount * 5}g of sugar` : sugarAmount})`;
}

console.log("Order #1 - " + orderCoffee({}));
console.log("Order #2 - " + orderCoffee({ addSugar: 0 }));
console.log("Order #3 - " + orderCoffee({ addSugar: 1 }));

// Output
// Order #1 - Coffee (ask customer for sugar)
// Order #2 - Coffee (0g of sugar)
// Order #3 - Coffee (5g of sugar)
```

We've got an `orderCoffee` function that takes some options, just one option for now, specifically `addSugar`. If the customer specifies how much sugar they want, we append the amount like, `(5g of sugar)`, to the end of our order string. If the customer doesn't tell us, we make a note to ask them.

Why not `||`?

The null coalescing operator is useful in this situation because `options.addSugar` can be a `number` or `undefined`. If we were to use the `||` operator, we might get some slightly different results.

Let's have a look at the output if we changed to using the logical or operator:

```
function orderCoffee(options = {}) {
  const sugarAmount = options.addSugar || "ask customer for sugar";

  return `Coffee (${Number.isInteger(sugarAmount) ? `${sugarAmount * 5}g of sugar` : sugarAmount})`;
}

console.log("Order #1 - " + orderCoffee({}));
console.log("Order #2 - " + orderCoffee({ addSugar: 0 }));
console.log("Order #3 - " + orderCoffee({ addSugar: 1 }));

// Output
// Order #1 - Coffee (ask customer for sugar)
// Order #2 - Coffee (ask customer for sugar)
// Order #3 - Coffee (5g of sugar)
```

Notice that order #1 is the same and order #3 is the same, but order #2 is incorrect. The customer has told us they don't want coffee but `||` sees the number 0 as a falsey value!

Where is it useful?

I've found `??` useful for defaults. In the example, we have an `options` object that's passed in to `orderCoffee`. `??` allows use to take the value that's passed in or, if nothing has been provided, we can choose a nice default instead.

What's the old way of doing it?

Without `??`, the common way would be to use `||` but this leads to bugs like we saw above. So, the next best thing would be to check explicitly like this:

```
function orderCoffee(options = {}) {
  let sugarAmount = "ask customer for sugar";
  if (Number.isInteger(options.addSugar)) {
    sugarAmount = `${options.addSugar * 5}g of sugar`;
  }

  return `Coffee (${sugarAmount})`;
}
```

This isn't too bad, but it can become quite verbose if we had more options, we'd have to have a lot more repeated if statements. It's a small improvement that won't be useful all the time but it can come in handy and is a nice tool to add to your JavaScript toolbelt.

Resources

- Nullish coalescing operator (??) on MDN