Lesson | Thursday

Introduction to Programming (/introduction-to-programming) / Arrays and Looping (/introduction-to-programming/arrays-and-looping) / Further Exploration: Looping with for...of

Text | Cheat sheet

Note: This lesson is a *further exploration*, and it is not required that you read it! `for...of` loops are more syntactic sugar on regular `for` loops. They are useful for certain kinds of problems such as looping directly over strings.

In this lesson, we'll cover some JavaScript syntactic sugar that we can use to loop over arrays, strings, and other iterable objects. Specifically, we'll take a look at looping with `for...of`, which has advantages over both `Array.prototype.forEach()` and standard `for` loops.

Looping with `for...of`

First, we'll take a look at how we can use this kind of loop with an array. Yes, are going to do the thing where we double the numbers again — if only so we can easily compare the difference between each kind of JavaScript loop.

Here's how we can create an array of doubled numbers using `for...of`:

```
> const array = [0,1,2,3,4,5];
> let doubledArray = [];
> for (const element of array) {
  doubledArray.push(element * 2);
}
> doubledArray;
(6) [0, 2, 4, 6, 8, 10]
```

At this point, you might be thinking "so what?" It's just as easy to use `Array.prototype.forEach()` — why not just use that to loop over arrays instead?

Well, check this out:

```
> const array = [0,1,2,3,4,5];
> let doubledArray = [];
> for (const element of array) {
  doubledArray.push(element * 2);
  console.log("Never mind!");
  break;
}
> doubledArray;
(1) [0]
```

Unlike with `Array.prototype.forEach()`, we can break out of a `for...of` loop with either a `break;` statement or a return statement. That's a pretty big advantage. As you can see, `for...of` loops don't have issues with OBOEs — we can just use them to iterate through any array, breaking out of them when we need to.

Pretty awesome — and more convenient than either a `for` loop or `Array.prototype.forEach()` when iterating over arrays.

But wait! `for...of` loops get better. We can use `for...of` to iterate over strings, too. For instance:

```
> const consonantString = "bdfmxtgl";
> let vowelizedString = "";
> for (const letter of consonantString) {
  vowelizedString = vowelizedString.concat(letter + "a");
}
> vowelizedString;
"badafamaxatagala"
```

This returns the string `"badafamaxatagala"`.

Very convenient! This is another huge advantage of `for...of`. `Array.prototype.forEach()` can't iterate over strings.

By the way, you might be thinking "Hey! Why didn't I learn about this *before* the Pig Latin project? It might've been helpful!" Well, maybe so — but it's also important to practice using methods like `String.prototype.split()` and `Array.prototype.join()` — plus good old fashioned basic `for` loops — all practice that we might be able to skip if we jumped right to `for...of` loops.

In general, you can (and should) use `for...of` whenever you want to iterate over strings or arrays — even if you don't plan to break or return out of these loops. There are always use cases when you'll want to use a different kind of JavaScript loop but this is a good all-purpose loop — a tool that you'll reach for often because it's flexible, concise, and gets the job done.

Previous (/introduction-to-programming/arrays-and-looping/practice-credit-card-validator-roman-numerals-or-cryptosquare)
Next (/introduction-to-programming/arrays-and-looping/further-exploration-while-loops)

disable dark mode

(http://www.epicodus.com)