Finally, we have the mighty for...of loop.

**For...of loop**

The **for...of loop** is used to loop over any type of data that is *iterable*.

You write a **for...of** loop almost exactly like you would write a **for...in** loop, except you swap out in with of and you can drop the **index**.

**const** digits = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9];

**for** (**const** digit **of** digits) {

console.log(digit);

}

***Prints:*** *0  
1  
2  
3  
4  
5  
6  
7  
8  
9*

This makes the for...of loop the most concise version of all the for loops.

***TIP:****It’s good practice to use plural names for objects that are collections of values. That way, when you loop over the collection, you can use the singular version of the name when referencing individual values in the collection. For example, for (const button of buttons) {...}.*

Play Video

Progression from the for loop to the for...in loop to the for...of loop.

But wait, there’s more! The for...of loop also has some additional benefits that fix the weaknesses of the for and for...in loops.

You can stop or break a for...of loop at anytime.

**const** digits = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9];

**for** (**const** digit **of** digits) {

**if** (digit % 2 === 0) {

**continue**;

}

console.log(digit);

}

***Prints:*** *1  
3  
5  
7  
9*

And you don’t have to worry about adding new properties to objects. The for...of loop will only loop over the values in the object.

Array.prototype.decimalfy = **function**() {

**for** (i = 0; i < **this**.length; i++) {

**this**[i] = **this**[i].toFixed(2);

}

};

**const** digits = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9];

**for** (**const** digit **of** digits) {

console.log(digit);

}

***Prints:*** *0  
1  
2  
3  
4  
5  
6  
7  
8  
9*

This time, the properties were *not* printed out to the console, like we saw on the prior page.