Lesson | Thursday

Introduction to Programming (/introduction-to-programming) / Arrays and Looping (/introduction-to-programming/arrays-and-looping) / Further Exploration: While Loops

Text | Cheat sheet

Note: This lesson is a *further exploration*, and it is not required that you read it! `while` and `do...while` loops aren't as commonly used these days. You can usually solve the same problems with more common loops (such as `for`). However, you will still see them in the real world and it's one more tool to be aware of.

We'll conclude this section with a quick peek at `while` and `do...while` loops. You probably won't use these very often. However, running a loop *while* a condition is true is such a fundamental part of programming that it's important to get some exposure to this technique. You'll see `while` loops in just about every programming language.

Looping with `while` and `do...while`

`while` and `do...while` loops return to the most basic concept of looping — run a loop until a condition is no longer true. Let's create a loop that counts down from ten:

```
> let number = 10;
> while (number > 0) {
  console.log(number);
  number --;
}
> console.log("Blast off!");
10
9
8
7
6
5
4
3
2
1
Blast off!
```

As we can see, we get a countdown from 10 logged in the console followed by `"Blast off!"`. The syntax of a `while` loop is simple:

```
// This is pseudo-code!
while (condition is true) {
  loop
}
```

In our countdown example, as long as `number > 0`, our loop will keep running. Note also that we used `number --`. We've mostly used `++` to *increment* a number by one, but we can just as easily use `--` to *decrement* a number by one.

We have to be very careful with `while` loops — it's very easy to accidentally create an infinite loop, which will crash our application. For instance, if we accidentally omitted `number --`, we'd create an infinite loop because our number would never decrement.

Since we've used the doubled array example for every other kind of loop, let's do that again here for comparison's sake:

```
> const array = [0,1,2,3,4,5];
> let index = 0;
> let doubledArray = [];
> while (index < array.length) {
  doubledArray.push(array[index] * 2);
  index ++;
}
> doubledArray;
(6) [0, 2, 4, 6, 8, 10]
```

It's really pretty clunky. In addition to initializing a `doubledArray`, we also have to initialize a variable containing an `index`. Then, when we specify the condition, we need to be very careful about OBOEs and other errors. And if we were to forget `index ++`, that would cause an infinite loop.

So as we can see, `while` loops aren't as concise as other kinds of loops — and when something goes wrong with a `while` loop, it can really go wrong, causing infinite loops and crashed apps.

`do...while`

Let's look at another type of `while` loop that has slightly different syntax:

```
> let number = 10;
> do {
  console.log(number);
  number --;
} while (number > 0);
> console.log("Blast off!");
10
9
8
7
6
5
4
3
2
1
Blast off!
```

The `do...while` loop works exactly the same as a `while` loop. It just has different (and honestly, more confusing) syntax:

```
// This is pseudo-code!
do {
  loop
} while (condition is true)
```

Even though you probably won't be using `while` loops very much, there are still valid use cases for them. You should just favor other "higher-level" loops where possible. And as always, try them out and practice writing them!

Previous (/introduction-to-programming/arrays-and-looping/further-exploration-looping-with-for-of)
Next (/introduction-to-programming/arrays-and-looping/optional-review-which-loop-should-i-use)

disable dark mode

(http://www.epicodus.com)