Theory: For loops

11 minutes 0 / 5 problems solved

Sometimes programmers need to repeat the same part of code several times over. Assume that you go to work every day, Monday to Friday, and greet your colleague Nessie as you get to your office: “Hi Nessie!” Let's simulate this situation using the tools we already have:

```
console.log("1, Hi Nessie!"); // Monday
console.log("2, Hi Nessie!"); // Tuesday
console.log("3, Hi Nessie!"); // Wednesday
console.log("4, Hi Nessie!"); // Thursday
console.log("5, Hi Nessie!"); // Friday
```

We can create a loop instead of all that repetition:

```
let i;
for (i = 1; i <= 5; i++) {
    console.log(i + ", Hi Nessie!");
}
```

This is way faster and more convenient, especially if we have to repeat the same stuff 10 times, 100 times or even more.

Let's zoom in on the syntax of the `for` loop. We set a variable before the loop: the `i` counter. Next, the condition inside the brackets has three parts: `(...;...;...)`. The initial value of the counter variable comes first; in our case, it equals `1`. The second part is the condition in which the loop stops, which in our case is `i <= 5`. Finally, in the third part, we indicate how exactly the counter should change from one iteration to another. The `i++` recording is the same as `i = i + 1`, that is, the counter ranges from `1` to `5` with the step of `1` at each iteration. Any part of the condition inside the brackets is optional and can be omitted.

§1. Infinite loop

An infinite loop is a loop whose execution never stops because there is no condition for exiting the loop (the second condition in the brackets). To write an infinite loop, just skip the condition for exiting the loop to make a construction like this:

```
for (i=0; ; i++) {
    console.log(i);
}
```

... or this:

```
for (i=0; ;) {
    console.log(i);
}
```

... or even this!

```
for (; ;) {
    console.log('Hi!');
}
```

Why is that? Since none of the conditions in brackets are mandatory, we can skip a few or even all of them. But since we skip the condition for exiting the loop, the loop becomes infinite.

Be careful with infinite loops: since there's no exiting an infinite loop, sooner or later the memory will overflow. This will make your computer freeze, and you will have to reboot it in emergency mode. Still, sometimes we need the

Current topic:

For loops

Topic depends on:

- ✕ Arrays
- ✕ Conditional operators
- ✕ Increment and decrement

Topic is required for:

- Break and continue
- Scope of variables

Table of contents:

loop to run endlessly, for example, when programming games or microcontrollers, so in these cases, the use of infinite loops is reasonable.

§2. For...in loop

Let's examine another loop type related to the `for` loop category, the `for...in` loop:

```
let animals = ['cat', 'dog', 'turtle'];

for (x in animals) {
  console.log(animals[x]);
}
```

We get the following as the output:

```
cat
dog
turtle
```

In this example, we created an array containing the names of animals. Next, we go through the indexes of each animal in the array and display the names of the animals, one per line.

Note that the `for...in` loop passes only through the enumerable properties of the object, that is, by properties that can be counted.

§3. Conclusion

In this topic, we figured out what loops are, where we need them. We also looked at the syntax of two types of loops and learned what an infinite loop would look like if we wrote it using the `for` loop.