

# Loops

JavaScript

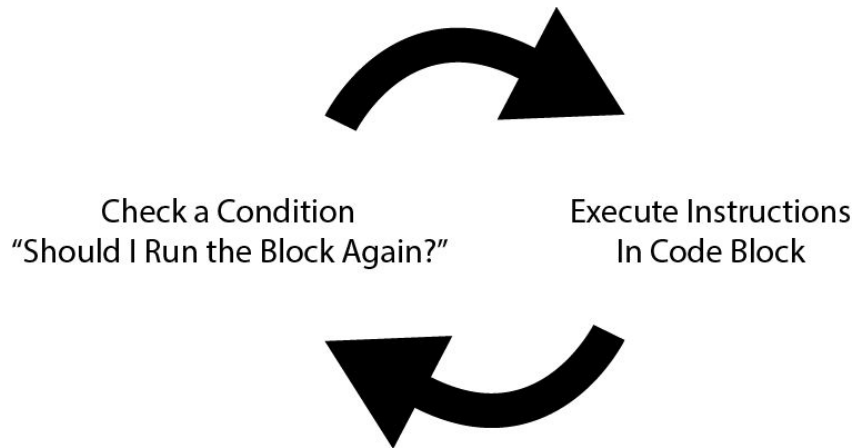


# What are loops?

Loops offer us a way to run through a block of code one or more times.

This can be very useful when trying automate a repetitive action, as you can avoid writing the same lines of code over and over.

They also give us a fast and easy way to iterate through each of the items stored in an array!



# Types of Loops

There are multiple types of loops you can use in most programming languages. Here are a few that you'll see most often in JavaScript...

- [Traditional for Loop](#)
- [for...in Loop \(Array Loops\)](#)
- [for...of Loop \(Object Loops\)](#)

*We'll look at this one later, but it is very similar to the for...in loop!*

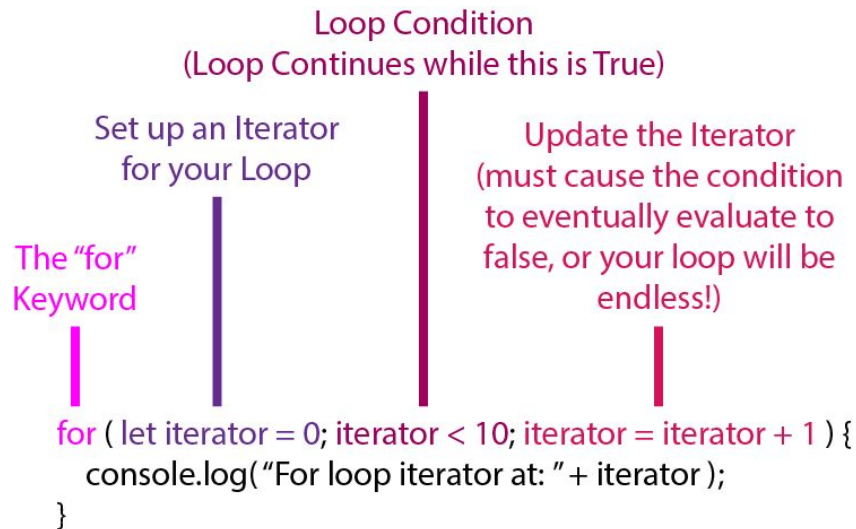
- [do...while Loop](#)
- [while Loop](#)

These types of statements follow a very similar syntax to what we saw with if statements.

```
keyword ( condition ) {  
    code block  
}
```

# Traditional for Loop

The traditional [for loop](#) is composed of three essential expressions:

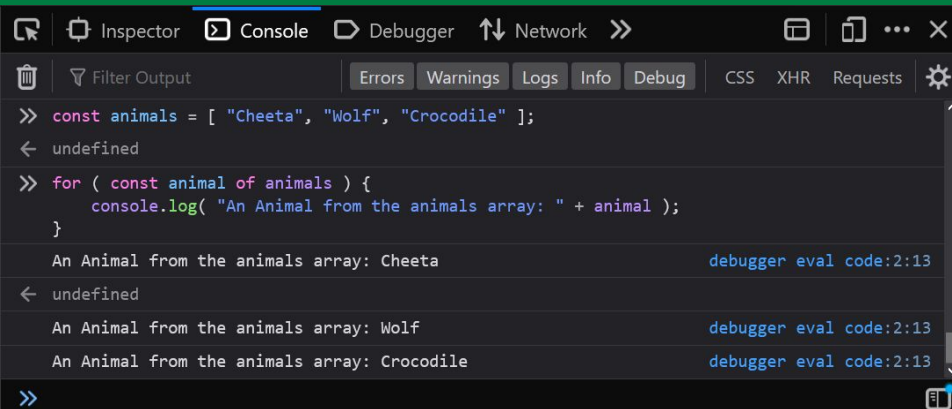


The screenshot shows a web browser's developer console with the following content:

- Inspector:** Empty.
- Console:** Shows the execution of a for loop. The first line of code is `>> for ( let iterator = 0; iterator < 10; iterator = iterator + 1 ) { console.log( "for loop iterator variable value: " + iterator ); }`. Below this, there are ten log entries, each showing the value of the iterator from 0 to 9. Each log entry is followed by the text `debugger eval code:2:13`.
- Debugger:** Empty.
- Network:** Empty.
- Filter Output:** Set to "Filter Output".
- Errors:** Empty.
- Warnings:** Empty.
- Logs:** Contains the log entries.
- Info:** Empty.
- Debug:** Empty.
- CSS:** Empty.
- XHR:** Empty.
- Requests:** Empty.
- Settings:** Empty.

# for...of Loop

The [for...of](#) loop is used for iterating through items contained within an array:



```
>> const animals = [ "Cheeta", "Wolf", "Crocodile" ];
← undefined

>> for ( const animal of animals ) {
  console.log( "An Animal from the animals array: " + animal );
}
An Animal from the animals array: Cheeta    debugger eval code:2:13
← undefined
An Animal from the animals array: Wolf      debugger eval code:2:13
An Animal from the animals array: Crocodile  debugger eval code:2:13
>>
```

```
const animals = [ "Cheeta", "Wolf", "Crocodile" ];
```

The "of" Keyword

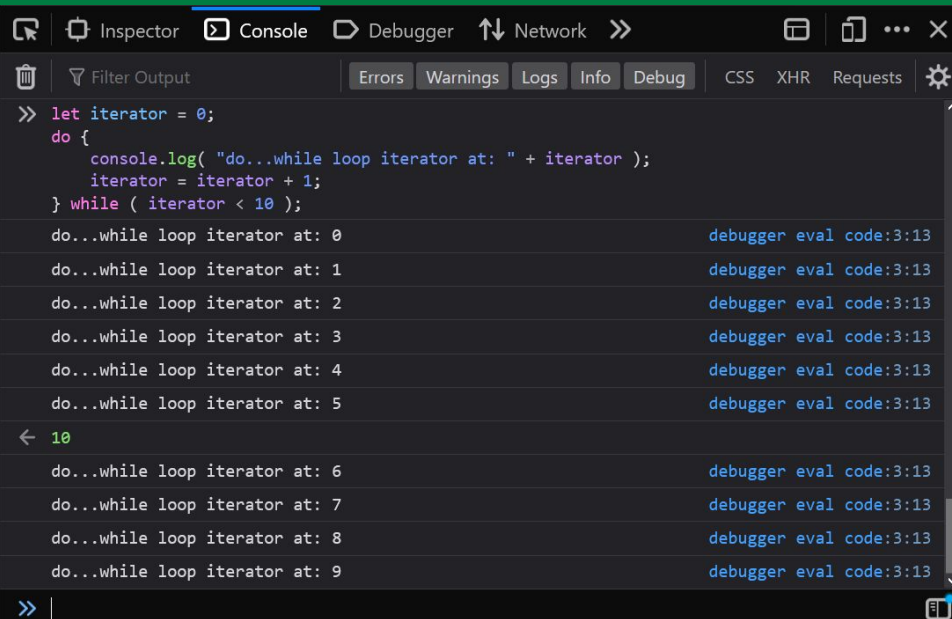
Variable Representing  
the Value of the Current  
Item in the Array

The "for"  
Keyword

An Array, or Variable  
Containing an Array

```
for ( const animal of animals ) {
  console.log( "An animal from the animals array: " + animal );
}
```

# do...while Loop



```
>> let iterator = 0;
do {
  console.log( "do...while loop iterator at: " + iterator );
  iterator = iterator + 1;
} while ( iterator < 10 );
```

do...while loop iterator at: 0  
do...while loop iterator at: 1  
do...while loop iterator at: 2  
do...while loop iterator at: 3  
do...while loop iterator at: 4  
do...while loop iterator at: 5

← 10

do...while loop iterator at: 6  
do...while loop iterator at: 7  
do...while loop iterator at: 8  
do...while loop iterator at: 9

Another common loop in programming is the do...while loop:

The "do"  
Keyword

let iterator = 0;

do {

console.log("do...while loop iterator at: " + iterator);

iterator = iterator + 1;

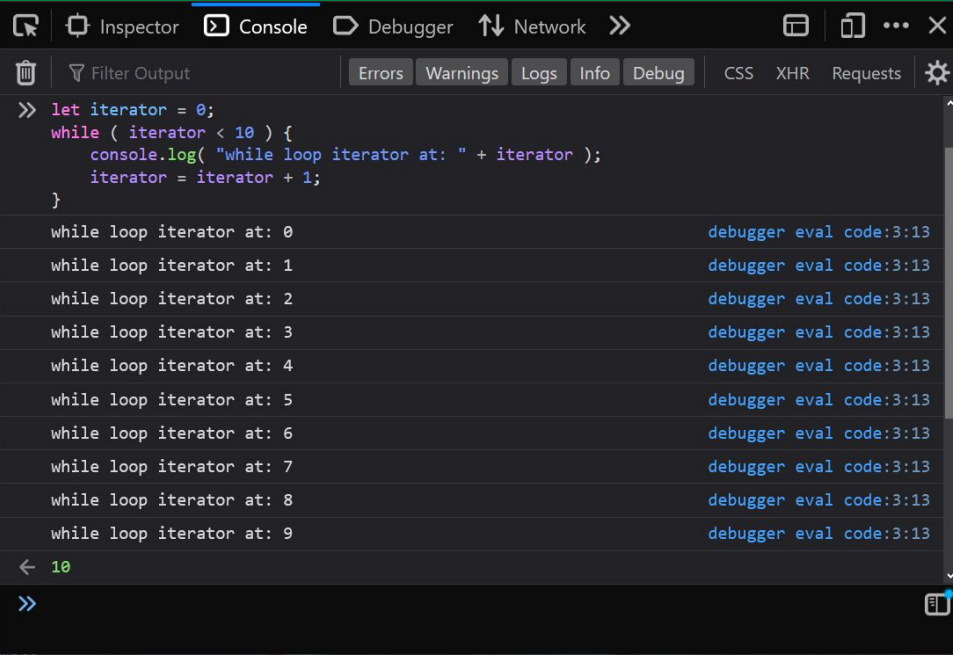
} while ( iterator < 10 );

The "while"  
Keyword

Update the Iterator  
(must cause the condition  
to eventually evaluate to  
false, or your loop will be  
endless!)

Loop Condition  
(Loop Continues while this is True)

# while Loop



The screenshot shows a web browser's developer console with the following content:

```
>> let iterator = 0;
while ( iterator < 10 ) {
  console.log( "while loop iterator at: " + iterator );
  iterator = iterator + 1;
}
```

The console output shows the loop executing 10 times, logging the iterator value from 0 to 9:

```
while loop iterator at: 0
while loop iterator at: 1
while loop iterator at: 2
while loop iterator at: 3
while loop iterator at: 4
while loop iterator at: 5
while loop iterator at: 6
while loop iterator at: 7
while loop iterator at: 8
while loop iterator at: 9
```

The console also shows the debugger's internal state, with the iterator variable being updated from 0 to 9. The console is filtered to show "Logs" and the "Debug" tab is selected.

Similar to do...while, is the [while](#) loop:

