

# The For Loop

This lesson will cover all the basic concepts related to while loops.

## WE'LL COVER THE FOLLOWING ^

- Example
- How it works
- The Loop Counter

You'll often need to write loops with conditions that are based on the value of a variable updated in the loop body, like in our example. JavaScript offers another loop type to account for this: the **for** loop.

## Example #

Here's the same program as above written instead with a **for** loop. It gives exactly the same result.

```
let number;  
for (number = 1; number <= 5; number++) {  
  console.log(number);  
}
```



## How it works #

Here's the for loop syntax.

```
for (initialization; condition; final expression) {  
  // code to run while the condition is true  
}
```



This is a little more complicated than the `while` loop syntax:

- *Initialization* only happens once, when the code first kicks off. It's often used to set the initial value of the variable associated to the loop condition.
- The *condition* is evaluated once before the loop runs each time. If it's true, the code runs. If not, the code doesn't run.
- The *final expression* is evaluated after the loop runs each time. It's often used to update the value of the variable associated with the loop condition, as we saw in the previous example.

## The Loop Counter #

The variable used during initialization, condition, and the final expression of a loop is called a *counter* and is often named `i`. This counter can be declared in the loop initialization to limit its scope to the loop body.

```
for (let i = 1; i <= 5; i++) {  
  console.log(i); // OK  
}  
console.log(i); // Error: the i variable is not visible here
```

