JavaScript while Loop

Summary: in this tutorial, you will learn how to use the JavaScript `while` statement to create a loop that executes a block as long as a condition is `true` .

Introduction to the JavaScript while loop statement

The JavaScript `while` statement creates a loop that executes a block as long as a condition evaluates to `true` .

The following illustrates the syntax of the `while` statement:

```
while (expression) {
    // statement
}
```

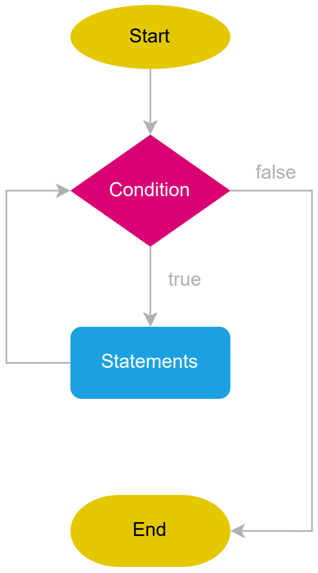
The `while` statement evaluates the `expression` before each iteration of the loop.

If the `expression` evaluates to `true` , the `while` statement executes the `statement` . Otherwise, the `while` loop exits.

Because the `while` loop evaluates the `expression` before each iteration, it is known as a pretest loop.

If the `expression` evaluates to `false` before the loop enters, the `while` loop will never execute.

The following flowchart illustrates the `while` loop statement:



Note that if you want to execute the statement a least once and check the condition after each iteration, you should use the do...while statement.

JavaScript while loop examples

Let's take some examples of using the JavaScript `while` loop statement.

Basic JavaScript while loop example

The following example uses the `while` statement to output the odd numbers between 1 and 10 to the console:

```
let count = 1;
while (count < 10) {
    console.log(count);
    count +=2;
}
```

Output:

```
1
3
5
7
9
```

How the script works

First, declare and initialize the `count` variable to `1` :

```
let count = 1;
```

Second, execute the statement inside the loop if the `count` variable is less than `10` . In each iteration, output the count to the console and increase the count by `2` :

```
while (count < 10) {
    console.log(count);
    count +=2;
}
```

Third, after `5` iterations, the `count` is `11` . Therefore, the condition `count < 10` is `false` , the loop exits.

Calculating the sum of a sequence of numbers

The following example uses the `while` loop to calculate the sum of sequence of numbers from 1 to 100:

```
let total = 0,
  n = 100,
  i = 0;

while (i <= 100) {
  total += i;
  i++;
}

console.log(total);
```

Output:

```
5050
```

How it works.

First, declare and initialize some variables:

```
let total = 0,
  n = 100,
  i = 0;
```

- The `total` variable stores the sum of numbers between `1` and `100`.
- The `n` variable serves as the ending number.
- The `i` variable is a loop variable.

Second, add value of `i` to the `total` as long as `i` is less than or equal to `100`:

```
while (i <= 100) {
  total += i;
  i++;
}
```

Note that you need to increment `i` by one in each iteration. If you don't do so, the condition `i <= 100` will always `true`, which causes an indefinite loop.

Third, display the `total` to the console:

```
console.log(total);
```

Using while loop with an array

The following example uses the `while` loop to iterate over elements of an array:

```
let mountains = ['Mount Everest', 'K2', 'Lhotse', 'Kangchenjunga'];
let i = 0;
while (i < mountains.length) {
  console.log(`${i + 1}.${mountains[i]}`);
  i++;
}
```

Output:

```
1.Mount Everest
2.K2
3.Lhotse
4.Kangchenjunga
```

How it works.

First, define an array that stores mountain names:

```
let mountains = ['Mount Everest', 'K2', 'Lhotse', 'Kangchenjunga'];
```

Second, declare and initialize a loop variable (`i`) to zero:

```
let i = 0;
```

Third, run the loop as long as the loop variable (`i`) is less than the number of elements in the `mountains` array:

```
while (i < mountains.length) {
```

Finally, show the mountain names and increment the loop variable (`i`) in each iteration:

```
  console.log(`${i + 1}.${mountains[i]}`);
  i++;
}
```

Since the `mountains` array has four elements, the `while` loop performs four iterations, each per element.

Summary

- Use a `while` loop statement to create a loop that executes a block as long as a `condition` is `true` .

Quiz