JavaScript Remainder Operator

Summary: in this tutorial, you'll learn about the JavaScript remainder operator (`%`) to get the remainder of a number divided by another number.

Introduction to the JavaScript remainder operator

JavaScript uses the `%` to represent the remainder operator. The remainder operator returns the remainder left over when one value is divided by another value.

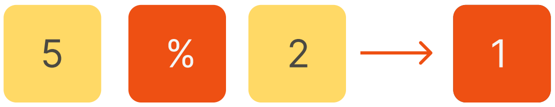
Here's the syntax of the remainder operator:

```
dividend % divisor
```

The following shows the equation for the remainder:

```
dividend = divisor * quotient + remainder
where |remainder| < |divisor|
```

In this equation, the `dividend` , `divisor` , `quotient` , and `remainder` are all integers. The sign of the `remainder` is the same as the sign of the `dividend` .



JavaScript remainder operator examples

Let's take some examples of using the JavaScript remainder operator.

Using the remainder operator with a positive dividend

The following example shows how to use the remainder operator with a positive dividend:

```
let remainder = 5 % -2;
console.log(remainder); // 1

remainder = 5 % 2;
console.log(remainder); // 1
```

Using the remainder operator with a negative dividend

The following example uses the remainder operator with a negative dividend:

```
let remainder = -5 % 3;
console.log(remainder); // -2

remainder = -5 % -3;
console.log(remainder); // -2
```

Using the remainder operator with special values

If a dividend is an `Infinity` and a divisor is a finite number, the remainder is `NaN`. For example:

```
let remainder = Infinity % 2;
console.log(remainder); // NaN
```

If a dividend is a finite number and a divisor is zero, the remainder is `NaN`:

```
let remainder = 10 % 0;
console.log(remainder); // NaN
```

If both dividend and divisor are `Infinity`, the remainder is `NaN`:

```
let remainder = Infinity % Infinity;
console.log(remainder); // NaN
```

If a dividend is a finite number and the divisor is an `Infinity`, the remainder is the dividend. For example:

```
let remainder = 10 % Infinity;
console.log(remainder); // 10
```

If the dividend is zero and the divisor is non-zero, the remainder is zero:

```
let remainder = 0 % 10;
console.log(remainder); // 0
```

If either dividend or divisor is not a number, it's converted to a number using the `Number()` function and applied the above rules. For example:

```
let remainder = '10' % 3;
console.log(remainder); // 1
```

Checking if a number is an odd number

To check if a number is an odd number, you use the remainder operator (`%`) like the following example:

```
let num = 13;
let isOdd = num % 2 == 1;
console.log(isOdd); // true
```

In this example, if the `num` is an odd number, the remainder is one. But if the `num` is an even number, the remainder is zero.

Later, you'll learn how to define a function that returns `true` if a number is odd or `false` otherwise like this:

```
function isOdd(num) {
  return num % 2;
}
```

Alternatively, you can use an arrow function in ES6:

```
const isOdd = (num) => num % 2;
```

Remainder vs Modulo operator

In JavaScript, the remainder operator (%) is not the modulo operator.

If you have been working with Python, you may find the `%` represents the modulo operator. However, this is not the case in JavaScript.

To get a modulo in JavaScript, you use the following expression:

```
((dividend % divisor) + divisor) % divisor
```

Alternatively, you can wrap it in a function:

```
const mod = (dividend, divisor) => ((dividend % divisor) + divisor) % divisor;
```

If the division and divisor have the same sign, the remainder and modulo operators return the same result. Otherwise, they return different results.

For example:

```
const mod = (dividend, divisor) => ((dividend % divisor) + divisor) % divisor;

// dividen and divisor have the same sign
console.log('remainder:', 5 % 3); // 2
console.log('modulo:', mod(5, 3)); // 2

// dividen and divisor have the different signs
```

```
console.log('remainder:', -5 % 3); // -2
console.log('modulo:', mod(-5, 3)); // 1
```

Output:

```
remainder: 2
modulo: 2
remainder: -2
modulo: 1
```

Summary

- Use the JavaScript remainder operator (`%`) get the remainder of a value divided by another value.

Quiz