JavaScript Unary Operators

Summary: in this tutorial, you will learn how to use JavaScript unary operators that take a single operand and perform an operation.

Introduction to the JavaScript unary operators

JavaScript unary operators work on one value. The following table shows the unary operators and their meanings:

Unary Operators	Name	Meaning
`+x`	Unary Plus	Convert a value into a number
`-x`	Unary Minus	Convert a value into a number and negate it
`++x`	Increment Operator (Prefix)	Add one to the value
`--x`	Decrement Operator (Prefix)	Subtract one from the value
`x++`	Increment Operator (Postfix)	Add one to the value
`x--`	Decrement Operator (Postfix)	Subtract one from the value

Unary plus (+)

The unary plus operator is a simple plus sign (`+`). If you place the unary plus before a numeric value, it does nothing. For example

```
let x = 10;
let y = +x;

console.log(y); // 10
```

When you apply the unary plus operator to a non-numeric value, it performs a number conversion using the `Number()` function with the following rules:

Value	Result
boolean	`false` to `0`, `true` to `1`
string	Convert the string value based on a set of specific rules
object	Call the `valueOf()` and/or `toString()` method to get the value to convert into a number

For example, the following uses the unary plus operator to convert the string `'10'` to the number `10`:

```
let s = '10';
console.log(+s); // 10
```

The following example uses the unary plus operator (`+`) converts a boolean value into a number, `false` to `0` and `true` to `1`.

```
let f = false,
    t = true;

console.log(+f); // 0
console.log(+t); // 1
```

Output:

```
0
1
```

Suppose you have a `product` object with the `toString()` method as follows:

```
let person = {
  name: 'John',
  toString: function () {
    return '25';
  },
};

console.log(+person);
```

Output:

```
25
```

In this example, we apply the unary plus operator (`+`) on the `person` object that has the `toString()` method, JavaScript engine calls `toString()` method to get the value (`'25'`) and convert it into a number.

The following adds the `valueOf()` method to the `person` object:

```
let person = {
  name: 'John',
  toString: function () {
    return '25';
  },
  valueOf: function () {
    return '30';
  },
};

console.log(+person);
```

Output:

```
30
```

In this example, the `person` object has the `valueOf()` method, the JavaScript engine calls it instead of the `toString()` method to get the value to convert.

Unary minus (-)

The unary minus operator is a single minus sign (`-`). If you apply the unary minus operator to a number, it negates the number. For example:

```
let x = 10;
let y = -x;

console.log(y); // -10
```

If you apply the unary minus operator to a non-numeric value, it converts the value into a number using the same rules as the unary plus operator and then negates the value.

Increment operator (++)

The increment operator has two plus signs (`++`). It has two version: prefix and postfix.

Prefix:

```
++x;
```

Postfix:

```
x++;
```

The following example uses the prefix increment operator to add one to a variable:

```
let age = 25;
++age;

console.log(age); // 26
```

It's equivalent to the following:

```
let age = 25;
age = age + 1;
console.log(age); // 26
```

When you use the prefix increment operator, JavaScript changes the variable before evaluating the statement. For example:

```
let weight = 90;
weight = ++weight + 5;

console.log(weight); // 96
```

In this example:

- First, increase the `weight` on the right-hand side so `++weight` is `91` .
- Second, add five to the `++weight` that returns `96` .
- Third, assign the result to the `weight` on the left-hand side.

Decrement operator (–)

The decrement operator has two minus signs (`--`). Like the increment operator, it has two version: prefix and postfix.

Prefix:

```
--x
```

Postfix:

```
x--
```

The following example uses the prefix decrement operator to subtract one from a variable:

```
let weight = 90;
--weight;

console.log(weight); // 89
```

It is equivalent to the following:

```
let weight = 90;
weight = weight - 1;
```

```
console.log(weight); // 89
```

Likewise, the following example uses a prefix decrement operator:

```
let weight = 90;
weight = --weight + 5;

console.log(weight); // 94
```

In this example:

- First, subtract one from the weight, `--weight` returns `89` .
- Second, add five to the `--weight` that returns `94` .
- Third, assign the result to the weight on the left-hand side.

The postfix increment or decrement operator changes the value after the statement is evaluated. For example:

```
let weight = 90;
let newWeight = weight++ + 5;

console.log(newWeight); // 95
console.log(weight); // 91
```

How it works.

- First, add five to `weight` (90) and assign the result to the `newWeight` (95)
- Second, add one to the `weight` variable after the second statement is completed, the weight becomes 91.
- Third, display both `newWeight` and `weight` to the console.

When applying the increment/decrement operator to a non-numeric value, it performs the following steps :

- First, convert the value into a number using the same rules as the unary plus (+) operator.
- Then, add one to or subtract one from the value.

Summary

- Unary operators work on one value.
- Unary plus (`+`) or minus (`-`) converts a non-numeric value into a number. The unary minus negates the value after the conversion.
- The prefix increment operator adds one to a value. The value is changed before the statement is evaluated.
- The postfix increment operator adds one to a value. The value is changed after the statement is evaluated.
- The prefix decrement operator subtracts one from a value. The value is changed before the statement is evaluated.

- The postfix decrement operator subtracts one from a value. The value is changed after the statement is evaluated.

Quiz