Unary Operators

In this lesson, we meet the unary operators in JavaScript.
Let's begin!

WE'LL COVER THE FOLLOWING

Operators

Every programming language needs expressions to describe calculations, algorithms, and even more complex concepts. Most languages use unary, binary, and sometimes ternary operators as basic building blocks of expressions.

It is the same for JavaScript, it uses operators to manipulate data values.

If you have already used curly-brace languages such as C, C++, Java, or C#, you will find the JavaScript operators familiar.

However, you must be more considerate with JavaScript operators, for the **loosely-typed nature** of the language delivers a few *surprising* things.

Unary operators

JavaScript supports the unary plus (`+`), unary minus (`-`), increment (`++`), and decrement (`--`) operators. The last two can be used either as **prefix** or

postfix operators. In most programming languages, you can apply these operators only on numeric types, but JavaScript is more flexible.

Usage of unary `+` and `-`

It is trivial that you can apply unary plus and unary minus on numbers:

```
var num = 43;
var num2 = +num;
var num3 = -num;
console.log(num2); // 43
console.log(num3); // -43
```

However,

---

you can use these operators in conjunction with non-numeric types as well.

In this case, the value is converted with the `Number()` casting function, and the unary operators are applied on this converted value, as shown in this code snippet:

```
var bool1 = false;
```

```
var undef;
var str1 = "12.5";
var str2 = "8";

var str3 = "dummy";
var obj1 = null;
var obj2 = new Object();
obj2.valueOf = function () {
  return -4.5;
}
var float = 15.6;

console.log(+bool1); // 0;
console.log(-undef); // NaN
console.log(+str2);  // 8
console.log(-str1);  // -12.5
console.log(-str3);  // NaN
console.log(+obj1);  // 0
console.log(-obj2);  // 4.5
console.log(-float); // -15.6
```

The increment and decrement operators have the same semantics as in C, C++, Java, and C#.

The **increment** operator increments its operand by one. If you use it as prefix operator where the `++` is written in front of its operand, the result of the operation is the value of the operand after it has been incremented.

The postfix operator where the `++` is written after its operand, results the value of the operand before it has been incremented.

The **decrement** operator uses exactly the same semantics, except that it decrements the operand by one.

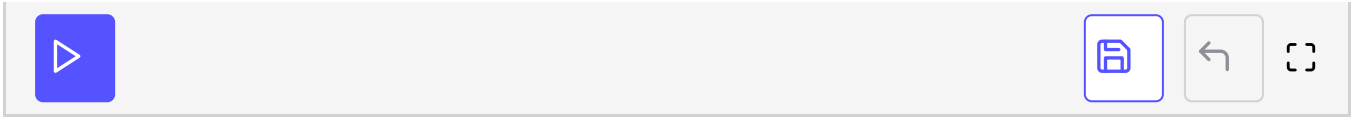
Examples of increment and decrement operators

Here are a few examples:

```
var num = 23;
var num1 = ++num;
console.log(num);  // 24
console.log(num1); // 24
var num2 = num++;
console.log(num);  // 25
console.log(num2); // 24
var num3 = --num;
console.log(num);  // 24
```

```
console.log(num3); // 24
var num4 = num--;


console.log(num);  // 23
console.log(num4); // 24
```



In contrast to other programming languages, the increment and decrement operators in JavaScript work on any values, including integers, strings, Booleans, floating-point values, and objects.

Operator rules regarding values

The operators follow these rules regarding values:

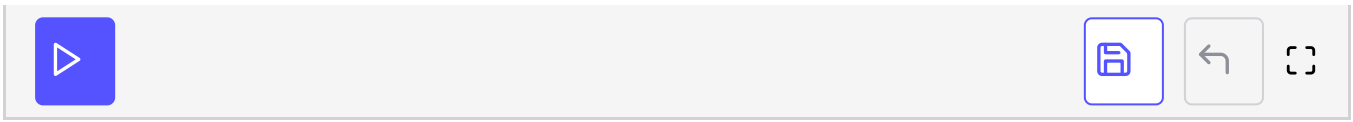
- When used on a floating-point value, the variable's value is changed by adding or subtracting 1.
- When used on another type, the variable's type is changed to a Number, according to the `Number()` casting function, and this value gets incremented or decremented.

This short code snippet shows a couple of examples:

```
var bool1 = false;
var undef;
var str1 = "12.5";
var str2 = "8";
var str3 = "dummy";
var obj1 = null;
var obj2 = new Object();
obj2.valueOf = function () {
  return -4.5;
}
var float = 15.6;
console.log(typeof bool1); // boolean initially
console.log(++bool1);      // 1
console.log(typeof bool1); // number
console.log(++undef);      // NaN
console.log(typeof undef); // number
console.log(++str1);       // 13.5
console.log(typeof str1);  // number
console.log(--str2);       // 7
```

```
console.log(typeof str2);  // number
console.log(--str3);       // NaN
console.log(typeof str3);  // number

console.log(obj1++);       // 0
console.log(typeof obj1);  // number
console.log(obj2--);       // -4.5
console.log(typeof obj2);  // number
console.log(++float);      // 16.6
console.log(typeof float); // number
```



Alright, so now that we've covered unary operators let's move onto the very handy arithmetic operators in the *next lesson*

See you there! :)