Для работы с переменными, со значениями, JavaScript поддерживает все стандартные операторы, большинство которых есть и в других языках программирования.

Несколько операторов мы знаем со школы – это обычные сложение +, умножение \*, вычитание и так далее.

**[Термины: «унарный», «бинарный», «операнд»](http://learn.javascript.ru/operators" \l "%D1%82%D0%B5%D1%80%D0%BC%D0%B8%D0%BD%D1%8B-%D1%83%D0%BD%D0%B0%D1%80%D0%BD%D1%8B%D0%B9-%D0%B1%D0%B8%D0%BD%D0%B0%D1%80%D0%BD%D1%8B%D0%B9-%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D0%BD%D0%B4)**

У операторов есть своя терминология, которая используется во всех языках программирования.

Прежде, чем мы двинемся дальше – несколько терминов, чтобы понимать, о чём речь.

Операнд – то, к чему применяется оператор. Например: 5 \* 2 – оператор умножения с левым и правым операндами. Другое название: «аргумент оператора».

Унарным называется оператор, который применяется к одному операнду. Например, оператор унарный минус "-" меняет знак числа на противоположный:

var x = 1;

x = -x;

alert( x ); // -1, применили унарный минус

Бинарнами називается оператори крторие применят с двумя операторами

var x = 1, y = 3;

alert( y - x ); // 2, бинарный минус

## [Сложение строк, бинарный +](http://learn.javascript.ru/operators" \l "%D1%81%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5-%D1%81%D1%82%D1%80%D0%BE%D0%BA-%D0%B1%D0%B8%D0%BD%D0%B0%D1%80%D0%BD%D1%8B%D0%B9)

Обично при помоши плуса складиваюца числа

var a = "моя" + "строка";

alert( a ); // моястрока

**Если хотя бы один аргумент является строкой, то второй будет также преобразован к строке!**

Причем не важно, справа или слева находится операнд-строка, в любом случае нестроковый аргумент будет преобразован. Например:

alert( '1' + 2 ); // "12"

alert( 2 + '1' ); // "21"

**Это приведение к строке – особенность исключительно бинарного оператора "+".**

Остальные арифметические операторы работают только с числами и всегда приводят аргументы к числу.

Например:

alert( 2 - '1' ); // 1

alert( 6 / '2' ); // 3

### [Преобразование к числу, унарный плюс +](http://learn.javascript.ru/operators" \l "%D0%BF%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D0%BA-%D1%87%D0%B8%D1%81%D0%BB%D1%83-%D1%83%D0%BD%D0%B0%D1%80%D0%BD%D1%8B%D0%B9-%D0%BF%D0%BB%D1%8E%D1%81)

Унарний то есть принемний к одному значению плус ничего не делать с числами

alert( +1 ); // 1

alert( +(1 - 2) ); // -1

Как видно, плюс ничего не изменил в выражениях. Результат – такой же, как и без него.

А что, если их нужно, к примеру, сложить? Бинарный плюс сложит их как строки:

var apples = "2";

var oranges = "3";

alert( +apples + +oranges ); // 5, число, оба операнда предварительно преобразованы в числа

## [Приоритет](http://learn.javascript.ru/operators" \l "%D0%BF%D1%80%D0%B8%D0%BE%D1%80%D0%B8%D1%82%D0%B5%D1%82)

В том случае, если в выражении есть несколько операторов – порядок их выполнения определяется приоритетом.

| **Приоритет** | **Название** | **Обозначение** |
| --- | --- | --- |
| … | … | … |
| 15 | унарный плюс | + |
| 15 | унарный минус | - |
| 14 | умножение | \* |
| 14 | деление | / |
| 13 | сложение | + |
| 13 | вычитание | - |
| … | … | … |
| 3 | присваивание | = |
| … | … |  |

## [Присваивание](http://learn.javascript.ru/operators" \l "%D0%BF%D1%80%D0%B8%D1%81%D0%B2%D0%B0%D0%B8%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)

Обратим внимание, в таблице приоритетов также есть оператор присваиванияж

У него – один из самых низких приоритетов: 3.

var x = 2 \* 2 + 1;

alert( x ); // 5

**Возможно присваивание по цепочке:**

var a, b, c;

a = b = c = 2 + 2;

a lert( a ); // 4

a lert( b ); // 4

alert( c ); // 4

Такое присваивание работает справа-налево, то есть сначала вычислятся самое правое

## [Взятие остатка %](http://learn.javascript.ru/operators" \l "%D0%B2%D0%B7%D1%8F%D1%82%D0%B8%D0%B5-%D0%BE%D1%81%D1%82%D0%B0%D1%82%D0%BA%D0%B0)

Оператор взятия остатка % интересен тем, что, несмотря на обозначение, никакого отношения к процентам не имеет.

Его результат a % b – это остаток от деления a на b.

Например:

alert( 5 % 2 ); // 1, остаток от деления 5 на 2

alert( 8 % 3 ); // 2, остаток от деления 8 на 3

alert( 6 % 3 ); // 0, остаток от деления 6 на 3

## [Инкремент/декремент: ++, --](http://learn.javascript.ru/operators" \l "%D0%B8%D0%BD%D0%BA%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D1%82-%D0%B4%D0%B5%D0%BA%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D1%82)

Одной из наиболее частых операций в JavaScript, как и во многих других языках программирования, является увеличение или уменьшение переменной на единицу.

**Инкремент** ++ увеличивает на 1:

var i = 2;

i++; // более короткая запись для i = i + 1.

alert(i); // 3

**Декремент** -- уменьшает на 1:

var i = 2;

i--; // более короткая запись для i = i - 1.

alert(i); // 1

## [Побитовые операторы](http://learn.javascript.ru/operators" \l "%D0%BF%D0%BE%D0%B1%D0%B8%D1%82%D0%BE%D0%B2%D1%8B%D0%B5-%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80%D1%8B)

AND(и) ( & )

OR(или) ( | )

XOR(побитовое исключающее или) ( ^ )

NOT(не) ( ~ )

LEFT SHIFT(левый сдвиг) ( << )

RIGHT SHIFT(правый сдвиг) ( >> )

ZERO-FILL RIGHT SHIFT(правый сдвиг с заполнением нулями) ( >>> )

## [Сокращённая арифметика с присваиванием](http://learn.javascript.ru/operators" \l "%D1%81%D0%BE%D0%BA%D1%80%D0%B0%D1%89%D1%91%D0%BD%D0%BD%D0%B0%D1%8F-%D0%B0%D1%80%D0%B8%D1%84%D0%BC%D0%B5%D1%82%D0%B8%D0%BA%D0%B0-%D1%81-%D0%BF%D1%80%D0%B8%D1%81%D0%B2%D0%B0%D0%B8%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%D0%BC)

Часто нужно применить оператор к переменной и сохранить результат в ней же, например:

var n = 2;

n = n + 5;

n = n \* 2;

Эту запись можно укоротить при помощи совмещённых операторов, вот так:

var n = 2;

n += 5; // теперь n=7 (работает как n = n + 5)

n \*= 2; // теперь n=14 (работает как n = n \* 2)

alert( n ); // 14

## [Оператор запятая](http://learn.javascript.ru/operators" \l "%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80-%D0%B7%D0%B0%D0%BF%D1%8F%D1%82%D0%B0%D1%8F)

Один из самих еобичних операторов ето запятая - ”,”

Ее можна визать очень легко

var a = (5, 6);

alert( a );