**ОПЕРАТОРЫ**

Переменные и различные величины взаимодействуют между собой с помощью операторов.

Оператор – это символ(ы) или ключевое слово, благодаря которому производятся некоторые виды вычислений с участием одного или нескольких значений. Значения, располагающиеся слева и справа от оператора, называются операндами. Оператор с одним операндом называется унарным, с двумя – бинарным, с тремя – тернарным.

В синтаксис языка JavaScript включены все основные математические операторы (+ — / \*), а также многие другие.

Присваивание =   
Присваивание ( = ) — используется для записи значения или вычисленного выражения в переменную.

let calculated = (12 + 8) \* 2; // 40

var userName = "Anakin Skywalker";

При наличии в выражении нескольких операторов присваивания они вычисляются справа налево.

let total = sum = 5 + 50 \* 2; // 105

Одна строка может содержать несколько последовательных присвоений.

let userLogin, adminLogin;

userLogin = adminLogin = "admin";

Если необходимо провести операцию с одной и той же переменной:

let number = 10;

number = number + 10; // 20

Можно использовать сокращенный синтаксис присваивания:

number += 10; // 20

Оператор += работает и с числами, и со строками. Для числовых операндов он выполняет сложение и присваивание, а для строковых – конкатенацию (соединение) и присваивание.

let name = "John";

name +=" Smith"; //John Smith

Бинарные операторы

Бинарные операторы применяются для двух операндов (элементов выражения).  
Бинарные операторы ( + – \* / ) производят арифметические операции с числами

let divide = 50 / 25; // 2

let multiply = 7 \* 4; // 28

let addition = 10 + 5; // 15

let subtraction = 5 - 1;// 4

В случае применения операторов к разным типам данных будет выполнено преобразование в число.  
Преобразование происходит по следующим правилам:

* • Логические значения true и false преобразуются в 1 и 0 соответственно.
* • Числа возвращаются без изменения.
* • Значение null преобразуется в 0.
* • Значение undefined преобразуется в NaN.

Для строк действуют особые правила:

* • Если строка содержит только цифры с начальным знаком + или -либо без знака, она всегда преобразуется в целое десятичное число. Начальные нули игнорируются, например «0011» преобразуется в 11.
* • Если строка представляет собой число с плавающей точкой с начальным знаком + или — либо без знака, она преобразуется в соответствующее число с плавающей точкой (начальные нули также игнорируются).
* • Если строка представляет собой число в шестнадцатеричном формате, она преобразуется в соответствующее целое десятичное число.
* • Если строка пустая, она преобразуется в 0.
* • Если строка содержит что-то отличное от предыдущих вариантов, она преобразуется в NaN.

let multiply = "12" \* "7"; // 84

let subtraction = 12 - "text"; // NaN

let divide = "20" / 2; // 10

let booleans = true + true; // 2

let subtr ="10" - true; // 9

let addition = 10 + true; // 11

let additionStr = "10" + true; // "10true"

В случае применения бинарного оператора + к строкам, они объединяются в одну строку (конкатенация).

let string = "Test" + "string"; // Teststring

Если хотя бы один операнд строка и используется бинарный +, тогда всё преобразуется в строку.

let string = 12 + "text" + 17; // 12test17;

Остаток от деления ( % ) возвращает целый остаток от деления левого операнда на правый. Результат деления с остатком будет иметь тот же знак, что и первый операнд

12 % 5; // 2

6 % 3; // 0

Оператор\*\*(возведение в степень) имеет два операнда. Первый операнд является основанием степени, второй операнд – показателем степени, в результате оператор возвращает основание, возведённое в указанную степень:

2 \*\* 4; // 16

Унарный оператор

Унарным называется оператор, который применяется к одному операнду.

Унарный оператор +, то есть применяемый к одному операнду, преобразовывает строки к числу.

let one = +"50"; // 50

let two = 12 + "70"; // "1270"

let three = 12 + +"70"; // 82

Унарный оператор минус «-» меняет знак числа на противоположный:

var x = 1;

x = -x;

console.log( x ); // -1

Инкремент и декремент

Инкремент ( ++ ) увеличивает значение переменной на 1.  
Декремент ( — ) уменьшает значение переменной на 1.  
Если значение операнда не является числом, оператор автоматически преобразует его в число, увеличивает на единицу и возвращает результат, который присваивается обратно операнду.

let number = 10;

number++; // 11

number--; // 10

Инкремент и декремент имеют две формы – постфиксную (оператор ставится после операнда)

let i = 1;

let oldValue = i++;

console.log(oldValue); // 1

и префиксную (оператор ставится перед операндом).

let i = 1;

let newValue = ++i;

console.log(newValue); // 2

Если они используются в постфиксной форме, то сначала возвращается исходное значение операнда, и только затем значение операнда увеличивается или уменьшается на единицу.

Операторы сравнения

Результатом выполнения всех операторов сравнения является значение логического типа true или false.

* • больше/меньше: a > b, a < b;
* • больше/меньше или равно: a >= b, a <= b;
* • равно: a == b;
* • строгое равенство: a === b;
* • не равно: a != b;
* • строгое «не равно»: a !== b.

Во всех случаях, кроме === и !==, при сравнении значений сначала происходит приведение типов (преобразование к числу).

15 > 5; // true

15 >= 14; // true

2 == 2; // true

2 != "2"; // false

2 == "2"; // true

"15" < 25; // true ("15"-> 15)

true > 0; // true (true -> 1)

true > 5; // false

false > 1; // false

false < 1; // true

false == 0; // true

null == 0; // false

null == null; // true

Сравнения строк происходит побуквенно (первая буква с первой, вторая со второй и т.д.)  
Каждый символ имеет числовой код, который и сравнивается. JavaScript использует кодировку Unicode.

"d" < "x"; // true

"d" != "x"; // true

"d" > "Z"; // true

"John" > "Smith"; // false

"15" > "5"; // false

У маленьких (строчных) букв, как правило, код больше чем у прописных. Для сравнения строк обычно используются только операторы равенства.

Операторы равенства

В не строгом равенстве (==) разные типы сначала приводятся к одному типу, а потом сравниваются.  
Преобразование типов выполняется по следующим правилам:

* • Если одно значение является числом, а другое – строкой, то строка преобразуется в число.
* • Если один из операндов имеет значение true, оно преобразуется в число 1, если значение false – в число 0.
* • Если значение ссылочного типа сравнивается с числом или строкой, оператор попытается преобразовать его к примитивному значению.

Значения считаются равными:

* • Если одно значение null, а другое undefined.
* • Если оба значения являются null или undefined.
* • Если оба значения являются true или false.
* • Если оба значения являются числами с одним и тем набором цифр (0 и -0 равны).
* • Если оба значения являются строками, и они содержат одни и те же 16-битные последовательности.
* • Если оба операнда ссылаются на один и тот же объект, массив или функцию.

Значения считаются не равными:

* • Если одно или оба значения являются NaN.
* • Если строки отличаются содержимым.
* • Если две строки выглядят одинаково, но содержат различные последовательности 16-битных значений.
* • Если оба операнда ссылаются на разные объекты, массивы или функции.

console.log( 0 == false ); // true

console.log( 0 == '' ); // true

console.log( '' == false ); // true

console.log( "12" == 12 ); // true

console.log( "12a" == 12 ); // false

console.log( "John" == "John" ); // true

console.log( "John" == "john" ); // false

Оператор === (строго равно) производит сравнение значений операндов без преобразования типов, должен совпасть и тип и значение.  
Значения считаются равными:

* • Если оба значения являются null или undefined.
* • Если оба значения являются true или false.
* • Если оба значения являются числами с одним и тем набором цифр (0 и -0 равны).
* • Если оба значения являются строками и они содержат одни и те же 16-битные последовательности.
* • Если оба операнда ссылаются на один и тот же объект, массив или функцию.

Значения считаются не равными:

* • Если два значения имеют различные типы.
* • Если одно или оба значения являются NaN.
* • Если строки отличаются содержимым.
* • Если две строки выглядят одинаково, но содержат различные последовательности 16-битных значений.

console.log('7' === 7); // false

console.log(null === null); // true

console.log(undefined === undefined); // true

console.log((2 + 2) === 4); // true

console.log(0 === -0); // true

console.log( "John" == "John" ); // true

console.log( "John" == "john" ); // false

Сравнение с null и undefined:

* • при математических операциях null преобразовывается к 0, а undefined – NaN;
* • null и undefined равны «==» только между собой, но не равны никакому другому значению.

null == 0; // false

undefined == 0; // false

null == undefined; // true

Логические операторы

Оператор || (логическое ИЛИ)  
Если хотя бы один из аргументов выражения «вычисляется» как true, то все выражение вычисляется как true.

true || false; // true

false || true; // true

false || false; // false

Сначала вычисляется значение левого операнда. Если операнд не логического, а другого (числового, строчного и т.д.) типа, то он для вычисления приводится к логическому (к true либо к false).

"test" || 0; // интерпретируется как true || false

null || true; // false || true

let data = 350;

let flag = true;

data < 10 || data == 350 || flag // false || true || true

Если левый операнд имеет значение true или значение, которое может быть преобразовано в true, то возвращается значение левого операнда, при этом значение правого операнда вычисляться не будет. Если левый операнд имеет значение false или значение, которое может быть преобразовано в false, оператор переходит к вычислению значения правого операнда, и возвращает значение правого операнда.  
То есть само выражение вернет именно тот элемент, который был «вычислен» как true.

console.log("test" || 0); // выводится "test"

console.log(null || true); // выводится true

console.log(null || "done" || false); // выводится done

Если все значения операндов интерпретируются как false, то возвращается последнее значение выражения.

let data = 15;

let empty = "";

console.log(null || false || data); // 15

console.log(null || false || empty); // ""

При помощи оператора || можно определять значения переменных. В переменную запишется первое значение, которое интерпретируется как true либо последнее значение.

let result = 0 || "Some text";

result; // "Some text";

Пример использования оператора ||:

let a = 0;

let b = "";

if (a || b) {

console.log("True.");

} else {

console.log("False.");

}

let a = 1;

let b = "";

if (a || b) {

console.log("True."); // "True."

} else {

console.log("False.");

}

Если переменная a или b (хотя бы одна из) имеет истинное значение (значение интерпретируется как true), то вывести сообщение в консоль.

Оператор && (логическое И) сначала вычисляет значение левого операнда. Если левый операнд имеет значение false или значение, которое может быть преобразовано в false, то возвращается значение левого операнда, при этом значение правого операнда вычисляться не будет иначе если левый операнд имеет значение true или значение, которое может быть преобразовано в true, оператор переходит к вычислению значения правого операнда, и возвращает значение правого операнда.

true && false; //false

true && true; //true

Как только хотя бы один операнд вычисляется как false, он же и возвращается. Если все операнды имеет значение true, то возвращается последний.

console.log("test" && 0); // 0

console.log(null && true); // null

console.log(true && 15 && "string"); // "string"

console.log(true && null && "string"); // null

let isReady = "done";

let open = true;

console.log(open && isReady); // "done"

Пример использования оператора &&:

let data = 350;

let flag = true;

if (data == 100 && flag) {

console.log("True.");

}

let data = 350;

let flag = true;

if (data > 100 && flag) {

console.log("True."); // "True."

}

Если переменные data > 100 и flag имеют истинное значение (значение интерпретируется как true), то вывести сообщение в консоль.

Оператор ! (логическое НЕ) преобразует (если необходимо) значение своего операнда к логическому типу, затем инвертирует это значение в противоположное и возвращает полученное в результате инвертирования булево значение.

console.log(!0); // true

Ноль приводится к логическому типу – false.  
Полученный результат меняется на противоположный – true.

console.log(!"true"); // false

Двойной оператор НЕ (!!) просто преобразует операнд к логическому типу, путем двойного последовательного инвертирования.

console.log(!!null); // false

console.log(!!"Some text"); // true

Комбинированное использование логических операторов.

console.log("True" || 0 && true || !0); // "True" (true || false && true || true)

Так как приоритет у оператора && больше, чем у оператора ||, то сначала выполняется 0 && true, поэтому в результате интерпретируется как:

console.log("True" || 0 && true || !0); // (true || false || true)