# Логические операторы

В JavaScript есть 7 логических операторов:

* || (ИЛИ)
  + ||= (Оператор логического присваивания ИЛИ)
* && (И)
  + &&= (Оператор логического присваивания И)
* ! (НЕ)
* ?? (Оператор нулевого слияния)
  + ??= (Оператор нулевого присваивания

данные операторы могут применяться к значениям любых типов. Полученные результаты также могут иметь различный тип.

**|| (ИЛИ)**

Существует всего четыре возможные логические комбинации:

alert( true || true ); // true

alert( false || true ); // true

alert( true || false ); // true

alert( false || false ); // false

результат операций всегда равен true, за исключением случая, когда оба аргумента false.

Если значение не логического типа, то оно к нему приводится в целях вычислений.

Обычно оператор || используется в if для проверки истинности любого из заданных условий.

К примеру:

let hour = 9;

if (hour < 10 || hour > 18) {

alert( 'Офис закрыт.' );

}

Больше чем 2 проверки в ИЛИ

При выполнении ИЛИ || с несколькими значениями:

result = value1 || value2 || value3;

Оператор || выполняет следующие действия:

* Вычисляет операнды слева направо.
* Каждый операнд конвертирует в логическое значение. Если результат true, останавливается и возвращает исходное значение этого операнда.
* Если все операнды являются ложными (false), возвращает последний из них.

Значение возвращается в исходном виде, без преобразования.

Другими словами, цепочка ИЛИ || возвращает первое истинное значение или последнее, если такое значение не найдено.

интересное применение оператора по сравнению с «чистым, традиционным, только булевым ИЛИ:

1. **Получение первого истинного значения из списка переменных или выражений.**

Например, есть переменные firstName, lastName и nickName, все они необязательные (т.е. они могут быть неопределенными или иметь ложные значения).

Давайте воспользуемся оператором ИЛИ ||, чтобы выбрать ту переменную, в которой есть данные, и показать её (или «Аноним», если ни в одной переменной данных нет):

let firstName = "";

let lastName = "";

let nickName = "Суперкодер";

alert( firstName || lastName || nickName || "Аноним"); // Суперкодер

Если бы все переменные были ложными, в качестве результата мы бы наблюдали "Аноним".

**||= (Логическое присваивание ИЛИ)**

Вот его синтаксис:

a ||= b;

Оператор ||= принимает два операнда и выполняет следующие действия:

* Вычисляет операнды слева направо.
* Конвертирует a в логическое значение.
* Если a ложно, присваивает a значение b.

Теперь давайте перепишем a ||= b под вид «сокращённого вычисления»:

Мы уже знаем, что ИЛИ || возвращает первое истинное значение, поэтому, если a является таковым, вычисление до правой части выражения не дойдёт.

a || (a = b);

Вот пример с очевидным использованием оператора ||=:

let johnHasCar = false;

johnHasCar ||= "У Джона нет машины!"; // то же самое, что false || (johnHasCar = "...")

alert( johnHasCar ); // "У Джона нет машины!"

…А здесь происходит преобразование к логическому значению:

let manufacturer = ""; // оператор ||= преобразует пустую строку "" к логическому значению false

manufacturer ||= "Неизвестный производитель"; // то же самое, что false || (manufacturer = "...")

alert( manufacturer ); // "Неизвестный производитель"

Оператор логического присваивания ИЛИ ||= – это «[синтаксический сахар](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%81%D0%B0%D1%85%D0%B0%D1%80)», добавленный в язык в качестве более короткого варианта записи if-выражений с присваиванием.

**&& (И)**

В традиционном программировании И возвращает true, если оба аргумента истинны, а иначе – false:

alert( true && true ); // true

alert( false && true ); // false

alert( true && false ); // false

alert( false && false ); // false

Допускается любое значение допускается в качестве операнда И.

**И «&&» находит первое ложное значение**

При нескольких подряд операторах И:

result = value1 && value2 && value3;

Оператор && выполняет следующие действия:

* Вычисляет операнды слева направо.
* Каждый операнд преобразует в логическое значение. Если результат false, останавливается и возвращает исходное значение этого операнда.
* Если все операнды были истинными, возвращается последний.

Другими словами, И возвращает первое ложное значение. Или последнее, если ничего не найдено.

Вышеуказанные правила схожи с поведением ИЛИ. Разница в том, что И возвращает первое ложное значение, а ИЛИ –  первое истинное.

// Если первый операнд истинный,

// И возвращает второй:

alert( 1 && 0 ); // 0

alert( 1 && 5 ); // 5

// Если первый операнд ложный,

// И возвращает его. Второй операнд игнорируется

alert( null && 5 ); // null

alert( 0 && "какая-то строка" ); // 0

Можно передать несколько значений подряд. В таком случае возвратится первое «ложное» значение, на котором остановились вычисления.

alert( 1 && 2 && null && 3 ); // null

Когда все значения верны, возвращается последнее

alert( 1 && 2 && 3 ); // 3

**Приоритет оператора && больше, чем у ||**

Приоритет оператора И && больше, чем ИЛИ ||, так что он выполняется раньше.

Таким образом, код a && b || c && d по существу такой же, как если бы выражения && были в круглых скобках: (a && b) || (c && d).

**Не заменяйте if на || или &&**

Иногда люди используют оператор И && как «более короткий способ записи if-выражения».

Например:

let x = 1;

(x > 0) && alert( 'x больше нуля!' );

Инструкция в правой части && будет выполнена только в том случае, если вычисление дойдет до нее. То есть, только если (x > 0) истинно.

Таким образом, мы имеем аналог для следующего кода:

let x = 1;

if (x > 0) alert( 'x больше нуля!' );

Несмотря на то, что вариант с && кажется более коротким, if более нагляден и, как правило, более читабелен. Поэтому мы рекомендуем использовать каждую конструкцию по назначению: использовать if, если нам нужно if, и использовать &&, если нам нужно И.

**&&= (Логическое присваивание И)**

Вот его синтаксис:

a &&= b;

Принцип действия &&= практически такой же, как и у оператора логического присваивания ИЛИ ||=. Единственное отличие заключается в том, что &&= присвоит a значение b только в том случае, если a истинно.

Концепция оператора логического присваивания И &&= также основывается на «сокращённом вычислении»:

a && (a = b);

Пример использования:

let greeting = "Привет"; // строка непустая, поэтому будет преобразована к логическому значению true оператором &&=

greeting &&= greeting + ", пользователь!"; // то же самое, что true && (greeting = greeting + "...")

alert( greeting ) // "Привет, пользователь!"

Так как оператор логического присваивания И &&= также как и ||= является «синтаксическим сахаром», мы можем без проблем переписать пример выше с использованием привычного для нас if:

let greeting = "Привет";

if (greeting) {

greeting = greeting + ", пользователь!"

}

alert( greeting ) // "Привет, пользователь!"

На практике, в отличие от ||=, оператор &&= используется достаточно редко – обычно, в комбинации с более сложными языковыми конструкциями, о которых мы будем говорить позже. Подобрать контекст для применения данного оператора – довольно непростая задача.

**! (НЕ)**

Оператор принимает один аргумент и выполняет следующие действия:

1. Сначала приводит аргумент к логическому типу true/false.
2. Затем возвращает противоположное значение.

В частности, двойное НЕ !! используют для преобразования значений к логическому типу, можно использовать как альтернативу Boolean:

alert( !!"непустая строка" ); // true

alert( !!null ); // false

То есть первое НЕ преобразует значение в логическое значение и возвращает обратное, а второе НЕ снова инвертирует его. В конце мы имеем простое преобразование значения в логическое.

Приоритет НЕ ! является наивысшим из всех логических операторов, поэтому он всегда выполняется первым, перед && или ||.