**Logical operators**

There are four logical operators in JavaScript: || (OR), && (AND), ! (NOT), ?? (Nullish Coalescing). Here we cover the first three, the ?? operator is in the next article.

Although they are called “logical”, they can be applied to values of any type, not only boolean. Their result can also be of any type.

Let’s see the details.

**[|| (OR)](https://javascript.info/logical-operators" \l "or)**

The “OR” operator is represented with two vertical line symbols:

result = a || b;

In classical programming, the logical OR is meant to manipulate boolean values only. If any of its arguments are true, it returns true, otherwise it returns false.

In JavaScript, the operator is a little bit trickier and more powerful. But first, let’s see what happens with boolean values.

There are four possible logical combinations:

alert( true || true ); // true

alert( false || true ); // true

alert( true || false ); // true

alert( false || false ); // false

As we can see, the result is always true except for the case when both operands are false.

If an operand is not a boolean, it’s converted to a boolean for the evaluation.

For instance, the number 1 is treated as true, the number 0 as false:

if (1 || 0) { // works just like if( true || false )

alert( 'truthy!' );

}

Most of the time, OR || is used in an if statement to test if *any* of the given conditions is true.

For example:

let hour = 9;

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

alert( 'The office is closed.' );

}

We can pass more conditions:

let hour = 12;

let isWeekend = true;

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

alert( 'The office is closed.' ); // it is the weekend

}

**[OR "||" finds the first truthy value](https://javascript.info/logical-operators" \l "or-finds-the-first-truthy-value)**

The logic described above is somewhat classical. Now, let’s bring in the “extra” features of JavaScript.

The extended algorithm works as follows.

Given multiple OR’ed values:

result = value1 || value2 || value3;

The OR || operator does the following:

* Evaluates operands from left to right.
* For each operand, converts it to boolean. If the result is true, stops and returns the original value of that operand.
* If all operands have been evaluated (i.e. all were false), returns the last operand.

A value is returned in its original form, without the conversion.

In other words, a chain of OR || returns the first truthy value or the last one if no truthy value is found.

For instance:

alert( 1 || 0 ); // 1 (1 is truthy)

alert( null || 1 ); // 1 (1 is the first truthy value)

alert( null || 0 || 1 ); // 1 (the first truthy value)

alert( undefined || null || 0 ); // 0 (all falsy, returns the last value)

This leads to some interesting usage compared to a “pure, classical, boolean-only OR”.

1. **Getting the first truthy value from a list of variables or expressions.**

For instance, we have firstName, lastName and nickName variables, all optional (i.e. can be undefined or have falsy values).

Let’s use OR || to choose the one that has the data and show it (or "Anonymous" if nothing set):

let firstName = "";

let lastName = "";

let nickName = "SuperCoder";

alert( firstName || lastName || nickName || "Anonymous"); // SuperCoder

If all variables were falsy, "Anonymous" would show up.

1. **Short-circuit evaluation.**

Another feature of OR || operator is the so-called “short-circuit” evaluation.

It means that || processes its arguments until the first truthy value is reached, and then the value is returned immediately, without even touching the other argument.

The importance of this feature becomes obvious if an operand isn’t just a value, but an expression with a side effect, such as a variable assignment or a function call.

In the example below, only the second message is printed:

true || alert("not printed");

false || alert("printed");

In the first line, the OR || operator stops the evaluation immediately upon seeing true, so the alert isn’t run.

Sometimes, people use this feature to execute commands only if the condition on the left part is falsy.

**[&& (AND)](https://javascript.info/logical-operators" \l "and)**

The AND operator is represented with two ampersands &&:

result = a && b;

In classical programming, AND returns true if both operands are truthy and false otherwise:

alert( true && true ); // true

alert( false && true ); // false

alert( true && false ); // false

alert( false && false ); // false

An example with if:

let hour = 12;

let minute = 30;

if (hour == 12 && minute == 30) {

alert( 'The time is 12:30' );

}

Just as with OR, any value is allowed as an operand of AND:

if (1 && 0) { // evaluated as true && false

alert( "won't work, because the result is falsy" );

}

**[AND “&&” finds the first falsy value](https://javascript.info/logical-operators" \l "and-finds-the-first-falsy-value)**

Given multiple AND’ed values:

result = value1 && value2 && value3;

The AND && operator does the following:

* Evaluates operands from left to right.
* For each operand, converts it to a boolean. If the result is false, stops and returns the original value of that operand.
* If all operands have been evaluated (i.e. all were truthy), returns the last operand.

In other words, AND returns the first falsy value or the last value if none were found.

The rules above are similar to OR. The difference is that AND returns the first *falsy* value while OR returns the first *truthy* one.

Examples:

// if the first operand is truthy,

// AND returns the second operand:

alert( 1 && 0 ); // 0

alert( 1 && 5 ); // 5

// if the first operand is falsy,

// AND returns it. The second operand is ignored

alert( null && 5 ); // null

alert( 0 && "no matter what" ); // 0

We can also pass several values in a row. See how the first falsy one is returned:

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

When all values are truthy, the last value is returned:

alert( 1 && 2 && 3 ); // 3, the last one

**Precedence of AND && is higher than OR ||**

The precedence of AND && operator is higher than OR ||.

So the code a && b || c && d is essentially the same as if the && expressions were in parentheses: (a && b) || (c && d).

**Don’t replace if with || or &&**

Sometimes, people use the AND && operator as a "shorter way to write if".

For instance:

let x = 1;

(x > 0) && alert( 'Greater than zero!' );

The action in the right part of && would execute only if the evaluation reaches it. That is, only if (x > 0) is true.

So we basically have an analogue for:

let x = 1;

if (x > 0) alert( 'Greater than zero!' );

Although, the variant with && appears shorter, if is more obvious and tends to be a little bit more readable. So we recommend using every construct for its purpose: use if if we want if and use && if we want AND.

**[! (NOT)](https://javascript.info/logical-operators" \l "not)**

The boolean NOT operator is represented with an exclamation sign !.

The syntax is pretty simple:

result = !value;

The operator accepts a single argument and does the following:

1. Converts the operand to boolean type: true/false.
2. Returns the inverse value.

For instance:

alert( !true ); // false

alert( !0 ); // true

A double NOT !! is sometimes used for converting a value to boolean type:

alert( !!"non-empty string" ); // true

alert( !!null ); // false

That is, the first NOT converts the value to boolean and returns the inverse, and the second NOT inverses it again. In the end, we have a plain value-to-boolean conversion.

There’s a little more verbose way to do the same thing – a built-in Boolean function:

alert( Boolean("non-empty string") ); // true

alert( Boolean(null) ); // false

The precedence of NOT ! is the highest of all logical operators, so it always executes first, before && or ||.

[**Tasks**](https://javascript.info/logical-operators#tasks)

**[What's the result of OR?](https://javascript.info/logical-operators" \l "what-s-the-result-of-or)**

**importance: 5**

What is the code below going to output?

alert( null || 2 || undefined );

**solution**

The answer is 2, that’s the first truthy value.

alert( null || 2 || undefined );

**[What's the result of OR'ed alerts?](https://javascript.info/logical-operators" \l "what-s-the-result-of-or-ed-alerts)**

**importance: 3**

What will the code below output?

alert( alert(1) || 2 || alert(3) );

**solution**

The answer: first 1, then 2.

alert( alert(1) || 2 || alert(3) );

The call to alert does not return a value. Or, in other words, it returns undefined.

1. The first OR || evaluates its left operand alert(1). That shows the first message with 1.
2. The alert returns undefined, so OR goes on to the second operand searching for a truthy value.
3. The second operand 2 is truthy, so the execution is halted, 2 is returned and then shown by the outer alert.

There will be no 3, because the evaluation does not reach alert(3).

**[What is the result of AND?](https://javascript.info/logical-operators" \l "what-is-the-result-of-and)**

**importance: 5**

What is this code going to show?

alert( 1 && null && 2 );

**solution**

The answer: null, because it’s the first falsy value from the list.

alert(1 && null && 2);

**[What is the result of AND'ed alerts?](https://javascript.info/logical-operators" \l "what-is-the-result-of-and-ed-alerts)**

**importance: 3**

What will this code show?

alert( alert(1) && alert(2) );

**solution**

The answer: 1, and then undefined.

alert( alert(1) && alert(2) );

The call to alert returns undefined (it just shows a message, so there’s no meaningful return).

Because of that, && evaluates the left operand (outputs 1), and immediately stops, because undefined is a falsy value. And && looks for a falsy value and returns it, so it’s done.

**[The result of OR AND OR](https://javascript.info/logical-operators" \l "the-result-of-or-and-or)**

**importance: 5**

What will the result be?

alert( null || 2 && 3 || 4 );

**solution**

The answer: 3.

alert( null || 2 && 3 || 4 );

The precedence of AND && is higher than ||, so it executes first.

The result of 2 && 3 = 3, so the expression becomes:

null || 3 || 4

Now the result is the first truthy value: 3.

**[Check the range between](https://javascript.info/logical-operators" \l "check-the-range-between)**

**importance: 3**

Write an if condition to check that age is between 14 and 90 inclusively.

“Inclusively” means that age can reach the edges 14 or 90.

**solution**

if (age >= 14 && age <= 90)

**[Check the range outside](https://javascript.info/logical-operators" \l "check-the-range-outside)**

**importance: 3**

Write an if condition to check that age is NOT between 14 and 90 inclusively.

Create two variants: the first one using NOT !, the second one – without it.

**solution**

The first variant:

if (!(age >= 14 && age <= 90))

The second variant:

if (age < 14 || age > 90)

**[A question about "if"](https://javascript.info/logical-operators" \l "a-question-about-if)**

**importance: 5**

Which of these alerts are going to execute?

What will the results of the expressions be inside if(...)?

if (-1 || 0) alert( 'first' );

if (-1 && 0) alert( 'second' );

if (null || -1 && 1) alert( 'third' );

**solution**

The answer: the first and the third will execute.

Details:

// Runs.

// The result of -1 || 0 = -1, truthy

if (-1 || 0) alert( 'first' );

// Doesn't run

// -1 && 0 = 0, falsy

if (-1 && 0) alert( 'second' );

// Executes

// Operator && has a higher precedence than ||

// so -1 && 1 executes first, giving us the chain:

// null || -1 && 1 -> null || 1 -> 1

if (null || -1 && 1) alert( 'third' );

**[Check the login](https://javascript.info/logical-operators" \l "check-the-login)**

**importance: 3**

Write the code which asks for a login with prompt.

If the visitor enters "Admin", then prompt for a password, if the input is an empty line or Esc – show “Canceled”, if it’s another string – then show “I don’t know you”.

The password is checked as follows:

* If it equals “TheMaster”, then show “Welcome!”,
* Another string – show “Wrong password”,
* For an empty string or cancelled input, show “Canceled”

The schema:

Please use nested if blocks. Mind the overall readability of the code.

Hint: passing an empty input to a prompt returns an empty string ''. Pressing ESC during a prompt returns null.

[Run the demo](https://javascript.info/logical-operators)

**solution**

let userName = prompt("Who's there?", '');

if (userName === 'Admin') {

let pass = prompt('Password?', '');

if (pass === 'TheMaster') {

alert( 'Welcome!' );

} else if (pass === '' || pass === null) {

alert( 'Canceled' );

} else {

alert( 'Wrong password' );

}

} else if (userName === '' || userName === null) {

alert( 'Canceled' );

} else {

alert( "I don't know you" );

}

Note the vertical indents inside the if blocks. They are technically not required, but make the code more readable.

**[Comments](https://javascript.info/logical-operators" \l "comments)**

read this before commenting…