**Nullish coalescing operator '??'**

**A recent addition**

This is a recent addition to the language. Old browsers may need [polyfills](https://javascript.info/polyfills).

The nullish coalescing operator is written as two question marks ??.

As it treats null and undefined similarly, we’ll use a special term here, in this article. For brevity, we’ll say that a value is “defined” when it’s neither null nor undefined.

The result of a ?? b is:

* if a is defined, then a,
* if a isn’t defined, then b.

In other words, ?? returns the first argument if it’s not null/undefined. Otherwise, the second one.

The nullish coalescing operator isn’t anything completely new. It’s just a nice syntax to get the first “defined” value of the two.

We can rewrite result = a ?? b using the operators that we already know, like this:

result = (a !== null && a !== undefined) ? a : b;

Now it should be absolutely clear what ?? does. Let’s see where it helps.

The common use case for ?? is to provide a default value.

For example, here we show user if its value isn’t null/undefined, otherwise Anonymous:

let user;

alert(user ?? "Anonymous"); // Anonymous (user is undefined)

Here’s the example with user assigned to a name:

let user = "John";

alert(user ?? "Anonymous"); // John (user is not null/undefined)

We can also use a sequence of ?? to select the first value from a list that isn’t null/undefined.

Let’s say we have a user’s data in variables firstName, lastName or nickName. All of them may be not defined, if the user decided not to fill in the corresponding values.

We’d like to display the user name using one of these variables, or show “Anonymous” if all of them are null/undefined.

Let’s use the ?? operator for that:

let firstName = null;

let lastName = null;

let nickName = "Supercoder";

// shows the first defined value:

alert(firstName ?? lastName ?? nickName ?? "Anonymous"); // Supercoder

**[Comparison with ||](https://javascript.info/nullish-coalescing-operator" \l "comparison-with)**

The OR || operator can be used in the same way as ??, as it was described in the [previous chapter](https://javascript.info/logical-operators#or-finds-the-first-truthy-value).

For example, in the code above we could replace ?? with || and still get the same result:

let firstName = null;

let lastName = null;

let nickName = "Supercoder";

// shows the first truthy value:

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

Historically, the OR || operator was there first. It’s been there since the beginning of JavaScript, so developers were using it for such purposes for a long time.

On the other hand, the nullish coalescing operator ?? was added to JavaScript only recently, and the reason for that was that people weren’t quite happy with ||.

The important difference between them is that:

* || returns the first *truthy* value.
* ?? returns the first *defined* value.

In other words, || doesn’t distinguish between false, 0, an empty string "" and null/undefined. They are all the same – falsy values. If any of these is the first argument of ||, then we’ll get the second argument as the result.

In practice though, we may want to use default value only when the variable is null/undefined. That is, when the value is really unknown/not set.

For example, consider this:

let height = 0;

alert(height || 100); // 100

alert(height ?? 100); // 0

* The height || 100 checks height for being a falsy value, and it’s 0, falsy indeed.
  + so the result of || is the second argument, 100.
* The height ?? 100 checks height for being null/undefined, and it’s not,
  + so the result is height “as is”, that is 0.

In practice, the zero height is often a valid value, that shouldn’t be replaced with the default. So ?? does just the right thing.

**[Precedence](https://javascript.info/nullish-coalescing-operator" \l "precedence)**

The precedence of the ?? operator is the same as ||. They both equal 3 in the [MDN table](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Operator_Precedence#Table).

That means that, just like ||, the nullish coalescing operator ?? is evaluated before = and ?, but after most other operations, such as +, \*.

So we may need to add parentheses in expressions like this:

let height = null;

let width = null;

// important: use parentheses

let area = (height ?? 100) \* (width ?? 50);

alert(area); // 5000

Otherwise, if we omit parentheses, then as \* has the higher precedence than ??, it would execute first, leading to incorrect results.

// without parentheses

let area = height ?? 100 \* width ?? 50;

// ...works this way (not what we want):

let area = height ?? (100 \* width) ?? 50;

**[Using ?? with && or ||](https://javascript.info/nullish-coalescing-operator" \l "using-with-or)**

Due to safety reasons, JavaScript forbids using ?? together with && and || operators, unless the precedence is explicitly specified with parentheses.

The code below triggers a syntax error:

let x = 1 && 2 ?? 3; // Syntax error

The limitation is surely debatable, it was added to the language specification with the purpose to avoid programming mistakes, when people start to switch from || to ??.

Use explicit parentheses to work around it:

let x = (1 && 2) ?? 3; // Works

alert(x); // 2

**[Summary](https://javascript.info/nullish-coalescing-operator" \l "summary)**

* The nullish coalescing operator ?? provides a short way to choose the first “defined” value from a list.

It’s used to assign default values to variables:

// set height=100, if height is null or undefined

height = height ?? 100;

* The operator ?? has a very low precedence, only a bit higher than ? and =, so consider adding parentheses when using it in an expression.
* It’s forbidden to use it with || or && without explicit parentheses.

**[Comments](https://javascript.info/nullish-coalescing-operator" \l "comments)**

read this before commenting…