Null coalescing operator

The **null coalescing operator** is a binary operator that is part of the syntax for a basic conditional expression in several programming languages, such as (in alphabetical order): C#[1] since version 2.0,[2] Dart[3] since version 1.12.0,[4] PHP since version 7.0.0,[5] Perl since version 5.10 as *logical defined-or*,[6] PowerShell since 7.0.0,[7] and Swift[8] as *nil-coalescing operator*. It is most commonly written as `x ?? y`, but varies across programming languages.

While its behavior differs between implementations, the null coalescing operator generally returns the result of its left-most operand if it exists and is not null, and otherwise returns the right-most operand. This behavior allows a default value to be defined for cases where a more specific value is not available.

Like the binary Elvis operator, usually written as `x ?: y`, the null coalescing operator is a short-circuiting operator and thus does not evaluate the second operand if its value is not used, which is significant if its evaluation has side-effects.

Examples by languages

Bourne-like shells

In Bourne shell (and derivatives), "If *parameter* is unset or null, the expansion of *word* is substituted. Otherwise, the value of *parameter* is substituted":[9]

```
#supplied_title='supplied title' # Uncomment this line to use the supplied title
title=${supplied_title:-'Default title'}
echo "$title" # prints: Default title
```

C#

In C#, the null coalescing operator is `??`.

It is most often used to simplify expressions as follows:

```
possiblyNullValue ?? valueIfNull
```

For example, if one wishes to implement some C# code to give a page a default title if none is present, one may use the following statement:

```
string pageTitle = suppliedTitle ?? "Default Title";
```

instead of the more verbose

```
string pageTitle = (suppliedTitle != null) ? suppliedTitle : "Default Title";
```

or

```
string pageTitle;

if (suppliedTitle != null)
{
    pageTitle = suppliedTitle;
}
else
{
    pageTitle = "Default Title";
}
```

The three forms result in the same value being stored into the variable named `pageTitle`.

`suppliedTitle` is referenced only once when using the `??` operator, and twice in the other two code examples.

The operator can also be used multiple times in the same expression:

```
return some_Value ?? some_Value2 ?? some_Value3;
```

Once a non-null value is assigned to number, or it reaches the final value (which may or may not be null), the expression is completed.

If, for example, a variable should be changed to another value if its value evaluates to null, since C# 8.0 the `??=` null coalescing assignment operator can be used:

```
some_Value ??= some_Value2;
```

Which is a more concise version of:

```
some_Value = some_Value ?? some_Value2;
```

In combination with the null-conditional operator `?.` or the null-conditional element access operator `?[]` the null coalescing operator can be used to provide a default value if an object or an object's member is null. For example, the following will return the default title if either the `page` object is null or `page` is not null but its `Title` property is:

```
string pageTitle = page?.Title ?? "Default Title";
```

CFML

As of ColdFusion 11,[10] Railo 4.1,[11] CFML supports the null coalescing operator as a variation of the ternary operator, `?:`. It is functionally and syntactically equivalent to its C# counterpart, above. Example:

```
possiblyNullValue ?: valueIfNull
```

Freemarker

Missing values in Apache FreeMarker will normally cause exceptions. However, both missing and null values can be handled, with an optional default value:[12]

```
${missingVariable!"defaultValue"}
```

or, to leave the output blank:

```
${missingVariable!}
```

JavaScript

JavaScript's nearest operator is `??`, the "nullish coalescing operator", which was added to the standard in ECMAScript's 11th edition.[13] In earlier versions, it could be used via a Babel plugin, and in TypeScript. It evaluates its left-hand operand and, if the result value is *not* "nullish" (`null` or `undefined`), takes that value as its result; otherwise, it evaluates the right-hand operand and takes the resulting value as its result.

In the following example, `a` will be assigned the value of `b` if the value of `b` is not `null` or `undefined`, otherwise it will be assigned 3.

```
const a = b ?? 3;
```

Before the nullish coalescing operator, programmers would use the logical OR operator (`||`). But where `??` looks specifically for `null` or `undefined`, the `||` operator looks for any falsy value: `null`, `undefined`, `""`, `0`, `NaN`, and of course, `false`.

In the following example, `a` will be assigned the value of `b` if the value of `b` is truthy, otherwise it will be assigned 3.

```
const a = b || 3;
```

Kotlin

Kotlin uses the `?:` operator.[14] This is an unusual choice of symbol, given that `?:` is typically used for the Elvis operator, not null coalescing, but it was inspired by Groovy (programming language) where null is considered false.

```
val title = suppliedTitle ?: "Default title"
```

Objective-C

In Obj-C, the nil coalescing operator is `?:`. It can be used to provide a default for nil references:

```
id value = valueThatMightBeNil ?: valueIfNil;
```

This is the same as writing

```
id value = valueThatMightBeNil ? valueThatMightBeNil : valueIfNil;
```

Perl

In Perl (starting with version 5.10), the operator is `//` and the equivalent Perl code is:

```
$possibly_null_value // $value_if_null
```

The *possibly_null_value* is evaluated as *null* or *not-null* (in Perl terminology, *undefined* or *defined*). On the basis of the evaluation, the expression returns either *value_if_null* when *possibly_null_value* is null, or *possibly_null_value* otherwise. In the absence of side-effects this is similar to the way ternary operators (**`?:`** statements) work in languages that support them. The above Perl code is equivalent to the use of the ternary operator below:

```
defined($possibly_null_value) ? $possibly_null_value : $value_if_null
```

This operator's most common usage is to minimize the amount of code used for a simple null check.

Perl additionally has a `//=` assignment operator, where

```
$a //= $b
```

is largely equivalent to:

```
$a = $a // $b
```

This operator differs from Perl's older `||` and `||=` operators in that it considers *definedness,* not *truth*. Thus they behave differently on values that are false but defined, such as 0 or "" (a zero-length string):

```
$a = 0;
$b = 1;
$c = $a // $b;  # $c = 0
$c = $a || $b;  # $c = 1
```

PHP

PHP 7.0 introduced[15] a null-coalescing operator with the `??` syntax. This checks strictly for NULL or a non-existent variable/array index/property. In this respect, it acts similarly to PHP's `isset()` pseudo-function:

```
$name = $request->input['name'] ?? $request->query['name'] ?? 'default name';

/* Equivalent to */

if (isset($request->input['name'])) {
    $name = $request->input['name'];
} elseif (isset($request->query['name'])) {
    $name = $request->query['name'];
} else {
    $name = 'default name';
}
```

```
$user = $this->getUser() ?? $this->createGuestUser();

/* Equivalent to */

$user = $this->getUser();

if ($user === null) {
    $user = $this->createGuestUser();
}
```

```
$pageTitle = $title ?? 'Default Title';

/* Equivalent to */

$pageTitle = isset($title) ? $title : 'Default Title';
```

Version 7.4 of PHP introduced the Null Coalescing Assignment Operator with the `??=` syntax:[16]

```
// The following lines are doing the same
$this->request->data['comments']['user_id'] = $this->request->data['comments']['user_id'] ?? 'value';
// Instead of repeating variables with long names, the equal coalesce operator is used
$this->request->data['comments']['user_id'] ??= 'value';
```

PowerShell

Since PowerShell 7, the `??` null coalescing operator provides this functionality.[7]

```
$myVar = $null
$x = $myVar ?? "something" # assigns "something"
```

R

Since R version 4.4.0 the `%||%` operator is included in base R (previously it was a feature of some packages like *rlang*).[17]

```
> NULL %||% 2
[1] 2
```

Rust

While there's no `null` in Rust, tagged unions are used for the same purpose. For example, `Result<T, E>` or `Option<T>`. Any type implementing the Try trait can be unwrapped.

`unwrap_or()` serves a similar purpose as the null coalescing operator in other languages. Alternatively, `unwrap_or_else()` can be used to use the result of a function as a default value.

```
// Option
// An Option can be either Some(value) or None
Some(1).unwrap_or(0); // evaluates to 1
None.unwrap_or(0); // evaluates to 0
None.unwrap_or_else(get_default); // evaluates to the result of calling the function get_default

// Result
// A Result can be either Ok(value) or Err(error)
Ok(1).unwrap_or(0); // evaluates to 1
Err("oh no").unwrap_or(1); // evaluates to 1
```

SQL

In Oracle's PL/SQL, the NVL() function provides the same outcome:

```
NVL(possibly_null_value, 'value if null');
```

In SQL Server/Transact-SQL there is the ISNULL function that follows the same prototype pattern:

```
ISNULL(possibly_null_value, 'value if null');
```

Attention should be taken to not confuse *ISNULL* with *IS NULL* – the latter serves to evaluate whether some contents are defined to be *NULL* or not.

The ANSI SQL-92 standard includes the COALESCE function implemented in Oracle,[18] SQL Server,[19] PostgreSQL,[20] SQLite[21] and MySQL.[22] The COALESCE function returns the first argument that is not null. If all terms are null, returns null.

```
COALESCE(possibly_null_value[, possibly_null_value, ...]);
```

The difference between ISNULL and COALESCE is that the type returned by ISNULL is the type of the leftmost value while COALESCE returns the type of the first non-null value.

Swift

In Swift, the nil coalescing operator is ??. It is used to provide a default when unwrapping an optional type:

```
optionalValue ?? valueIfNil
```

For example, if one wishes to implement some Swift code to give a page a default title if none is present, one may use the following statement:

```
var suppliedTitle: String? = ...
var pageTitle: String = suppliedTitle ?? "Default Title"
```

instead of the more verbose

```
var pageTitle: String = (suppliedTitle != nil) ? suppliedTitle! : "Default Title";
```

See also

- ?: (conditional)
- Elvis operator (binary ?:)
- Null-conditional operator
- Operator (computer programming)

References

1. "?? and ??= operators - the null-coalescing operators" (https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/null-coalescing-operator). *Microsoft Learn*. 2023-07-27.
2. "ECMA-334, 3rd edition, June 2005" (https://www.ecma-international.org/wp-content/uploads/ECMA-334_3rd_edition_june_2005.pdf) (PDF). *ecma-international.org*. Ecma International. June 2005. p. 63.
3. "Conditional expression" (https://dart.dev/language/operators#conditional-expressions). *Dart*.
4. "Dart SDK Changelog, 1.12.0" (https://github.com/dart-lang/sdk/blob/main/CHANGELOG.md#1120---2015-08-31). *GitHub*. 2015-08-31.
5. "PHP 7.0.0 Released" (http://php.net/archive/2015.php#id2015-11-12-1). *PHP*. 2015-11-12.
6. "perlop - Perl expressions: operators, precedence, string literals" (https://perldoc.perl.org/perlop#Logical-Defined-Or). *Perldoc Browser*.
7. "PowerShell 7 Preview 5" (https://devblogs.microsoft.com/powershell/powershell-7-preview-5/). *PowerShell*. 2019-10-23. Retrieved 2020-02-15.
8. "The Swift Programming Language (Swift 5): Basic Operators: Nil-Coalescing Operator" (https://docs.swift.org/swift-book/documentation/the-swift-programming-language/basicoperators/#Nil-Coalescing-Operator). *docs.swift.org*.
9. "Bash man page" (https://linux.die.net/man/1/bash).
10. "Elvis operator" (https://wikidocs.adobe.com/wiki/display/coldfusionen/Elvis+operator). *wikidocs.adobe.com*.
11. "[RAILO-2195] add support for the Elvis Operator" (https://issues.jboss.org/browse/RAILO-2195). *JBoss Issue Tracker*.
12. "Expressions" (http://freemarker.org/docs/dgui_template_exp.html#dgui_template_exp_missing). *Apache FreeMarker Manual*. 2 June 2024.

13. "ECMAScript 2020 Language Specification" (http://www.ecma-international.org/ecma-262/11.0/index.html). Ecma International. June 2020.
14. "Null safety" (http://confluence.jetbrains.com/display/Kotlin/Null-safety)..
15. "PHP: rfc:isset_ternary" (https://wiki.php.net/rfc/isset_ternary). Retrieved 16 December 2014.
16. Kocak, Midori. "PHP RFC: Null Coalescing Assignment Operator" (https://wiki.php.net/rfc/null_coalesce_equal_operator). *PHP.net*. Retrieved 20 July 2017.
17. Hyde, Russ (25 April 2024). "What's new in R 4.4.0?" (https://www.jumpingrivers.com/blog/whats-new-r44/). *Jumping Rivers*.
18. "Database SQL Language Reference" (http://docs.oracle.com/cd/B28359_01/server.111/b28286/functions023.htm#SQLRF00617). *docs.oracle.com*.
19. "COALESCE (SQL Server Compact)" (https://technet.microsoft.com/en-us/library/ms174075.aspx). *technet.microsoft.com*. 24 March 2011.
20. "PostgreSQL: Documentation: 9.1: Conditional Expressions" (http://www.postgresql.org/docs/9.1/static/functions-conditional.html#FUNCTIONS-COALESCE-NVL-IFNULL). *www.postgresql.org*. 27 October 2016.
21. "SQLite Query Language: Core Functions" (http://www.sqlite.org/lang_corefunc.html). *www.sqlite.org*.
22. "MySQL :: MySQL 5.5 Reference Manual :: 12.3.2 Comparison Functions and Operators" (http://dev.mysql.com/doc/refman/5.5/en/comparison-operators.html#function_coalesce). *dev.mysql.com*.