# Introduction to Type Juggling

In PHP, [type juggling](https://www.php.net/manual/en/language.types.type-juggling.php) is an internal behavior that results in the conversion of variables to other data types in certain contexts, such as comparisons. While this is not inherently a security vulnerability, it can result in unexpected or undesired outcomes, resulting in security vulnerabilities depending on the concrete web application.

## PHP Loose vs. Strict Comparisons

Different from other programming languages, PHP supports two different types of comparisons: loose comparisons, which are done with two equal signs (==), and strict comparisons, which are done with three equal signs (===). A loose comparison compares two values after type juggling, while a strict comparison compares two values and their data type. As an example, consider the following code snippet:

Code: php

$a = 42;  
$b = "42";  
  
// loose comparison  
if ($a == $b) { echo "Loose Comparison";}  
  
// strict comparison  
if ($a === $b) { echo "Strict Comparison";}

We have two variables, an integer 42 and a string "42". The loose comparison results in type juggling, which converts the variable b to the number 42. Afterward, the values are compared such that the comparison evaluates to true and the string "Loose Comparison" is printed. On the other hand, the strict comparison also compares the data types. Since a is an integer and b is a string, the comparison is evaluated to false, and the string "Strict Comparison" is not printed.

The behavior of type juggling in a comparison context is documented [here](https://www.php.net/manual/en/language.operators.comparison.php#language.operators.comparison.types). Here are some important cases:

|  |  |  |
| --- | --- | --- |
| **Operand 1** | **Operand 2** | **Behavior** |
| string | string | Numerical or lexical comparison |
| null | string | Convert null to "" |
| null | anything but string | Convert both sides to bool |
| bool | anything | Convert both sides to bool |
| int | string | Convert string to int |
| float | string | Convert string to float |

For example, consider the comparison 1 == "1HelloWorld" which evaluates to true. Since the first operand is an int and the second operand is a string, PHP converts the string to an integer. When converting "1HelloWorld" to an integer, the result is 1. Thus, the comparison evaluates to true after type juggling.

A potentially even more odd example is the result of min(-1, null, 1), which is null. The function min computes the minimum of the provided arguments and returns it. To do so, the function compares the different arguments. When evaluating null < 1, both sides are converted to booleans. The integer 1 is converted to true while null is converted to false. It holds that false < true. Furthermore, the same methodology is applied when evaluating null < -1. The integer -1 is also converted to true. Thus, overall it holds that null < 1 and null < -1. Thus, null is the minimum of the provided arguments.

As a final example, let us consider the comparison "00" == "0e123". Intuitively, this comparison should evaluate to false since the arguments are both strings, and the strings are obviously different. This is a special case in which PHP executes a numerical comparison of the two strings, leading to a conversion to numbers. The e in the second argument is the scientific notation for floats, as we can see [here](https://www.php.net/manual/en/language.types.float.php). When both arguments are converted to numbers, the result is 0 for both sides. Thus, PHP evaluates the comparison as true.

Note: PHP only compares two strings numerically if both strings are of a valid number format.

Now let us have a look at the full behavior of a loose comparison which can be found [here](https://www.php.net/manual/en/types.comparisons.php):

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **true** | **false** | **1** | **0** | **-1** | **"1"** | **"0"** | **"-1"** | **null** | **[]** | **"php"** | **""** |
| true | ✓ | ✗ | ✓ | ✗ | ✓ | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✓ |
| false | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✓ | ✓ | ✗ | ✓ |
| 1 | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| 0 | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ (< PHP 8.0.0) | ✓ (< PHP 8.0.0) |
| -1 | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ |
| "1" | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| "0" | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ |
| "-1" | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ |
| null | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✗ | ✓ |
| [] | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✗ | ✗ |
| "php" | ✓ | ✗ | ✗ | ✓ (< PHP 8.0.0) | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ |
| "" | ✗ | ✓ | ✗ | ✓ (< PHP 8.0.0) | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✓ |

As we can see, the behavior of type juggling was changed in PHP 8.0.0. Notably, the comparison 0 == "php" evaluates to true in prior PHP versions, while this was changed to false in PHP 8.0.0.

On the other hand, the same table for a strict comparison looks like this:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **true** | **false** | **1** | **0** | **-1** | **"1"** | **"0"** | **"-1"** | **null** | **[]** | **"php"** | **""** |
| true | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| false | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| 1 | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| 0 | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| -1 | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| "1" | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| "0" | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ |
| "-1" | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ |
| null | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ |
| [] | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✗ |
| "php" | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ |
| "" | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |

We can see that there is no type juggling for strict comparisons, and the comparison only evaluates to true if both operands share the same data type and are equal.

## Other Programming Languages

While we focus on PHP here, the concept of type juggling also exists in other programming languages. For example, JavaScript implements loose and strict comparisons, similar to PHP. For more details, check out [this](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Equality_comparisons_and_sameness) page.

Just like in PHP, type juggling is executed during loose comparisons. However, this is not as lenient as it is in PHP. For instance, the comparison "0" == "0e1" evaluates to false in JavaScript since both arguments are treated as strings. However, the comparison 0 == "0e1" evaluates to true due to type juggling.