Search

Keyboard Shortcuts

?
: This help

j
: Next menu item

k
: Previous menu item

g p
: Previous man page

g n
: Next man page

G
: Scroll to bottom

g g
: Scroll to top

g h
: Goto homepage

g s
: Goto search
(current page)

/
: Focus search box

Переменные »
« Объявления типов

- Руководство по PHP
- Справочник языка
- Типы

Change language: Russian

Манипуляции с типами

PHP не требует явного определения типа при объявлении переменной. Если тип переменной не указан, он будет определён значением, которое она хранит. То есть, если переменной *$var* присваивается значение типа строка (string), то *$var* изменит тип на строку (string). Если после этого переменной *$var* будет присвоено значение типа целое число (int), то она изменит тип на целое число (int).

В ряде контекстов PHP может попытаться автоматически преобразовать тип значения в другой.
Существующие контексты:

- Числовой
- Строчный
- Логический
- Контекст целых чисел и строк
- Сравнительный
- Контекст функций

> **Замечание**: Когда значение нужно интерпретировать как другой тип, само значение *не* меняет тип.

О том, как принудительно установить тип переменной, рассказано в разделе «Приведение типа». О том, как изменить тип переменной, рассказано в описании функции settype().

Числовой контекст

Числовой контекст возникает при работе с арифметическими операторами.

В этом контексте, если один из операндов — число с плавающей точкой (float) (или не интерпретируется как целое число (int)), оба операнда интерпретируются как число с плавающей точкой (float) и результатом будет число с плавающей точкой (float). В противном случае операнды будут интерпретированы как целое число (int) и результатом также будет целое число (int). Начиная с PHP 8.0.0, если один из операндов не может быть интерпретирован, выбрасывается исключение TypeError.

Строчный контекст

Строчный контекст возникает при работе с языковыми конструкциями echo, print, при интерполяции строк или строковом операторе конкатенации.

В этом контексте значение будет интерпретироваться как строка (string). Если значение не может быть интерпретировано, выбрасывается исключение TypeError. До версии PHP 7.4.0 выдавалась ошибка уровня `E_RECOVERABLE_ERROR`.

Логический контекст

Логический контекст возникает при работе с условными операторами, тернарным оператором или логическим оператором.

В данном контексте значение будет интерпретироваться как логическое значение (bool).

Контекст целых чисел и строк

Контекст целых чисел и строк возникает при работе с побитовыми операторами.

В этом контексте, если у всех операндов тип строка (string), результат также будет строкой (string). В противном случае операнды будут интерпретированы как целое число (int) и результат также будет целым числом (int). Начиная с PHP 8.0.0, если один из операндов не может быть интерпретирован, то будет выброшено исключение TypeError.

Сравнительный контекст

Сравнительный контекст возникает при работе с операторами сравнения.

Преобразования типов, которые происходят в этом контексте, объяснены в таблице сравнения типов раздела «Операторы сравнения».

Контекст функций

Контекст функций возникает, когда значение передаётся типизированному параметру, свойству или возвращается из функции, в которой объявлен тип возвращаемого значения.

В этом контексте значение должно быть значением типа. Есть два исключения, первое: если тип значения — целое число (int), а объявленный тип — число с плавающей точкой (float), то целое число преобразовывается в число с плавающей точкой. Второй: если объявленный тип — это *скалярный* тип, значение преобразуемо в скалярный тип и режим строгой типизации активен (по умолчанию), значение может быть преобразовано в допустимое скалярное значение. Описание такого поведения дано ниже.

Внимание

Встроенные функции автоматически подставляют `null` к скалярным типам, это поведение *УСТАРЕЛО* с PHP 8.1.0.

Принудительная типизация с простым объявлением типов

- Объявление типа bool: значение интерпретируется как логическое значение (bool).
- Объявление типа int: значение интерпретируется как целое число (int), если преобразование надёжно определимо. Например, когда строка — числовая строка.
- Объявление типа float: значение интерпретируется как число с плавающей точкой (float), если преобразование надёжно определимо. Например, когда строка — числовая строка.
- Объявление типа string: значение интерпретируется как строка (string).

Принудительная типизация с объединением типов

Когда директива `strict_types` не включена, объявления скалярных типов подлежат ограниченному неявному приведению типов. Если точный тип значения — не часть объединения, то целевой тип выбран из списка в порядке убывания приоритета:

1. int
2. float
3. string
4. bool

Если тип и существует в объединении и значение может быть приведено к этому типу в соответствии с существующей семантикой проверки типов PHP, то PHP выбирает этот тип. В противном случае будет выбран следующий по порядку тип.

Предостережение

В качестве исключения, если значение — строка, а целое число (int) и число с плавающей точкой (float) — это часть объединения, предпочтительный тип будет определён существующей семантикой числовой строки. Например, для `«42»` PHP выбирает целое число (int), а для `«42.0»` — число с плавающей точкой (float).

> **Замечание**:
>
> Типы, которые не входят в список предпочтений, не станут целями для неявного приведения. Говоря конкретнее, никакого неявного приведения к типам null, false и true не будет.

Пример #1 Пример принудительного включения типов в состав объединения

```php
<?php

// int|string
42 --> 42 // точный тип
"42" --> "42" // точный тип
new ObjectWithToString --> "Результат __toString()"
// объект не совместим с int, переход к string
42.0 --> 42 // float совместимый с int
42.1 --> 42 // float совместимый с int
1e100 --> "1.0E+100" // float слишком велик для типа int, переход к string
INF --> "INF" // float слишком велик для типа int, переход к string
true --> 1 // bool совместимый с int
[] --> TypeError // массив не совместим с int или string
```

```
// int|float|bool
"45" --> 45 // int числовая строка
"45.0" --> 45.0 // float числовая строка

"45X" --> true // не числовая строка, переход к bool
"" --> false // не числовая строка, переход к bool
"X" --> true // не числовая строка, переход к bool
[] --> TypeError // массив не совместимый с int, float или bool
?>
```

Приведение типов

Приведение типа преобразовывает значение к выбранному типу, записывая тип в круглых скобках перед преобразуемым значением.

```
<?php

$foo = 10; // $foo — это целое число
$bar = (bool) $foo; // $bar — это логическое значение
?>
```

Разрешены следующие приведения типов:

- `(int)` — приведение типа к целому числу (int)
- `(bool)` — приведение типа к логическому значению (bool)
- `(float)` — приведение типа к числу с плавающей точкой (float)
- `(string)` — приведение типа к строке (string)
- `(array)` — приведение типа к массиву (array)
- `(object)` — приведение типа к объекту (object)
- `(unset)` — приведение типа к NULL

> **Замечание**:
>
> `(integer)` — псевдоним приведения типа `(int)`. `(boolean)` — псевдоним приведения типа `(bool)`. `(binary)` — псевдоним приведения типа `(string)`. `(double)` и `(real)` — псевдонимы приведения типа `(float)`. Эти приведения не используют каноническое имя типа и не рекомендуются.

Внимание

Псевдоним приведения типа `(real)` устарел с PHP 8.0.0.

Внимание

Приведение типа `(unset)` устарело с версии PHP 7.2.0. Обратите внимание, что приведение `(unset)` равносильно присвоению переменной или вызову значения NULL. Приведение `(unset)` удалено в PHP 8.0.0.

Предостережение

Приведение типа `(binary)` и префикс `b` существуют для прямой поддержки. Типы `(binary)` и `(string)` идентичны, однако, это может измениться, не нужно на это полагаться.

> **Замечание**:
>
> PHP игнорирует пробелы в круглых скобках при приведении типа. Поэтому следующие два приведения типов эквивалентны:
>
> ```
> <?php
>
> $foo = (int) $bar;
> $foo = ( int ) $bar;
> ?>
> ```

Приведение строк (string) и переменных к бинарным строкам (string):

```
<?php
```

```
$binary = (binary) $string;
$binary = b"binary string";
?>
```

Замечание: Вместо приведения переменной к типу строка (string) можно также заключить переменную в двойные кавычки.

```
<?php

$foo = 10; // $foo — целое число
$str = "$foo"; // $str — строка
$fst = (string) $foo; // $fst тоже строка

// Выводит, что «они одинаковые»
if ($fst === $str) {
echo "они одинаковые";
}
?>
```

Может быть неочевидно, что произойдёт при преобразовании между разными типами. Получить дополнительную информацию можно в разделах:

- Преобразование типа к логическому значению (boolean)
- Преобразование типа к целому числу (integer)
- Преобразование типа к числу с плавающей точкой (float)
- Преобразование типа к строке (string)
- Преобразование типа к массиву (array)
- Преобразование типа к объекту (object)
- Преобразование типа к ресурсу (resource)
- Преобразование типа к NULL
- Таблицы сравнения типов

Замечание: Поскольку PHP поддерживает индексацию в строках (string) через смещения, используя тот же синтаксис, что и индексация в массивах (array), следующий пример справедлив для всех версий PHP:

```
<?php

$a = 'car'; // $a — строка
$a[0] = 'b'; // $a по-прежнему строка
echo $a; // bar
?>
```

Дополнительную информацию можно найти разделе «Доступ к символу в строке и его изменение».

+ add a note

User Contributed Notes 7 notes

up
down
66
Raja ¶
19 years ago
Uneven division of an integer variable by another integer variable will result in a float by automatic conversion -- you do not have to cast the variables to floats in order to avoid integer truncation (as you would in C, for example):

```
$dividend = 2;
$divisor = 3;
$quotient = $dividend/$divisor;
print $quotient; // 0.66666666666667
```

up
down
27
fardelian ¶
10 years ago

Casting objects to arrays is a pain. Example:

```
<?php

class MyClass {

private $priv = 'priv_value';
protected $prot = 'prot_value';
public $pub = 'pub_value';
public $MyClasspriv = 'second_pub_value';

}

$test = new MyClass();
echo '<pre>';
print_r((array) $test);

/*
Array
(
[MyClasspriv] => priv_value
[*prot] => prot_value
[pub] => pub_value
[MyClasspriv] => second_pub_value
)
*/

?>
```

Yes, that looks like an array with two keys with the same name and it looks like the protected field was prepended with an asterisk. But that's not true:

```
<?php

foreach ((array) $test as $key => $value) {
$len = strlen($key);
echo "{$key} ({$len}) => {$value}<br />";
for ($i = 0; $i < $len; ++$i) {
echo ord($key[$i]) . ' ';
}
echo '<hr />';
}

/*
MyClasspriv (13) => priv_value
0 77 121 67 108 97 115 115 0 112 114 105 118
*prot (7) => prot_value
0 42 0 112 114 111 116
pub (3) => pub_value
112 117 98
MyClasspriv (11) => second_pub_value
77 121 67 108 97 115 115 112 114 105 118
*/

?>
```

The char codes show that the protected keys are prepended with '\0*\0' and private keys are prepended with '\0'.__CLASS__.'\0' so be careful when playing around with this.

11
Anonymous ¶

3 years ago

Cast operators have a very high precedence, for example (int)$a/$b is evaluated as ((int)$a)/$b, not as (int)($a/$b) [which would be like intdiv($a,$b) if both $a and $b are integers].

The only exceptions (as of PHP 8.0) are the exponentiation operator ** [i.e. (int)$a**$b is evaluated as (int)($a**$b) rather than ((int)$a)**$b] and the special access/invocation operators ->, ::, [] and () [i.e. in each of (int)$a->$b, (int)$a::$b, (int)$a[$b] and (int)$a($b), the cast is performed last on the result of the variable expression].

up

down

11

miracle at 1oo-percent dot de ¶

17 years ago

If you want to convert a string automatically to float or integer (e.g. "0.234" to float and "123" to int), simply add 0 to the string - PHP will do the rest.

e.g.

```
$val = 0 + "1.234";
(type of $val is float now)

$val = 0 + "123";
(type of $val is integer now)
```

up

down

11

rmirabelle ¶

13 years ago

The object casting methods presented here do not take into account the class hierarchy of the class you're trying to cast your object into.

```
/**
* Convert an object to a specific class.
* @param object $object
* @param string $class_name The class to cast the object to
* @return object
*/
public static function cast($object, $class_name) {
if($object === false) return false;
if(class_exists($class_name)) {
$ser_object = serialize($object);
$obj_name_len = strlen(get_class($object));
$start = $obj_name_len + strlen($obj_name_len) + 6;
$new_object = 'O:' . strlen($class_name) . ':"' . $class_name . '":';
$new_object .= substr($ser_object, $start);
$new_object = unserialize($new_object);
/**
* The new object is of the correct type but
* is not fully initialized throughout its graph.
* To get the full object graph (including parent
* class data, we need to create a new instance of
* the specified class and then assign the new
* properties to it.
*/
$graph = new $class_name;
foreach($new_object as $prop => $val) {
$graph->$prop = $val;
}
return $graph;
} else {
throw new CoreException(false, "could not find class $class_name for casting in DB::cast");
return false;
}
}
```

up
down
16
Anonymous ¶
21 years ago
```
Printing or echoing a FALSE boolean value or a NULL value results in an empty string:
(string)TRUE //returns "1"
(string)FALSE //returns ""
echo TRUE; //prints "1"
echo FALSE; //prints nothing!
```
up
down
12
ieee at REMOVE dot bk dot ru ¶
11 years ago
```
There are some shorter and faster (at least on my machine) ways to perform a type cast.
<?php
$string='12345.678';
$float=+$string;
$integer=0|$string;
$boolean=!!$string;
?>
```
＋add a note