



- [Downloads](#)
- [Documentation](#)
- [Get Involved](#)
- [Help](#)
- 

[Dutch PHP Conference 2024](#)

[Getting Started](#)

[Introduction](#)

[A simple tutorial](#)

[Language Reference](#)

[Basic syntax](#)

[Types](#)

[Variables](#)

[Constants](#)

[Expressions](#)

[Operators](#)

[Control Structures](#)

[Functions](#)

[Classes and Objects](#)

[Namespaces](#)

[Enumerations](#)

[Errors](#)

[Exceptions](#)

[Fibers](#)

[Generators](#)

[Attributes](#)

[References Explained](#)

[Predefined Variables](#)

[Predefined Exceptions](#)

[Predefined Interfaces and Classes](#)

[Predefined Attributes](#)

[Context options and parameters](#)

[Supported Protocols and Wrappers](#)

[Security](#)

[Introduction](#)

[General considerations](#)

[Installed as CGI binary](#)

[Installed as an Apache module](#)

[Session Security](#)

[Filesystem Security](#)

[Database Security](#)

[Error Reporting](#)

[User Submitted Data](#)

[Hiding PHP](#)

[Keeping Current](#)

[Features](#)

[HTTP authentication with PHP](#)

[Cookies](#)

[Sessions](#)

[Dealing with XForms](#)

[Handling file uploads](#)

[Using remote files](#)

[Connection handling](#)

[Persistent Database Connections](#)

[Command line usage](#)

[Garbage Collection](#)
[DTrace Dynamic Tracing](#)

[Function Reference](#)

[Affecting PHP's Behaviour](#)
[Audio Formats Manipulation](#)
[Authentication Services](#)
[Command Line Specific Extensions](#)
[Compression and Archive Extensions](#)
[Cryptography Extensions](#)
[Database Extensions](#)
[Date and Time Related Extensions](#)
[File System Related Extensions](#)
[Human Language and Character Encoding Support](#)
[Image Processing and Generation](#)
[Mail Related Extensions](#)
[Mathematical Extensions](#)
[Non-Text MIME Output](#)
[Process Control Extensions](#)
[Other Basic Extensions](#)
[Other Services](#)
[Search Engine Extensions](#)
[Server Specific Extensions](#)
[Session Extensions](#)
[Text Processing](#)
[Variable and Type Related Extensions](#)
[Web Services](#)
[Windows Only Extensions](#)
[XML Manipulation](#)
[GUI Extensions](#)

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

[Предопределённые переменные »](#)
[« Переменные](#)

- [Руководство по PHP](#)
- [Справочник языка](#)
- [Переменные](#)

Change language: Russian ▾

Основы

Переменные в PHP представлены знаком доллара с последующим именем переменной. Имя переменной чувствительно к регистру.

Имена переменных соответствуют тем же правилам, что и остальные наименования в PHP. Правильное имя переменной должно начинаться с буквы или символа подчёркивания и состоять из букв, цифр и символов подчёркивания в любом количестве. Это можно отобразить регулярным выражением: `^[a-zA-Z_\x80-\xff][a-zA-Z0-9_\x80-\xff]*$`

Замечание: Под буквами здесь подразумеваются символы a-z, A-Z и байты от 128 до 255 (0x80-0xff).

Замечание: `$this` — это специальная переменная, которой нельзя ничего присваивать. До PHP 7.1.0 было возможно косвенное присвоение (например, с использованием [переменных переменных](#)).

Подсказка

Смотрите также [Руководство по именованию](#).

Для информации о функциях работы с переменными обращайтесь к разделу [функций работы с переменными](#).

```
<?php
$var = 'Боб';
$Var = 'Джо';
echo "$var, $Var"; // выведет "Боб, Джо"

$4site = 'ещё нет'; // неверно; начинается с цифры
$_4site = 'ещё нет'; // верно; начинается с символа подчёркивания
$täyte = 'mansikka'; // верно; 'ä' это (Расширенный) ASCII 228.
?>
```

По умолчанию переменные всегда присваиваются по значению. То есть, когда вы присваиваете выражение переменной, все значение оригинального выражения копируется в эту переменную. Это означает, к примеру, что после того как одной переменной присвоено значение другой, изменение одной из них не влияет на другую. Дополнительную информацию об этом способе присвоения смотрите в разделе [Выражения](#).

PHP также предлагает иной способ присвоения значений переменным: [присвоение по ссылке](#). Это означает, что новая переменная просто ссылается (иначе говоря, "становится псевдонимом" или "указывает") на оригинальную переменную. Изменения в новой переменной отражаются на оригинале, и наоборот.

Для присвоения по ссылке, просто добавьте амперсанд (&) к началу имени присваиваемой (исходной) переменной. Например, следующий фрагмент кода дважды выводит 'Меня зовут Боб':

```
<?php
$foo = 'Боб'; // Присваивает $foo значение 'Боб'
$bar = &$foo; // Ссылка на $foo через $bar.
$bar = "Меня зовут $bar"; // Изменение $bar...
echo $bar;
echo $foo; // меняет и $foo.
?>
```

Важно отметить, что по ссылке могут быть присвоены только именованные переменные.

```
<?php
$foo = 25;
$bar = &$foo; // Это верное присвоение.
$bar = &(24 * 7); // Неверно; ссылка на неименованное выражение.

function test()
{
    return 25;
}
```

```
$bar = &test(); // Неверно.  
?>
```

Хорошей практикой считается инициализировать переменные, хотя в PHP это и не обязательное требование. Неинициализированные переменные принимают значение по умолчанию в зависимости от их типа, который определяется из контекста их первого использования: логические переменные принимают значение **false**, целые числа и числа с плавающей точкой — ноль, строки (например, при вызове с конструкцией [echo](#)) — пустую строку, а массивы становятся пустыми массивами.

Пример #1 Значения по умолчанию в неинициализированных переменных

```
<?php  
// Неустановленная И не имеющая ссылок (то есть без контекста использования) переменная; выведет NULL  
var_dump($unset_var);  
  
// Использование логической переменной; выведет 'false' (Подробнее по этому синтаксису смотрите раздел о тернарном операторе)  
echo $unset_bool ? "true\n" : "false\n";  
  
// Строковое использование; выведет 'string(3) "abc"'  
$unset_str .= 'abc';  
var_dump($unset_str);  
  
// Целочисленное использование; выведет 'int(25)'  
$unset_int += 25; // 0 + 25 => 25  
var_dump($unset_int);  
  
// Использование в качестве числа с плавающей точкой (float); выведет 'float(1.25)'  
$unset_float += 1.25;  
var_dump($unset_float);  
  
// Использование в качестве массива; выведет array(1) { [3]=> string(3) "def" }  
$unset_arr[3] = "def"; // array() + array(3 => "def") => array(3 => "def")  
var_dump($unset_arr);  
  
// Использование в качестве объекта; создаёт новый объект stdClass (смотрите  
http://www.php.net/manual/ru/reserved.classes.php)  
// Выведет: object(stdClass)#1 (1) { ["foo"]=> string(3) "bar" }  
$unset_obj->foo = 'bar';  
var_dump($unset_obj);  
?>
```

Полагаться на значения по умолчанию неинициализированных переменных довольно проблематично при включении файла в другой файл, использующий переменную с таким же именем. В случае работы с неинициализированной переменной вызывается ошибка уровня **E_WARNING** (до PHP 8.0.0 выбрасывалась ошибка уровня **E_NOTICE**), за исключением случая добавления элементов в неинициализированный массив. Для обнаружения инициализации переменной может быть использована языковая конструкция [isset\(\)](#).

[+add a note](#)

User Contributed Notes 2 notes

[up](#)

[down](#)

75

[jeff dot phpnet at tanasity dot com ¶](#)

13 years ago

This page should include a note on variable lifecycle:

Before a variable is used, it has no existence. It is unset. It is possible to check if a variable doesn't exist by using `isset()`. This returns true provided the variable exists and isn't set to null. With the exception of null, the value a variable holds plays no part in determining whether a variable is set.

Setting an existing variable to null is a way of unsetting a variable. Another way is variables may be destroyed by using the unset() construct.

```
<?php
print isset($a); // $a is not set. Prints false. (Or more accurately prints '')
$b = 0; // isset($b) returns true (or more accurately '1')
$c = array(); // isset($c) returns true
$b = null; // Now isset($b) returns false;
unset($c); // Now isset($c) returns false;
?>
```

is_null() is an equivalent test to checking that isset() is false.

The first time that a variable is used in a scope, it's automatically created. After this isset is true. At the point at which it is created it also receives a type according to the context.

```
<?php
$a_bool = true; // a boolean
$a_str = 'foo'; // a string
?>
```

If it is used without having been given a value then it is uninitialized and it receives the default value for the type. The default values are the _empty_ values. E.g Booleans default to FALSE, integers and floats default to zero, strings to the empty string '', arrays to the empty array.

A variable can be tested for emptiness using empty();

```
<?php
$a = 0; //This is set, but is empty
?>
```

Unset variables are also empty.

```
<?php
empty($vessel); // returns true. Also $vessel is unset.
?>
```

Everything above applies to array elements too.

```
<?php
$item = array();
//Now isset($item) returns true. But isset($item['unicorn']) is false.
//empty($item) is true, and so is empty($item['unicorn'])

$item['unicorn'] = '';
//Now isset($item['unicorn']) is true. And empty($item) is false.
//But empty($item['unicorn']) is still true;

$item['unicorn'] = 'Pink unicorn';
//isset($item['unicorn']) is still true. And empty($item) is still false.
//But now empty($item['unicorn']) is false;
?>
```

For arrays, this is important because accessing a non-existent array item can trigger errors; you may want to test arrays and array items for existence with isset before using them.

[up](#)

[down](#)

18

[anisgazig at gmail dot com ¶](#)

3 years ago

clear concept of variable declaration rules and classification

variable declaration rules:

- 1.start with dollar sign(\$)
- 2.first letter of variable name comes from a-zA-z_
- 3.next letters of variable name comes from a-zA-Z0-9_
- 4.no space,no syntex

classification of variables:

Variable are mainly Two types

- 1.Predefined Variable
- 2.User Define Variable

Predefined Variable

There are 12 predefined variables in php 8

- 1.\$GLOBALS
- 2.\$_SERVER
- 3.\$_REQUEST
- 4.\$_FILES
- 5.\$_ENV
- 6.\$_SESSION
- 7.\$_COOKIE
- 8.\$_GET
- 9.\$_POST
- 10.\$http_response_header
- 11.\$argc
- 12.\$argv

User Define Variable

User Define variable are 3 types

- 1.variable scope
- 2.variable variables
- 3.reference variable

Variable Scope

variable scope are 3 types

- 1.local scope
- 2.global scope
- 3.static variable

[+add a note](#)

- [Переменные](#)
 - [ОСНОВЫ](#)
 - [Предопределённые переменные](#)
 - [Область видимости переменной](#)
 - [Переменные переменных](#)
 - [Переменные извне PHP](#)
- [Copyright © 2001-2024 The PHP Group](#)
- [My PHP.net](#)
- [Contact](#)
- [Other PHP.net sites](#)
- [Privacy policy](#)

