



- [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

[Аргументы функции »](#)
[« Функции](#)

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

Change language: Russian ▾

Функции, определяемые пользователем

Приведём пример синтаксиса, используемого для описания функций:

Пример #1 Псевдокод для демонстрации использования функций

```
<?php
function foo($arg_1, $arg_2, /* ..., */ $arg_n)
{
    echo "Пример функции.\n";
    return $retval;
}
?>
```

Внутри функции можно использовать любой корректный PHP-код, в том числе другие функции и даже объявления [классов](#).

Имена функций следуют тем же правилам, что и другие метки в PHP. Корректное имя функции начинается с буквы или знака подчёркивания, за которым следует любое количество букв, цифр или знаков подчёркивания. В качестве регулярного выражения оно может быть выражено так: `^[a-zA-Z_\x80-\xff][a-zA-Z0-9_\x80-\xff]*$`.

Подсказка

Смотрите также [Руководство по именованию](#).

Функции не обязаны быть определены до их использования, *исключая* тот случай, когда функции определяются условно, как это показано в двух последующих примерах.

В случае, когда функция определяется в зависимости от какого-либо условия, например, как это показано в двух приведённых ниже примерах, обработка описания функции должна *предшествовать* её вызову.

Пример #2 Функции, зависящие от условий

```
<?php

$makefoo = true;

/* Мы не можем вызвать функцию foo() в этом месте,
поскольку она ещё не определена, но мы можем
обратиться к bar() */

bar();

if ($makefoo) {
    function foo()
    {
        echo "Я не существую до тех пор, пока выполнение программы меня не достигнет.\n";
    }
}

/* Теперь мы благополучно можем вызывать foo(),
поскольку $makefoo была интерпретирована как true */

if ($makefoo) foo();

function bar()
{
    echo "Я существую сразу с начала старта программы.\n";
}

?>
```

Пример #3 Вложенные функции

```

<?php
function foo()
{
function bar()
{
echo "Я не существую пока не будет вызвана foo().\n";
}
}

/* Мы пока не можем обратиться к bar(),
поскольку она ещё не определена. */

foo();

/* Теперь мы можем вызвать функцию bar(),
обработка foo() сделала её доступной. */

bar();

?>

```

Все функции и классы PHP имеют глобальную область видимости - они могут быть вызваны вне функции, даже если были определены внутри и наоборот.

PHP не поддерживает перегрузку функции, также отсутствует возможность переопределить или удалить объявленную ранее функцию.

Замечание: Имена функций регистронезависимы для символов ASCII от A до Z, тем не менее, предпочтительнее вызывать функции так, как они были объявлены.

Функции PHP поддерживают как [списки аргументов переменной длины](#), так и [значения аргументов по умолчанию](#). Смотрите также описания функций [func_num_args\(\)](#), [func_get_arg\(\)](#) и [func_get_args\(\)](#) для более детальной информации.

Можно вызывать функции PHP рекурсивно.

Пример #4 Рекурсивные функции

```

<?php
function recursion($a)
{
if ($a < 20) {
echo "$a\n";
recursion($a + 1);
}
}

?>

```

Замечание: Рекурсивный вызов методов/процедур с глубиной более 100-200 уровней рекурсии может вызвать переполнение стека и привести к аварийному завершению скрипта. В частности, бесконечная рекурсия будет считаться программной ошибкой.

[+add a note](#)

User Contributed Notes

There are no user contributed notes for this page.

- [Функции](#)
 - [Функции, определяемые пользователем](#)
 - [Аргументы функции](#)
 - [Возврат значений](#)
 - [Обращение к функциям через переменные](#)
 - [Встроенные функции](#)
 - [Анонимные функции](#)
 - [Стрелочные функции](#)

- [Синтаксис callable-объектов первого класса](#)

- [Copyright © 2001-2024 The PHP Group](#)
- [My PHP.net](#)
- [Contact](#)
- [Other PHP.net sites](#)
- [Privacy policy](#)

