



- [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

[for »](#)
[« while](#)

- [Руководство по PHP](#)
- [Справочник языка](#)
- [Управляющие конструкции](#)

Change language: Russian ▾

do-while

(PHP 4, PHP 5, PHP 7, PHP 8)

Цикл `do-while` очень похож на цикл `while`, с тем отличием, что истинность выражения проверяется в конце итерации, а не в начале. Главное отличие от обычного цикла `while` в том, что первая итерация цикла `do-while` гарантированно выполнится (истинность выражения проверяется в конце итерации), тогда как она может не выполниться в обычном цикле `while` (истинность выражения которого проверяется в начале выполнения каждой итерации, и если изначально имеет значение **false**, то выполнение цикла будет прервано сразу).

Есть только один вариант синтаксиса цикла `do-while`:

```
<?php
$i = 0;
do {
    echo $i;
} while ($i > 0);
?>
```

В примере цикл будет выполнен ровно один раз, так как после первой итерации, когда проверяется истинность выражения, она будет вычислена как **false** (`$i` не больше 0) и выполнение цикла прекратится.

Опытные пользователи С могут быть знакомы с другим использованием цикла `do-while`, которое позволяет остановить выполнение хода программы в середине блока, для этого нужно обернуть нужный блок кода вызовом `do-while (0)` и использовать [break](#). Следующий фрагмент кода демонстрирует этот подход:

```
<?php
do {
    if ($i < 5) {
        echo "i ещё недостаточно велико";
        break;
    }
    $i *= $factor;
    if ($i < $minimum_limit) {
        break;
    }
    echo "значение i уже подходит";

    /* обработка i */

} while (0);
?>
```

Можно использовать оператор [goto](#) вместо подобного "хака".

[+add a note](#)

User Contributed Notes 1 note

[up](#)

[down](#)

30

[jayreardon at gmail dot com](#) ¶

16 years ago

There is one major difference you should be aware of when using the `do--while` loop vs. using a simple `while` loop: And that is when the check condition is made.

In a `do--while` loop, the test condition evaluation is at the end of the loop. This means that the code inside of the loop will iterate once through before the condition is ever evaluated. This is ideal for tasks that need to execute once before a test is made to continue, such as test that is dependant upon the results of the loop.

Conversely, a plain `while` loop evaluates the test condition at the begining of the loop before any execution in the loop

block is ever made. If for some reason your test condition evaluates to false at the very start of the loop, none of the code inside your loop will be executed.

[+add a note](#)

- [Управляющие конструкции](#)
 - [Введение](#)
 - [if](#)
 - [else](#)
 - [elseif/else if](#)
 - [Альтернативный синтаксис управляющих структур](#)
 - [while](#)
 - [do-while](#)
 - [for](#)
 - [foreach](#)
 - [break](#)
 - [continue](#)
 - [switch](#)
 - [match](#)
 - [declare](#)
 - [return](#)
 - [require](#)
 - [include](#)
 - [require_once](#)
 - [include_once](#)
 - [goto](#)
- [Copyright © 2001-2024 The PHP Group](#)
- [My PHP.net](#)
- [Contact](#)
- [Other PHP.net sites](#)
- [Privacy policy](#)

