



- [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

[Правила разрешения имён »](#)
[« Глобальное пространство](#)

- [Руководство по PHP](#)
- [Справочник языка](#)
- [Пространства имён](#)

Change language: Russian

Пространства имён: возврат к глобальному пространству для функций и констант

(PHP 5 >= 5.3.0, PHP 7, PHP 8)

Когда внутри пространства имён PHP встречается неполное имя класса, функции или контекст константы, он разрешает эти имена с разными приоритетами. Имена классов разрешаются в текущее имя пространства имён. Поэтому, чтобы получить доступ ко внутреннему классу или пользовательскому классу вне пространства имён, необходимо обращаться к ним по абсолютному имени. Например:

Пример #1 Доступ к глобальным классам внутри пространства имён

```
<?php

namespace A\B\C;

class Exception extends \Exception {}

$a = new Exception('hi'); // Переменная $a — это объект класса A\B\C\Exception
$b = new \Exception('hi'); // Переменная $b — это объект класса Exception

$c = new ArrayObject; // Фатальная ошибка, класс A\B\C\ArrayObject не найден

?>
```

PHP будет обращаться к глобальным функциям или константам, если функция или константа не существует в текущем пространстве имён.

Пример #2 Возврат к глобальным функциям или константам внутри пространства имён

```
<?php

namespace A\B\C;

const E_ERROR = 45;
function strlen($str)
{
    return \strlen($str) - 1;
}

echo E_ERROR, "\n"; // Выводит 45
echo INI_ALL, "\n"; // Выводит «7» — возвращается к глобальной константе INI_ALL

echo strlen('hi'), "\n"; // Выводит «1»
if (is_array('hi')) { // Выводит строку «это не массив»
    echo "Это массив\n";
} else {
    echo "Это не массив\n";
}

?>
```

[+add a note](#)

User Contributed Notes 1 note

[up](#)
[down](#)

33

[markus at malkusch dot de](#) 

9 years ago

You can use the fallback policy to provide mocks for built-in functions like `time()`. You therefore have to call those

functions unqualified:

```
<?php
namespace foo;

function time() {
    return 1234;
}

assert (1234 == time());
?>
```

However there's a restriction that you have to define the mock function before the first usage in the tested class method. This is documented in Bug #68541.

You can find the mock library php-mock at GitHub.

[+add a note](#)

- [Пространства имён](#)
 - [Обзор](#)
 - [Пространства имён](#)
 - [Подпространства имён](#)
 - [Несколько пространств имён в одном файле](#)
 - [Основы](#)
 - [Пространства имён и динамические особенности языка](#)
 - [Ключевое слово namespace и константа `__NAMESPACE__`](#)
 - [Псевдонимирование и импорт](#)
 - [Глобальное пространство](#)
 - [Возврат к глобальному пространству](#)
 - [Правила разрешения имён](#)
 - [FAQ](#)
- [Copyright © 2001-2024 The PHP Group](#)
- [My PHP.net](#)
- [Contact](#)
- [Other PHP.net sites](#)
- [Privacy policy](#)

