



- [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

[Пространства имён и динамические особенности языка »](#)
[« Несколько пространств имён в одном файле](#)

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

Change language: Russian ▾

Пространства имён: основы

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

Прежде чем обсуждать работу с пространствами имён, важно понять, как PHP узнаёт, какие элементы из пространства имён запрашиваются в коде. Можно провести аналогию между пространствами имён PHP и файловой системой. Есть три способа обратиться к файлу в файловой системе:

1. Относительное имя файла наподобие `foo.txt` разрешится в `currentdirectory/foo.txt`, где `currentdirectory` — текущая директория, в которой мы находимся. Тогда, если текущая директория — `/home/foo`, то имя преобразуется в `/home/foo/foo.txt`.
2. Относительное имя пути наподобие `subdirectory/foo.txt` разрешится в `currentdirectory/subdirectory/foo.txt`.
3. Абсолютное имя пути наподобие `/main/foo.txt` останется таким же: `/main/foo.txt`.

Тот же принцип сохранится при разрешении элементов из пространств имён PHP. Например, имя класса разрешено указывать тремя способами:

1. Неполное имя, или имя класса без префикса, наподобие `$a = new foo();` или `foo::staticmethod();`. Если текущее пространство имён — `currentnamespace`, то это имя разрешится в `currentnamespace\foo`. В коде в глобальном пространстве имён имя останется таким же: `foo`. Предостережение: неполные имена функций и констант будут разрешаться в глобальные функции и константы, если они не определены в текущем пространстве имён. Подробнее об этом рассказано в разделе «[Пространства имён: возврат к глобальному пространству для функций и констант](#)».
2. Полное имя, или имя класса с префиксами наподобие `$a = new subnamespace\foo();` или `subnamespace\foo::staticmethod();`. Если текущее пространство имён — `currentnamespace`, то это имя разрешится в `currentnamespace\subnamespace\foo`. В коде в глобальном пространстве имён имя разрешится в `subnamespace\foo`.
3. Абсолютное имя, или имя с префиксом в начале, который указывает на глобальное пространство имён наподобие `$a = new \currentnamespace\foo();` или `\currentnamespace\foo::staticmethod();`. Такое имя разрешается в буквальное имя, заданное в коде: `currentnamespace\foo`.

Вот пример трёх видов синтаксиса в реальном коде:

file1.php

```
<?php
```

```
namespace Foo\Bar\subnamespace;
```

```
const FOO = 1;
function foo() {}
class foo
{
    static function staticmethod() {}
}
?>
```

file2.php

```
<?php
```

```
namespace Foo\Bar;
include 'file1.php';
```

```
const FOO = 2;
function foo() {}
class foo
{
    static function staticmethod() {}
}
```

```
/* Неполные имена */
```

```
foo(); // Разрешается в функцию Foo\Bar\foo
```

```
foo::staticmethod(); // Разрешается в метод staticmethod класса Foo\Bar\foo
echo F00; // Разрешается в константу Foo\Bar\F00

/* Полные имена */
subnamespace\foo(); // Разрешается в функцию Foo\Bar\subnamespace\foo
subnamespace\foo::staticmethod(); // Разрешается в метод staticmethod класса Foo\Bar\subnamespace\foo
echo subnamespace\F00; // Разрешается в константу Foo\Bar\subnamespace\F00

/* Абсолютные имена */
\Foo\Bar\foo(); // Разрешается в функцию Foo\Bar\foo
\Foo\Bar\foo::staticmethod(); // Разрешается в метод staticmethod класса Foo\Bar\foo
echo \Foo\Bar\F00; // Разрешается в константу Foo\Bar\F00

?>
```

Обратите внимание, что для доступа к глобальным классам, функциям или константам разрешается указывать абсолютное имя, например, `\strlen()`, `\Exception` или `\INI_ALL`.

Пример #1 Доступ к глобальным классам, функциям и константам из пространства имён

```
<?php

namespace Foo;

function strlen() {}
const INI_ALL = 3;
class Exception {}

$a = \strlen('hi'); // Вызывает глобальную функцию strlen
$b = \INI_ALL; // Получает доступ к глобальной константе INI_ALL
$c = new \Exception('error'); // Создаёт экземпляр глобального класса Exception

?>
```

[+add a note](#)

User Contributed Notes 5 notes

[up](#)

[down](#)

199

[richard at richard-sumilang dot com ¶](#)

15 years ago

Syntax for extending classes in namespaces is still the same.

Lets call this Object.php:

```
<?php

namespace com\rsumilang\common;

class Object{
// ... code ...
}

?>
```

And now lets create a class called String that extends object in String.php:

```
<?php

class String extends com\rsumilang\common\Object{
// ... code ...
}
```

```
}
```

```
?>
```

Now if your class `String` was defined in the same namespace as `Object` then you don't have to specify a full namespace path:

```
<?php
```

```
namespace com\rsumilang\common;
```

```
class String extends Object
```

```
{  
    // ... code ...  
}
```

```
?>
```

Lastly, you can also alias a namespace name to use a shorter name for the class you are extending in case your class is in a separate namespace:

```
<?php
```

```
namespace com\rsumilang\util;  
use com\rsumilang\common as Common;
```

```
class String extends Common\Object
```

```
{  
    // ... code ...  
}
```

```
?>
```

- Richard Sumilang

[up](#)

[down](#)

106

[Anonymous ¶](#)

9 years ago

```
<?php
```

```
namespace Foo;
```

```
try {  
    // Something awful here  
    // That will throw a new exception from SPL  
}  
catch (Exception as $ex) {  
    // We will never get here  
    // This is because we are catching Foo\Exception  
}  
?>
```

Instead use fully qualified name for the exception to catch it

```
<?php
```

```
namespace Foo;
```

```
try {  
    // something awful here  
    // That will throw a new exception from SPL  
}
```

```
catch (\Exception as $ex) {
// Now we can get here at last
}
```

[up](#)

[down](#)

48

[Lukas Z ¶](#)

12 years ago

Well variables inside namespaces do not override others since variables are never affected by namespace but always global:

"Although any valid PHP code can be contained within a namespace, only four types of code are affected by namespaces:

classes, interfaces, functions and constants. "

Source: "Defining Namespaces"

<http://www.php.net/manual/en/language.namespaces.definition.php>

[up](#)

[down](#)

39

[tom at tomwardrop dot com ¶](#)

11 years ago

It seems the file system analogy only goes so far. One thing that's missing that would be very useful is relative navigation up the namespace chain, e.g.

```
<?php
namespace MyProject {
class Person {}
}

namespace MyProject\People {
class Adult extends ..\Person {}
}
?>
```

That would be really nice, especially if you had really deep namespaces. It would save you having to type out the full namespace just to reference a resource one level up.

[up](#)

[down](#)

16

[philip dot preisser at arcor dot de ¶](#)

12 years ago

Working with variables can overwrite equal variables in other namespaces

```
<?php // php5 - package-version : 5.3.5-1ubuntu7.2
```

```
namespace
main
{}
```

```
namespace
main\sub1
{
$data = 1;
}
```

```
namespace
main\sub2
{
echo $data;// 1
$data = 2;
}
```

```
namespace
```

```
main\sub1
{
echo $data;// 2
$data = 1;
}

namespace
{
echo $data;// 1
}
```

?>

[+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](#)

