



- [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 |

[Неявное использование механизма ссылок »](#)
[« Возврат по ссылке](#)

- [Руководство по PHP](#)
- [Справочник языка](#)
- [Объяснение ссылок](#)

Change language: Russian

Сброс переменных-ссылок

При сбросе ссылки, просто разрывается связь имени и содержимого переменной. Это не означает, что содержимое переменной будет уничтожено. Например:

```
<?php
$a = 1;
$b =& $a;
unset($a);
?>
```

Этот код не сбросит *\$b*, а только *\$a*.

Опять же, можно провести аналогию с вызовом **unlink** (в Unix).

[+add a note](#)

User Contributed Notes 7 notes

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

413
[ojars26 at NOSPAM dot inbox dot lv ¶](#)
15 years ago

Simple look how PHP Reference works

```
<?php
/* Imagine this is memory map
```

| pointer | value | variable |
|---------|-------|----------|
| ----- | | |
| 1 | NULL | --- |
| 2 | NULL | --- |
| 3 | NULL | --- |
| 4 | NULL | --- |
| 5 | NULL | --- |

```
Create some variables */
$a=10;
$b=20;
$c=array ('one'=>array (1, 2, 3));
/* Look at memory
```

| pointer | value | variable's |
|---------|-------|---------------|
| ----- | | |
| 1 | 10 | \$a |
| 2 | 20 | \$b |
| 3 | 1 | \$c['one'][0] |
| 4 | 2 | \$c['one'][1] |
| 5 | 3 | \$c['one'][2] |

```
do */
$a=&$c['one'][2];
/* Look at memory
```

| pointer | value | variable's |
|---------|-------|---|
| ----- | | |
| 1 | NULL | --- //value of \$a is destroyed and pointer is free |
| 2 | 20 | \$b |
| 3 | 1 | \$c['one'][0] |
| 4 | 2 | \$c['one'][1] |
| 5 | 3 | \$c['one'][2] , \$a // \$a is now here |

```
-----
do */
$b=&$a; // or $b=&$c['one'][2]; result is same as both "$c['one'][2]" and "$a" is at same pointer.
/* Look at memory
```

| pointer | value | variable's |
|---------|-------|---------------------------|
| 1 | NULL | --- |
| 2 | NULL | --- |
| 3 | 1 | \$c['one'][0] |
| 4 | 2 | \$c['one'][1] |
| 5 | 3 | \$c['one'][2] , \$a , \$b |

```
next do */
unset($c['one'][2]);
/* Look at memory
```

| pointer | value | variable's |
|---------|-------|---------------|
| 1 | NULL | --- |
| 2 | NULL | --- |
| 3 | 1 | \$c['one'][0] |
| 4 | 2 | \$c['one'][1] |
| 5 | 3 | \$a , \$b |

```
next do */
$c['one'][2]=500; //now it is in array
/* Look at memory
```

| pointer | value | variable's |
|---------|-------|---------------|
| 1 | 500 | \$c['one'][2] |
| 2 | NULL | --- |
| 3 | 1 | \$c['one'][0] |
| 4 | 2 | \$c['one'][1] |
| 5 | 3 | \$a , \$b |

```
lets tray to return $c['one'][2] at old pointer an remove reference $a,$b. */
$c['one'][2]=&$a;
unset($a);
unset($b);
/* look at memory
```

| pointer | value | variable's |
|---------|-------|---------------|
| 1 | NULL | --- |
| 2 | NULL | --- |
| 3 | 1 | \$c['one'][0] |
| 4 | 2 | \$c['one'][1] |
| 5 | 3 | \$c['one'][2] |

I hope this helps.

[up](#)
[down](#)
40
[sony-santos at bol dot com dot br ¶](#)
17 years ago

```
<?php
//if you do:

$a = "hihaha";
$b = &$a;
```

```
$c = "eita";  
$b = $c;  
echo $a; // shows "eita"
```

```
$a = "hihaha";  
$b = &$a;  
$c = "eita";  
$b = &$c;  
echo $a; // shows "hihaha"
```

```
$a = "hihaha";  
$b = &$a;  
$b = null;  
echo $a; // shows nothing (both are set to null)
```

```
$a = "hihaha";  
$b = &$a;  
unset($b);  
echo $a; // shows "hihaha"
```

```
$a = "hihaha";  
$b = &$a;  
$c = "eita";  
$a = $c;  
echo $b; // shows "eita"
```

```
$a = "hihaha";  
$b = &$a;  
$c = "eita";  
$a = &$c;  
echo $b; // shows "hihaha"
```

```
$a = "hihaha";  
$b = &$a;  
$a = null;  
echo $b; // shows nothing (both are set to null)
```

```
$a = "hihaha";  
$b = &$a;  
unset($a);  
echo $b; // shows "hihaha"  
?>
```

I tested each case individually on PHP 4.3.10.

[up](#)

[down](#)

5

[lazer erazer](#) 

17 years ago

Your idea about unsetting all referenced variables at once is right,
just a tiny note that you changed NULL with unset()...
again, unset affects only one name and NULL affects the data,
which is kept by all the three names...

```
<?php  
$a = 1;  
$b =& $a;  
$b = NULL;  
?>
```

This does also work!

```
<?php
$a = 1;
$b =& $a;
$c =& $b;
$b = NULL;
?>
```

[up](#)

[down](#)

3

[donny at semeleer dot nl ¶](#)

17 years ago

Here's an example of unsetting a reference without losing an ealier set reference

```
<?php
$foo = 'Bob'; // Assign the value 'Bob' to $foo
$bar = &$foo; // Reference $foo via $bar.
$bar = "My name is $bar"; // Alter $bar...
echo $bar;
echo $foo; // $foo is altered too.
$foo = "I am Frank"; // Alter $foo and $bar because of the reference
echo $bar; // output: I am Frank
echo $foo; // output: I am Frank

$foobar = &$bar; // create a new reference between $foobar and $bar
$foobar = "hello $foobar"; // alter $foobar and with that $bar and $foo
echo $foobar; //output : hello I am Frank
unset($bar); // unset $bar and destroy the reference
$bar = "dude!"; // assign $bar
/* even though the reference between $bar and $foo is destroyed, and also the
reference between $bar and $foobar is destroyed, there is still a reference
between $foo and $foobar. */
echo $foo; // output : hello I am Frank
echo $bar; // output : due!
?>
```

[up](#)

[down](#)

0

[smcbride at msn dot com ¶](#)

2 years ago

A little quirk on unset() when using references that may help someone.

If you want to delete the element of a reference to an array, you need to have the reference point to the parent of the key that you want to delete.

```
<?php
$arr = array('foo' => array('foo_sub1' => 'hey', 'foo_sub2' => 'you'), 'bar' => array('bar_sub1' => 'good', 'bar_sub2' => 'bye'));

$parref = &$arr['foo'];
$childref = &$parref['foo_sub1'];

unset($childref); // this will simply unset the reference to child
unset($parref['foo_sub1']); // this will actually unset the data in $arr;
$parref['foo_sub1'] = NULL; // this will set the element to NULL, but not delete it. If you run it after unset(), it add the key back and set it to NULL

?>
```

This is nice to use for passing something dynamically to a function by reference without copying the entire array to the function, but you want to do some maintenance on the array.

[up](#)

[down](#)

1
[frowa at foxmail dot com ¶](#)

2 years ago

it's my way to remember.

<?php

```
// the var $a is point to the value 1, as a line connect to value 1
```

```
$a = 1;
```

```
// the var $b point to the value which the var $a point to, as a new line connect to value 1
```

```
$b =& $a;
```

```
// cut the line of the var $a to value 1,now $a is freedom,it's nothing point to. so the value of $a is null
```

```
unset($a);
```

```
?>
```

```
$a-----> 1
```

```
↑
```

```
|
```

```
|
```

```
$b
```

[up](#)

[down](#)

-4

[libi ¶](#)

18 years ago

clerca at inp-net dot eu dot org

"

If you have a lot of references linked to the same contents, maybe it could be useful to do this :

<?php

```
$a = 1;
```

```
$b = & $a;
```

```
$c = & $b; // $a, $b, $c reference the same content '1'
```

```
$b = NULL; // All variables $a, $b or $c are unset
```

```
?>
```

"

NULL will not result in unsetting the variables.

Its only change the value to "null" for all the variables.

because they all points to the same "part" in the memory.

[+add a note](#)

- [Объяснение ссылок](#)
 - [Что такое ссылки](#)
 - [Что делают ссылки](#)
 - [Чем ссылки не являются](#)
 - [Передача по ссылке](#)
 - [Возврат по ссылке](#)
 - [Сброс переменных-ссылок](#)
 - [Неявное использование механизма ссылок](#)

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

