Search

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

var_export »
« unset

var_dump

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

var_dump — Выводит информацию о переменной

Описание

var_dump(mixed `$value`, mixed `...$values`): void

Функция отображает структурированную информацию об одном или нескольких выражениях, включая их тип и значение. Массивы и объекты анализируются рекурсивно, значениям задаются отступы, чтобы показать структуру.

Все общедоступные, закрытые и защищённые свойства объекта будут возвращены в выводе, кроме объектов, в которых реализован метод __debugInfo().

Подсказка

Как и всё, что выводит результат в браузер, функции контроля вывода можно вызывать, чтобы перехватить выводимые этой функцией данные и сохранять их, например в строку (string).

Список параметров

`value`

Выражение, которое нужно вывести.

`values`

Следующие выражения для вывода.

Возвращаемые значения

Функция не возвращает значения после выполнения.

Примеры

Пример #1 Пример использования функции var_dump()

```php
<?php

$a = array(1, 2, array("a", "b", "c"));
var_dump($a);
?>
```

Результат выполнения приведённого примера:

```
array(3) {
  [0]=>
  int(1)
  [1]=>
  int(2)
  [2]=>
  array(3) {
    [0]=>
    string(1) "a"
    [1]=>
    string(1) "b"
    [2]=>
    string(1) "c"
  }
}
```

```php
<?php
```

```
$b = 3.1;
$c = true;
var_dump($b, $c);

?>
```

Результат выполнения приведённого примера:

```
float(3.1)
bool(true)
```

Смотрите также

- print_r() - Выводит удобочитаемую информацию о переменной
- debug_zval_dump() - Сбрасывает строковое представление внутренней структуры zval на вывод
- var_export() - Выводит или возвращает интерпретируемое строковое представление переменной
- __debugInfo()

User Contributed Notes 16 notes

149
rich dot schu at gmail dot com ¶
15 years ago
Keep in mind if you have xdebug installed it will limit the var_dump() output of array elements and object properties to 3 levels deep.

To change the default, edit your xdebug.ini file and add the folllowing line:
xdebug.var_display_max_depth=n

More information here:
http://www.xdebug.org/docs/display

110
edwardzyang at thewritingpot dot com ¶
18 years ago
If you're like me and uses var_dump whenever you're debugging, you might find these two "wrapper" functions helpful.

This one automatically adds the PRE tags around the var_dump output so you get nice formatted arrays.

```
<?php

function var_dump_pre($mixed = null) {
echo '<pre>';
var_dump($mixed);
echo '</pre>';
return null;
}

?>
```

This one returns the value of var_dump instead of outputting it.

```
<?php

function var_dump_ret($mixed = null) {
ob_start();
var_dump($mixed);
$content = ob_get_contents();
```

```
ob_end_clean();
return $content;
}

?>
```

Fairly simple functions, but they're infinitely helpful (I use var_dump_pre() almost exclusively now).

up
down
35

b dot bergloev at gmail dot com ¶

9 years ago

I post a new var_dump function with colors and collapse features. It can also adapt to terminal output if you execute it from there. No need to wrap it in a pre tag to get it to work in browsers.

```
<?php
function dump_debug($input, $collapse=false) {
$recursive = function($data, $level=0) use (&$recursive, $collapse) {
global $argv;

$isTerminal = isset($argv);

if (!$isTerminal && $level == 0 && !defined("DUMP_DEBUG_SCRIPT")) {
define("DUMP_DEBUG_SCRIPT", true);

echo '<script language="Javascript">function toggleDisplay(id) {';
echo 'var state = document.getElementById("container"+id).style.display;';
echo 'document.getElementById("container"+id).style.display = state == "inline" ? "none" : "inline";';
echo 'document.getElementById("plus"+id).style.display = state == "inline" ? "inline" : "none";';
echo '}</script>'."\n";
}

$type = !is_string($data) && is_callable($data) ? "Callable" : ucfirst(gettype($data));
$type_data = null;
$type_color = null;
$type_length = null;

switch ($type) {
case "String":
$type_color = "green";
$type_length = strlen($data);
$type_data = "\"" . htmlentities($data) . "\""; break;

case "Double":
case "Float":
$type = "Float";
$type_color = "#0099c5";
$type_length = strlen($data);
$type_data = htmlentities($data); break;

case "Integer":
$type_color = "red";
$type_length = strlen($data);
$type_data = htmlentities($data); break;

case "Boolean":
$type_color = "#92008d";
$type_length = strlen($data);
$type_data = $data ? "TRUE" : "FALSE"; break;

case "NULL":
$type_length = 0; break;
```

```
case "Array":
$type_length = count($data);
}

if (in_array($type, array("Object", "Array"))) {
$notEmpty = false;

foreach($data as $key => $value) {
if (!$notEmpty) {
$notEmpty = true;

if ($isTerminal) {
echo $type . ($type_length !== null ? "(" . $type_length . ")" : "")."\n";

} else {
$id = substr(md5(rand().":".$key.":".$level), 0, 8);

echo "<a href=\"javascript:toggleDisplay('". $id ."');\" style=\"text-decoration:none\">";
echo "<span style='color:#666666'>" . $type . ($type_length !== null ? "(" . $type_length . ")" : "") . "</span>";
echo "</a>";
echo "<span id=\"plus". $id ."\" style=\"display: " . ($collapse ? "inline" : "none") . ";\"> &#10549;</span>";
echo "<div id=\"container". $id ."\" style=\"display: " . ($collapse ? "" : "inline") . ";\">";
echo "<br />";
}

for ($i=0; $i <= $level; $i++) {
echo $isTerminal ? "| " : "<span style='color:black'>|</span>        ";
}

echo $isTerminal ? "\n" : "<br />";
}

for ($i=0; $i <= $level; $i++) {
echo $isTerminal ? "| " : "<span style='color:black'>|</span>        ";
}

echo $isTerminal ? "[" . $key . "] => " : "<span style='color:black'>[" . $key . "] => </span>";

call_user_func($recursive, $value, $level+1);
}

if ($notEmpty) {
for ($i=0; $i <= $level; $i++) {
echo $isTerminal ? "| " : "<span style='color:black'>|</span>        ";
}

if (!$isTerminal) {
echo "</div>";
}

} else {
echo $isTerminal ?
$type . ($type_length !== null ? "(" . $type_length . ")" : "") . " " :
"<span style='color:#666666'>" . $type . ($type_length !== null ? "(" . $type_length . ")" : "") . "</span>  ";
}

} else {
echo $isTerminal ?
$type . ($type_length !== null ? "(" . $type_length . ")" : "") . " " :
"<span style='color:#666666'>" . $type . ($type_length !== null ? "(" . $type_length . ")" : "") . "</span>  ";
```

```
if ($type_data != null) {
echo $isTerminal ? $type_data : "<span style='color:" . $type_color . "'>" . $type_data . "</span>";
}
}

echo $isTerminal ? "\n" : "<br />";
};

call_user_func($recursive, $input);
}
?>
```

up
down
16

anon ¶

19 years ago

```
var_dump(get_defined_vars());
```

will dump all defined variables to the browser.

up
down
7

thriller dot ze at gmail dot com ¶

16 years ago

As Bryan said, it is possible to capture var_dump() output to a string. But it's not quite exact if the dumped variable contains HTML code.

You can use this instead:

```
<?php
echo '<pre>'; // This is for correct handling of newlines
ob_start();
var_dump($var);
$a=ob_get_contents();
ob_end_clean();
echo htmlspecialchars($a,ENT_QUOTES); // Escape every HTML special chars (especially > and < )
echo '</pre>';
?>
```

up
down
4

stlawson *AT* joyfulearthtech *DOT* com ¶

12 years ago

```
<?php
/**
* Better GI than print_r or var_dump -- but, unlike var_dump, you can only dump one variable.
* Added htmlentities on the var content before echo, so you see what is really there, and not the mark-up.
*
* Also, now the output is encased within a div block that sets the background color, font style, and left-justifies it
* so it is not at the mercy of ambient styles.
*
* Inspired from: PHP.net Contributions
* Stolen from: [highstrike at gmail dot com]
* Modified by: stlawson *AT* JoyfulEarthTech *DOT* com
*
* @param mixed $var -- variable to dump
* @param string $var_name -- name of variable (optional) -- displayed in printout making it easier to sort out what
variable is what in a complex output
* @param string $indent -- used by internal recursive call (no known external value)
* @param unknown_type $reference -- used by internal recursive call (no known external value)
*/
function do_dump(&$var, $var_name = NULL, $indent = NULL, $reference = NULL)
{
```

```
$do_dump_indent = "<span style='color:#666666;'>|</span>    ";
$reference = $reference.$var_name;
$keyvar = 'the_do_dump_recursion_protection_scheme'; $keyname = 'referenced_object_name';

// So this is always visible and always left justified and readable
echo "<div style='text-align:left; background-color:white; font: 100% monospace; color:black;'>";

if (is_array($var) && isset($var[$keyvar]))
{
$real_var = &$var[$keyvar];
$real_name = &$var[$keyname];
$type = ucfirst(gettype($real_var));
echo "$indent$var_name <span style='color:#666666'>$type</span> = <span style='color:#e87800;'>&amp;$real_name</span>
<br>";
}
else
{
$var = array($keyvar => $var, $keyname => $reference);
$avar = &$var[$keyvar];

$type = ucfirst(gettype($avar));
if($type == "String") $type_color = "<span style='color:green'>";
elseif($type == "Integer") $type_color = "<span style='color:red'>";
elseif($type == "Double"){ $type_color = "<span style='color:#0099c5'>"; $type = "Float"; }
elseif($type == "Boolean") $type_color = "<span style='color:#92008d'>";
elseif($type == "NULL") $type_color = "<span style='color:black'>";

if(is_array($avar))
{
$count = count($avar);
echo "$indent" . ($var_name ? "$var_name => ":"") . "<span style='color:#666666'>$type ($count)</span><br>$indent(<br>";
$keys = array_keys($avar);
foreach($keys as $name)
{
$value = &$avar[$name];
do_dump($value, "['$name']", $indent.$do_dump_indent, $reference);
}
echo "$indent)<br>";
}
elseif(is_object($avar))
{
echo "$indent$var_name <span style='color:#666666'>$type</span><br>$indent(<br>";
foreach($avar as $name=>$value) do_dump($value, "$name", $indent.$do_dump_indent, $reference);
echo "$indent)<br>";
}
elseif(is_int($avar)) echo "$indent$var_name = <span style='color:#666666'>$type(".strlen($avar).")</span>
$type_color".htmlentities($avar)."</span><br>";
elseif(is_string($avar)) echo "$indent$var_name = <span style='color:#666666'>$type(".strlen($avar).")</span>
$type_color\"".htmlentities($avar)."\"</span><br>";
elseif(is_float($avar)) echo "$indent$var_name = <span style='color:#666666'>$type(".strlen($avar).")</span>
$type_color".htmlentities($avar)."</span><br>";
elseif(is_bool($avar)) echo "$indent$var_name = <span style='color:#666666'>$type(".strlen($avar).")</span> $type_color".
($avar == 1 ? "TRUE":"FALSE")."</span><br>";
elseif(is_null($avar)) echo "$indent$var_name = <span style='color:#666666'>$type(".strlen($avar).")</span>
{$type_color}NULL</span><br>";
else echo "$indent$var_name = <span style='color:#666666'>$type(".strlen($avar).")</span> ".htmlentities($avar)."<br>";

$var = $var[$keyvar];
}

echo "</div>";
}
```

```
?>
```
up
down
2
Anonymous ¶
7 years ago
Be careful this outputs to stdout stream (1) instead of the proper stderr stream (2).
up
down
4
kobrasrealm at gmail dot com ¶
12 years ago
I wrote this dandy little function for using var_dump() on HTML documents so I don't have to view the source.

```
<?php
function htmldump($variable, $height="9em") {
echo "<pre style=\"border: 1px solid #000; height: {$height}; overflow: auto; margin: 0.5em;\">";
var_dump($variable);
echo "</pre>\n";
}
?>
```

You can pass arguments like this:

```
<?php
$sql = mysql_query("SELECT id, name, value FROM table WHERE cond = 'value'");
$s = mysql_fetch_assoc($sql);

// Dump variable containing the result of the MySQL query
var_dump($s);
?>
```

The second parameter lets you specify the height of the box. Default is 9em, but if you're expecting a huge output you'll probably want a higher value.

```
<?php
var_dump($s, "17em");
?>
```

Happy var_dumping.
up
down
3
php at mikeboers dot com ¶
16 years ago
Howdy!

I am working on a pretty large project where I needed to dump a human readable form of whatever into the log files... and I thought var_export was too difficult to read. BigueNique at yahoo dot ca has a nice solution, although I needed to NOT modify whatever was being passed to dump.

So borrowing heavily from BigueNique's (just reworked his function) and someone's idea over in the object cloning page, I came up with the following function.

It makes a complete copy of whatever object you initially pass it, including all recursive definitions and outside objects references, then does the same thing as BigueNique's function. I also heavily reworked what it output, to suit my needs.

```
<?php

function var_log(&$varInput, $var_name='', $reference='', $method = '=', $sub = false) {

static $output ;
static $depth ;
```

```
if ( $sub == false ) {
$output = '' ;
$depth = 0 ;
$reference = $var_name ;
$var = serialize( $varInput ) ;
$var = unserialize( $var ) ;
} else {
++$depth ;
$var =& $varInput ;

}

// constants
$nl = "\n" ;
$block = 'a_big_recursion_protection_block';

$c = $depth ;
$indent = '' ;
while( $c -- > 0 ) {
$indent .= '| ' ;
}

// if this has been parsed before
if ( is_array($var) && isset($var[$block])) {

$real =& $var[ $block ] ;
$name =& $var[ 'name' ] ;
$type = gettype( $real ) ;
$output .= $indent.$var_name.' '.$method.'& '.($type=='array'?'Array':get_class($real)).' '.$name.$nl;

// havent parsed this before
} else {

// insert recursion blocker
$var = Array( $block => $var, 'name' => $reference );
$theVar =& $var[ $block ] ;

// print it out
$type = gettype( $theVar ) ;
switch( $type ) {

case 'array' :
$output .= $indent . $var_name . ' '.$method.' Array ('.$nl;
$keys=array_keys($theVar);
foreach($keys as $name) {
$value=&$theVar[$name];
var_log($value, $name, $reference.'["'.$name.'"]', '=', true);
}
$output .= $indent.')'.$nl;
break ;

case 'object' :
$output .= $indent.$var_name.' = '.get_class($theVar).' {'.$nl;
foreach($theVar as $name=>$value) {
var_log($value, $name, $reference.'->'.$name, '->', true);
}
$output .= $indent.'}'.$nl;
break ;

case 'string' :
$output .= $indent . $var_name . ' '.$method.' "'.$theVar.'"'.$nl;
```

```
break ;

default :
$output .= $indent . $var_name . ' '.$method.' ('.$type.') '.$theVar.$nl;
break ;

}

// $var=$var[$block];

}

-- $depth ;

if( $sub == false )
return $output ;

}

// var_log( $var, '$name' ) ;

?>
```

Hope it works well for you!

up
down
0

david at exposito dot mobi ¶

14 years ago

If you want to save exactly the content of an array into a variable to save ir later for example, use this:

```
<?php $xml = var_export($xml, true); ?>
```

You'll have in your same variable the content readable, instead of written down or anything else

up
down
0

egorinsk at gmail com ¶

15 years ago

Note that var_dump reset()'s array internal pointer!

up
down
0

highstrike at gmail dot com ¶

16 years ago

made 2 nifty functions based of what some people contributed here. Hope you find them usefull

usage ... call for the dump function. EG: dump($array, "Array dump");

```
<?php
////////////////////////////////////////////////////////
// Function: dump
// Inspired from: PHP.net Contributions
// Description: Helps with php debugging

function dump(&$var, $info = FALSE)
{
$scope = false;
$prefix = 'unique';
$suffix = 'value';

if($scope) $vals = $scope;
```

```
else $vals = $GLOBALS;

$old = $var;
$var = $new = $prefix.rand().$suffix; $vname = FALSE;
foreach($vals as $key => $val) if($val === $new) $vname = $key;
$var = $old;

echo "<pre style='margin: 0px 0px 10px 0px; display: block; background: white; color: black; font-family: Verdana; border:
1px solid #cccccc; padding: 5px; font-size: 10px; line-height: 13px;'>";
if($info != FALSE) echo "<b style='color: red;'>$info:</b><br>";
do_dump($var, '$'.$vname);
echo "</pre>";
}

////////////////////////////////////////////////////
// Function: do_dump
// Inspired from: PHP.net Contributions
// Description: Better GI than print_r or var_dump

function do_dump(&$var, $var_name = NULL, $indent = NULL, $reference = NULL)
{
$do_dump_indent = "<span style='color:#eeeeee;'>|</span>    ";
$reference = $reference.$var_name;
$keyvar = 'the_do_dump_recursion_protection_scheme'; $keyname = 'referenced_object_name';

if (is_array($var) && isset($var[$keyvar]))
{
$real_var = &$var[$keyvar];
$real_name = &$var[$keyname];
$type = ucfirst(gettype($real_var));
echo "$indent$var_name <span style='color:#a2a2a2'>$type</span> = <span style='color:#e87800;'>&amp;$real_name</span>
<br>";
}
else
{
$var = array($keyvar => $var, $keyname => $reference);
$avar = &$var[$keyvar];

$type = ucfirst(gettype($avar));
if($type == "String") $type_color = "<span style='color:green'>";
elseif($type == "Integer") $type_color = "<span style='color:red'>";
elseif($type == "Double"){ $type_color = "<span style='color:#0099c5'>"; $type = "Float"; }
elseif($type == "Boolean") $type_color = "<span style='color:#92008d'>";
elseif($type == "NULL") $type_color = "<span style='color:black'>";

if(is_array($avar))
{
$count = count($avar);
echo "$indent" . ($var_name ? "$var_name => ":"") . "<span style='color:#a2a2a2'>$type ($count)</span><br>$indent(<br>";
$keys = array_keys($avar);
foreach($keys as $name)
{
$value = &$avar[$name];
do_dump($value, "['$name']", $indent.$do_dump_indent, $reference);
}
echo "$indent)<br>";
}
elseif(is_object($avar))
{
echo "$indent$var_name <span style='color:#a2a2a2'>$type</span><br>$indent(<br>";
foreach($avar as $name=>$value) do_dump($value, "$name", $indent.$do_dump_indent, $reference);
echo "$indent)<br>";
```

```
}
elseif(is_int($avar)) echo "$indent$var_name = <span style='color:#a2a2a2'>$type(".strlen($avar).")</span>
$type_color$avar</span><br>";
elseif(is_string($avar)) echo "$indent$var_name = <span style='color:#a2a2a2'>$type(".strlen($avar).")</span>
$type_color\"$avar\"</span><br>";
elseif(is_float($avar)) echo "$indent$var_name = <span style='color:#a2a2a2'>$type(".strlen($avar).")</span>
$type_color$avar</span><br>";
elseif(is_bool($avar)) echo "$indent$var_name = <span style='color:#a2a2a2'>$type(".strlen($avar).")</span> $type_color".
($avar == 1 ? "TRUE":"FALSE")."</span><br>";
elseif(is_null($avar)) echo "$indent$var_name = <span style='color:#a2a2a2'>$type(".strlen($avar).")</span>
{$type_color}NULL</span><br>";
else echo "$indent$var_name = <span style='color:#a2a2a2'>$type(".strlen($avar).")</span> $avar<br>";

$var = $var[$keyvar];
}
}
?>
```

up
down
-1

vladimir at pixeltomorrow dot com ¶

16 years ago

You can also use the PEAR package available at http://pear.php.net/package/Var_Dump

which parses the variable content in a very pleasant manner, a lot more easier to "follow" than the built-in var_dump() function.

Of course there are many others, but I prefer this one, because it's simply to use.

Just add at the begining of your file:

```
<?php

require('Var_Dump.php'); // make sure the pear package path is set in php.ini
Var_Dump::displayInit(array('display_mode' => 'HTML4_Text'), array('mode' => 'normal','offset' => 4));
?>
```

then, instead of simply using var_dump($foo), use:

```
<?php

Var_Dump::display($foo);

?>
```

Read the documentation if you're looking for different output layouts.

Cheers!
Vladimir Ghetau

up
down
-2

divinity76 at gmail dot com ¶

12 years ago

a html-encoded var_dump

```
<?php
function htmlvardump(){ob_start(); $var = func_get_args(); call_user_func_array('var_dump', $var); echo
htmentities(ob_get_clean());}
?>
```

(creds: ekneuss / mOo @ irc.freenode.net/#php)

up
down
-4
jonbarnett at gmail dot com ¶
17 years ago

```
dumping objects that reference each other could lead to infinite recursion
<?php
$brother = new Sibling();
$sister = new Sibling();
$brother->sister = $sister;
$sister->brother = $brother;

var_dump($brother);
/* dumps all of $brother's properties, including "sister", which dumps all of $sister's properties, including "brother",
etc. */
?>
```

up
down
-3
fabien dot villepinte at gmail dot com ¶
4 years ago

```
It's important to note that the output depends on the precision directive.
<?php
var_dump(10000000000000.5); // float(10000000000000)
ini_set('precision',-1);
var_dump(10000000000000.5); // float(10000000000000.5)
?>
```

＋add a note