php

- Downloads
- Documentation
- Get Involved
- Help

Search

Getting Started
- Introduction
- A simple tutorial

Language Reference
- Basic syntax
- Types
- Variables
- Constants
- Expressions
- Operators
- Control Structures
- Functions
- Classes and Objects
- Namespaces
- Errors
- Exceptions
- Generators
- References Explained
- Predefined Variables
- Predefined Exceptions
- Predefined Interfaces and Classes
- 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
- Using Register Globals
- User Submitted Data
- Magic Quotes
- 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
- Safe Mode
- 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
- Credit Card Processing
- 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

String Functions ¶

See Also

For even more powerful string handling and manipulating functions take a look at the POSIX regular expression functions and the Perl compatible regular expression functions.

Table of Contents ¶

- addcslashes — Quote string with slashes in a C style
- addslashes — Quote string with slashes
- bin2hex — Convert binary data into hexadecimal representation
- chop — Alias of rtrim
- chr — Return a specific character
- chunk_split — Split a string into smaller chunks
- convert_cyr_string — Convert from one Cyrillic character set to another
- convert_uudecode — Decode a uuencoded string
- convert_uuencode — Uuencode a string
- count_chars — Return information about characters used in a string
- crc32 — Calculates the crc32 polynomial of a string
- crypt — One-way string hashing
- echo — Output one or more strings
- explode — Split a string by string
- fprintf — Write a formatted string to a stream
- get_html_translation_table — Returns the translation table used by htmlspecialchars and htmlentities
- hebrev — Convert logical Hebrew text to visual text
- hebrevc — Convert logical Hebrew text to visual text with newline conversion
- hex2bin — Decodes a hexadecimally encoded binary string
- html_entity_decode — Convert all HTML entities to their applicable characters
- htmlentities — Convert all applicable characters to HTML entities
- htmlspecialchars_decode — Convert special HTML entities back to characters
- htmlspecialchars — Convert special characters to HTML entities

- implode — Join array elements with a string
- join — Alias of implode
- lcfirst — Make a string's first character lowercase
- levenshtein — Calculate Levenshtein distance between two strings
- localeconv — Get numeric formatting information
- ltrim — Strip whitespace (or other characters) from the beginning of a string
- md5_file — Calculates the md5 hash of a given file
- md5 — Calculate the md5 hash of a string
- metaphone — Calculate the metaphone key of a string
- money_format — Formats a number as a currency string
- nl_langinfo — Query language and locale information
- nl2br — Inserts HTML line breaks before all newlines in a string
- number_format — Format a number with grouped thousands
- ord — Return ASCII value of character
- parse_str — Parses the string into variables
- print — Output a string
- printf — Output a formatted string
- quoted_printable_decode — Convert a quoted-printable string to an 8 bit string
- quoted_printable_encode — Convert a 8 bit string to a quoted-printable string
- quotemeta — Quote meta characters
- rtrim — Strip whitespace (or other characters) from the end of a string
- setlocale — Set locale information
- sha1_file — Calculate the sha1 hash of a file
- sha1 — Calculate the sha1 hash of a string
- similar_text — Calculate the similarity between two strings
- soundex — Calculate the soundex key of a string
- sprintf — Return a formatted string
- sscanf — Parses input from a string according to a format
- str_getcsv — Parse a CSV string into an array
- str_ireplace — Case-insensitive version of str_replace.
- str_pad — Pad a string to a certain length with another string
- str_repeat — Repeat a string
- str_replace — Replace all occurrences of the search string with the replacement string
- str_rot13 — Perform the rot13 transform on a string
- str_shuffle — Randomly shuffles a string
- str_split — Convert a string to an array
- str_word_count — Return information about words used in a string
- strcasecmp — Binary safe case-insensitive string comparison
- strchr — Alias of strstr
- strcmp — Binary safe string comparison
- strcoll — Locale based string comparison
- strcspn — Find length of initial segment not matching mask
- strip_tags — Strip HTML and PHP tags from a string
- stripcslashes — Un-quote string quoted with addcslashes
- stripos — Find the position of the first occurrence of a case-insensitive substring in a string
- stripslashes — Un-quotes a quoted string
- stristr — Case-insensitive strstr
- strlen — Get string length
- strnatcasecmp — Case insensitive string comparisons using a "natural order" algorithm
- strnatcmp — String comparisons using a "natural order" algorithm
- strncasecmp — Binary safe case-insensitive string comparison of the first n characters
- strncmp — Binary safe string comparison of the first n characters
- strpbrk — Search a string for any of a set of characters
- strpos — Find the position of the first occurrence of a substring in a string

- strrchr — Find the last occurrence of a character in a string
- strrev — Reverse a string
- strripos — Find the position of the last occurrence of a case-insensitive substring in a string
- strrpos — Find the position of the last occurrence of a substring in a string
- strspn — Finds the length of the initial segment of a string consisting entirely of characters contained within a given mask.
- strstr — Find the first occurrence of a string
- strtok — Tokenize string
- strtolower — Make a string lowercase
- strtoupper — Make a string uppercase
- strtr — Translate characters or replace substrings
- substr_compare — Binary safe comparison of two strings from an offset, up to length characters
- substr_count — Count the number of substring occurrences
- substr_replace — Replace text within a portion of a string
- substr — Return part of a string
- trim — Strip whitespace (or other characters) from the beginning and end of a string
- ucfirst — Make a string's first character uppercase
- ucwords — Uppercase the first character of each word in a string
- vfprintf — Write a formatted string to a stream
- vprintf — Output a formatted string
- vsprintf — Return a formatted string
- wordwrap — Wraps a string to a given number of characters

add a note

User Contributed Notes 23 notes

up
down
3
administrador(ensaimada)sphoera(punt)com ¶
10 years ago

I've prepared this simple function to obtain a string delimited between tags (not only XML tags!). Anybody needs something like this?.

```
<?php

function get_string_between($string, $start, $end){
    $string = " ".$string;
     $ini = strpos($string,$start);
     if ($ini == 0) return "";
     $ini += strlen($start);
     $len = strpos($string,$end,$ini) - $ini;
     return substr($string,$ini,$len);
}

$string = "this [custom] function is useless!!";
echo get_string_between($string,"[","]");
// must return "custom";
?>
```

more functions at http://www.sphoera.com

up
down
1

admin at fivestarbuy dot com ¶

11 years ago

This example lets you parse an unparsed strings variables. Warning: This could cause security leaks if you allow users to pass $variables through this engine. I recommend only using this for your Content Management System.

```
<?
$mytime=time();
$mydog="My Dog Ate My PHP!";

# Your Parsing String:
$s1 = 'Hyphen Variable Preserving: $mytime, and $mydog';
echo "Before: <br><br>$s1<br><br>";

# Remember, wherever you define this, it will not be defined GLOBAL into the function
# which is why we define it here. Defining it global could lead to security issues.
$vardata=get_defined_vars();

# Parse the string
$s1 = StrParse($s1,$vardata);

echo "After: <br><br>$s1";

function StrParse($str,$vardata) {
# Takes a string, or piece of data, that contains PHP Variables

# For example, unparsed variables like:  Test using time: $mytime
# This example shows $mytime, and not the actual variable value.
# The end result shows the actual variable value of $mytime.

# This is useful for building a content management system,
# and directing your variables into your content data,
# where content is stored in a file or database, unparsed.
# Of course this could slow down page loads, but it's a good way
# to parse data from current variables into your loaded new data
# making it compatible.

# Then the variables are replaced with the actual variable..
$getvarkeys=array_keys($vardata);
$ret=$str;
for ($x=0; $x < count($getvarkeys); $x++) {
    $myvar=$getvarkeys[$x];
    #echo "Variable: " . $myvar . " [" . $vardata[$myvar] . "]<br>";
    $ret=str_replace('$' . $myvar, $vardata[$myvar], $ret);
}
return $ret;

}

?>
```

up
down
0

Stephen Dewey ¶

8 years ago

If you want a function to return all text in a string up to the Nth occurrence of a substring, try the below function.

Works in PHP >= 5.

(Pommef provided another sample function for this purpose below, but I believe it is incorrect.)

```
<?php

// Returns all of $haystack up to (but excluding) the $n_occurrence occurrence of $needle. Therefore:
//        If there are < $n_occurrence occurrences of $needle in $haystack, the entire string will be returned.
//        If there are >= $n_occurrence occurrences of $needle in $haystack, the returned string will end before the $n_occurrence'th needle.
// This function only makes sense for $n_occurrence >= 1
function nsubstr($needle, $haystack, $n_occurrence)
{
    // After exploding by $needle, every entry in $arr except (possibly) part of the last entry should have its content returned.
    $arr = explode($needle,$haystack,$n_occurrence);
    // Examine last entry in $arr. If it contains $needle, cut out all text except for the text before $needle.
    $last = count($arr) - 1;
    $pos_in_last = strpos($arr[$last],$needle);

    if ($pos_in_last !== false)
        $arr[$last] = substr($arr[$last],0,$pos_in_last);

    return implode($needle,$arr);
}

$string = 'd24jkdslgjldk2424jgklsjg24jskgldjk24';

print 'S:  ' . $string . '<br>';
print '1: ' . nsubstr('24',$string,1) . '<br>';
print '2: ' . nsubstr('24',$string,2) . '<br>';
print '3: ' . nsubstr('24',$string,3) . '<br>';
print '4: ' . nsubstr('24',$string,4) . '<br>';
print '5: ' . nsubstr('24',$string,5) . '<br>';
print '6: ' . nsubstr('24',$string,6) . '<br>';
print '7: ' . nsubstr('24',$string,7) . '<br>';

/*
// prints:
S: d24jkdslgjldk2424jgklsjg24jskgldjk24
1: d
2: d24jkdslgjldk
3: d24jkdslgjldk24
4: d24jkdslgjldk2424jgklsjg
5: d24jkdslgjldk2424jgklsjg24jskgldjk
6: d24jkdslgjldk2424jgklsjg24jskgldjk24
7: d24jkdslgjldk2424jgklsjg24jskgldjk24
*/
```

```
?>
```

Note that this function can be combined with wordwrap() to accomplish a routine but fairly difficult web design goal, namely, limiting inline HTML text to a certain number of lines. wordwrap() can break your string using
, and then you can use this function to only return text up to the N'th
.

You will still have to make a conservative guess of the max number of characters per line with wordwrap(), but you can be more precise than if you were simply truncating a multiple-line string with substr().

See example:

```
<?php

$text = 'Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Pellentesque id massa. Duis sollicitudin ipsum vel diam. Aliquam pulvinar sagittis felis. Nullam hendrerit semper elit. Donec convallis mollis risus. Cras blandit mollis turpis. Vivamus facilisis, sapien at tincidunt accumsan, arcu dolor suscipit sem, tristique convallis ante ante id diam. Curabitur mollis, lacus vel gravida accumsan, enim quam condimentum est, vitae rutrum neque magna ac enim.';

$wrapped_text = wordwrap($text,100,'<br>',true);

$three_lines = nsubstr('<br>',$wrapped_text,3);

print '<br><br>' . $three_lines;

$four_lines = nsubstr('<br>',$wrapped_text,4);

print '<br><br>' . $four_lines;

/*
prints:

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Pellentesque id massa. Duis sollicitudin ipsum vel diam. Aliquam pulvinar sagittis felis. Nullam hendrerit semper elit. Donec convallis mollis risus. Cras blandit mollis turpis. Vivamus facilisis, sapien at tincidunt accumsan, arcu

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Pellentesque id massa. Duis sollicitudin ipsum vel diam. Aliquam pulvinar sagittis felis. Nullam hendrerit semper elit. Donec convallis mollis risus. Cras blandit mollis turpis. Vivamus facilisis, sapien at tincidunt accumsan, arcu dolor suscipit sem, tristique convallis ante ante id diam. Curabitur mollis, lacus vel gravida

*/

?>
```

up
down
0
terry dot greenlaw at logicalshift dot com ¶
12 years ago

Here's a simpler "simplest" way to toggle through a set of 1..n colors for web backgrounds:

```
<?php
```

```
$colours = array('#000000', '#808080', '#A0A0A0', '#FFFFFF');

// Get a colour
$color = next($colors) or $color = reset($colors);
?>
```

The code doesn't need to know anything about the number of elements being cycled through. That way you won't have to tracking down all the code when changing the number of colors or the color values.

up
down
-1

str at maphpia dot com ¶
3 years ago

I was looking for a function to find the common substring in 2 different strings. I tried both the mb_string_intersect and string_intersect functions listed here but didn't work for me. I found the algorithm at http://en.wikibooks.org/wiki/Algorithm_implementation/Strings/Longest_common_substring#PHP so here I post you the function

```
<?php

/**
* Finds the matching string between 2 strings
*
* @param string $string1
* @param string $string2
* @param number $minChars
*
* @return NULL|string
*
* @link http://en.wikibooks.org/wiki/Algorithm_implementation/Strings/Longest_common_substring#PHP
*/
function string_intersect($string_1, $string_2)
{
    $string_1_length = strlen($string_1);
    $string_2_length = strlen($string_2);
    $return          = "";

    if ($string_1_length === 0 || $string_2_length === 0) {
        // No similarities
        return $return;
    }

    $longest_common_subsequence = array();

    // Initialize the CSL array to assume there are no similarities
    for ($i = 0; $i < $string_1_length; $i++) {
        $longest_common_subsequence[$i] = array();
        for ($j = 0; $j < $string_2_length; $j++) {
            $longest_common_subsequence[$i][$j] = 0;
        }
    }

    $largest_size = 0;
```

```
    for ($i = 0; $i < $string_1_length; $i++) {
        for ($j = 0; $j < $string_2_length; $j++) {
            // Check every combination of characters
            if ($string_1[$i] === $string_2[$j]) {
                // These are the same in both strings
                if ($i === 0 || $j === 0) {
                    // It's the first character, so it's clearly only 1 character long
                    $longest_common_subsequence[$i][$j] = 1;
                } else {
                    // It's one character longer than the string from the previous character
                    $longest_common_subsequence[$i][$j] = $longest_common_subsequence[$i - 1][$j - 1]
+ 1;
                }

                if ($longest_common_subsequence[$i][$j] > $largest_size) {
                    // Remember this as the largest
                    $largest_size = $longest_common_subsequence[$i][$j];
                    // Wipe any previous results
                    $return       = "";
                    // And then fall through to remember this new value
                }

                if ($longest_common_subsequence[$i][$j] === $largest_size) {
                    // Remember the largest string(s)
                    $return = substr($string_1, $i - $largest_size + 1, $largest_size);
                }
            }
            // Else, $CSL should be set to 0, which it was already initialized to
        }
    }

    // Return the list of matches
    return $return;
}
```

up
down
-1
SteveRusin ¶
10 years ago

The functions below:

```
function beginsWith( $str, $sub )
function endsWith( $str, $sub )
```

Are correct, but flawed. You'd need to use the === operator instead:

```
function beginsWith( $str, $sub ) {
   return ( substr( $str, 0, strlen( $sub ) ) === $sub );
}
function endsWith( $str, $sub ) {
   return ( substr( $str, strlen( $str ) - strlen( $sub ) ) === $sub );
}
```

Otherwise, endsWith would return "foobar.0" ends with ".0" as well as "0" or "00" or any amount of zeros because numerically .0 does equal 0.

up
down
-2

rh at richardhoward dot net ¶
11 years ago

```
<?php
/**
Utility class: static methods for cleaning & escaping untrusted (i.e.
user-supplied) strings.

Any string can (usually) be thought of as being in one of these 'modes':

pure = what the user actually typed / what you want to see on the page /
       what is actually stored in the DB
gpc  = incoming GET, POST or COOKIE data
sql  = escaped for passing safely to RDBMS via SQL (also, data from DB
       queries and file reads if you have magic_quotes_runtime on--which
       is rare)
html = safe for html display (htmlentities applied)

Always knowing what mode your string is in--using these methods to
convert between modes--will prevent SQL injection and cross-site scripting.

This class refers to its own namespace (so it can work in PHP 4--there is no
self keyword until PHP 5). Do not change the name of the class w/o changing
all the internal references.

Example usage: a POST value that you want to query with:
$username = Str::gpc2sql($_POST['username']);
*/

//This sets SQL escaping to use slashes; for Sybase(/MSSQL)-style escaping
// ( ' --> '' ), set to true.
define('STR_SYBASE', false);

class Str {
    function gpc2sql($gpc, $maxLength = false)
    {
        return Str::pure2sql(Str::gpc2pure($gpc), $maxLength);
    }
    function gpc2html($gpc, $maxLength = false)
    {
        return Str::pure2html(Str::gpc2pure($gpc), $maxLength);
    }
    function gpc2pure($gpc)
    {
        if (ini_get('magic_quotes_sybase'))
            $pure = str_replace("''", "'", $gpc);
        else $pure = get_magic_quotes_gpc() ? stripslashes($gpc) : $gpc;
        return $pure;
    }
    function html2pure($html)
```

```
    {
        return html_entity_decode($html);
    }
    function html2sql($html, $maxLength = false)
    {
        return Str::pure2sql(Str::html2pure($html), $maxLength);
    }
    function pure2html($pure, $maxLength = false)
    {
        return $maxLength ? htmlentities(substr($pure, 0, $maxLength))
                          : htmlentities($pure);
    }
    function pure2sql($pure, $maxLength = false)
    {
        if ($maxLength) $pure = substr($pure, 0, $maxLength);
        return (STR_SYBASE)
               ? str_replace("'", "''", $pure)
               : addslashes($pure);
    }
    function sql2html($sql, $maxLength = false)
    {
        $pure = Str::sql2pure($sql);
        if ($maxLength) $pure = substr($pure, 0, $maxLength);
        return Str::pure2html($pure);
    }
    function sql2pure($sql)
    {
        return (STR_SYBASE)
               ? str_replace("''", "'", $sql)
               : stripslashes($sql);
    }
}
?>
```

up
down
-2

t0russ at gmail dot com ¶

11 years ago

to kristin at greenaple dot on dot ca:
thanx for sharing.
your function in recursive form proved to be slightly faster and it returns false (as it should) when the character is not found instead of number 0:

```
<?php
function strnposr($haystack, $needle, $occurance, $pos = 0) {
    return ($occurance<2)?strpos($haystack, $needle, $pos):strnposr($haystack,$needle,$occurance-1,strpos($haystack, $needle, $pos) + 1);
}
?>
```

up
down
-2

admin at rotarymulundeast dot org ¶

10 years ago

Here's an easier way to find nth...

```
function nth($numbex){
  if ($numbex%10 == 1 && $numbex%100 != 11) $sth='st';
  elseif ($numbex%10 == 2 && $numbex%100 != 12) $sth='nd';
  elseif ($numbex%10 == 3 && $numbex%100 != 13) $sth='rd';
  else $sth = 'th';
  return $sth;
}
```

there is is no need to check if the user has entered a non-integer as we may be using this function for expressing variables as well eg. ith value of x , nth root of z ,etc...

up
down
-2

Pommef ¶
11 years ago

Example: Give me everything up to the fourth occurance of '/'.

```
<?php

   $haystack = "/home/username/www/index.php";
   $needle = "/";

   function strnpos($haystack, $needle, $occurance, $pos = 0) {

        $res = implode($needle,$haystack);

        $res = array_slice($res, $pos,  $occurance);

        return explode ($needle,$res);
    }
?>
```

up
down
-3

james dot d dot baker at gmail dot com ¶
11 years ago

```
<?php
/*
Written By James Baker, May 27th 2005

sentenceCase($string);
    $string: The string to convert to sentence case.

Converts a string into proper sentence case (First letter of each sentance capital, all the others
smaller)

Example Usage:
echo sentenceCase("HELLO WORLD!!! THIS IS A CAPITALISED SENTENCE. this isn't.");

Returns:
Hello world!!! This is a capitalised sentence. This isn't.
*/
```

```
function sentenceCase($s){
    $str = strtolower($s);
    $cap = true;

    for($x = 0; $x < strlen($str); $x++){
        $letter = substr($str, $x, 1);
        if($letter == "." || $letter == "!" || $letter == "?"){
            $cap = true;
        }elseif($letter != " " && $cap == true){
            $letter = strtoupper($letter);
            $cap = false;
        }

        $ret .= $letter;
    }

    return $ret;
}
?>
```

up
down
-2
m ¶
9 years ago

Regarding the code for the function beginsWith($str, $sub), I found that it has problems when only one character is present after the string searched for. I found that this works better instead:

```
<?php
function beginsWith($str, $sub) {
    return (strncmp($str, $sub, strlen($sub)) == 0);
}
?>
```

up
down
-3
php at moechofe dot com ¶
11 years ago

```
<?php
  /*
  * str_match
  *
  * return a string with only cacacteres defined in a expression return false if the expression is not valid
  *
  * @param $str string the string
  * @param $match the expression based on the class definition off a PCRE regular expression.
  *   the '[', ']', '\' and '^' at class start need to be escaped.
  *   like : -a-z0-9_@.
  */
  function str_match( $str, $match )
  {
    $return = '';
    if( eregi( '(.*)', $match, $class ) )
```

```
    {
      $match = '['.$regs[1].']';
      for( $i=0; $i<strlen($str); $i++ )
      if( ereg( '['.$class[1].']', $str[$i] ) )
      $return .= $str{$i};
      return $return;
    }
    else return false;
  }

  /*
  * example
  * accept only alphanum caracteres from the GET/POST parameters 'a'
  */

  if( ! empty($_REQUEST['a']) )
    $_REQUEST['a'] = str_match( $_REQUEST['a'], 'a-zA-Z0-9' );
  else
    $_REQUEST['a'] = 'default';
?>
```

up
down
-3

Anonymous ¶
11 years ago

to: james dot d dot baker at gmail dot com

PHP has a builtin function for doing what your function does,

http://php.net/ucfirst
http://php.net/ucwords

up
down
-3

andy a t onesandzeros d o t biz ¶
12 years ago

I use these little doo-dads quite a bit. I just thought I'd share them and maybe save someone a little time. No biggy. :)

```
// returns true if $str begins with $sub
function beginsWith( $str, $sub ) {
    return ( substr( $str, 0, strlen( $sub ) ) == $sub );
}

// return tru if $str ends with $sub
function endsWith( $str, $sub ) {
    return ( substr( $str, strlen( $str ) - strlen( $sub ) ) == $sub );
}

// trims off x chars from the front of a string
// or the matching string in $off is trimmed off
function trimOffFront( $off, $str ) {
    if( is_numeric( $off ) )
        return substr( $str, $off );
```

```
    else
        return substr( $str, strlen( $off ) );
}

// trims off x chars from the end of a string
// or the matching string in $off is trimmed off
function trimOffEnd( $off, $str ) {
    if( is_numeric( $off ) )
        return substr( $str, 0, strlen( $str ) - $off );
    else
        return substr( $str, 0, strlen( $str ) - strlen( $off ) );
}
```

up
down
-3
webmaster at cafe-clope dot net ¶
11 years ago

A comprehensive concatenation function, that works with array and strings

```
<?php
function str_cat() {
  $args = func_get_args() ;

  // Asserts that every array given as argument is $dim-size.
  // Keys in arrays are stripped off.
  // If no array is found, $dim stays unset.
  foreach($args as $key => $arg) {
    if(is_array($arg)) {
      if(!isset($dim))
        $dim = count($arg) ;
      elseif($dim != count($arg))
        return FALSE ;
      $args[$key] = array_values($arg) ;
    }
  }

  // Concatenation
  if(isset($dim)) {
    $result = array() ;
    for($i=0;$i<$dim;$i++) {
      $result[$i] = '' ;
      foreach($args as $arg)
        $result[$i] .= ( is_array($arg) ? $arg[$i] : $arg ) ;
    }
    return $result ;
  } else {
    return implode($args) ;
  }
}
?>
```

A simple example :

```
<?php
```

```
str_cat(array(1,2,3), '-', array('foo' => 'foo', 'bar' => 'bar', 'noop' => 'noop')) ;
?>

will return :
Array (
  [0] => 1-foo
  [1] => 2-bar
  [2] => 3-noop
)

More usefull :

<?php
$myget = $_GET ; // retrieving previous $_GET values
$myget['foo'] = 'b a r' ; // changing one value
$myget = str_cat(array_keys($myget), '=', array_map('rawurlencode', array_values($myget))) ;
$querystring = implode(ini_get('arg_separator.output'), $myget)) ;
?>

will return a valid querystring with some values changed.

Note that <?php str_cat('foo', '&', 'bar') ; ?> will return 'foo&bar', while <?php
str_cat(array('foo'), '&', 'bar') ; ?> will return array(0 => foo&bar)
```

up
down
-3
Anonymous ¶
12 years ago

```
In response to hackajar <matt> yahoo <trot> com,

No string-to-array function exists because it is not needed. If you reference a string with an offset
like you do with an array, the character at that offset will be return. This is documented in section
III.11's "Strings" article under the "String access and modification by character" heading.
```

up
down
-4
Verdauga ¶
8 years ago

```
Just a note in regards to bloopletech a few posts down:

The word "and" should not be used when converting numbers to text.  "And" (at least in US English)
should only be used to indicate the decimal place.

Example:
1,796,706 => one million, seven hundred ninety-six thousand, seven hundred six.
594,359.34 => five hundred ninety four thousand, three hundred fifty nine and thirty four hundredths
```

up
down
-4
mike "eyes" moe ¶
9 years ago

```
Here is a truly random string generator it uses the most common string functions it will work on
anywhere.
```

```
<?php
    function random_string($max = 20){
        $chars = explode(" ", "a b c d e f g h i j k l m n o p q r s t u v w x y z 0 1 2 3 4 5 6 7 8
9");
        for($i = 0; $i < $max; $i++){
            $rnd = array_rand($chars);
            $rtn .= base64_encode(md5($chars[$rnd]));
        }
        return substr(str_shuffle(strtolower($rtn)), 0, $max);
    }
?>
```

up
down
-5
kristin at greenapple dot on dot ca ¶
12 years ago

I really searched for a function that would do this as I've seen it in other languages but I couldn't find it here. This is particularily useful when combined with substr() to take the first part of a string up to a certain point.

strnpos() - Find the nth position of needle in haystack.

```
<?php

    function strnpos($haystack, $needle, $occurance, $pos = 0) {

        for ($i = 1; $i <= $occurance; $i++) {
            $pos = strpos($haystack, $needle, $pos) + 1;
        }
        return $pos - 1;

    }

?>
```

Example: Give me everything up to the fourth occurance of '/'.

```
<?php

    $haystack = "/home/username/www/index.php";
    $needle = "/";

    $root_dir = substr($haystack, 0, strnpos($haystack, $needle, 4));

    echo $root_dir;

?>
```

Returns: /home/username/www

Use this example with the server variable $_SERVER['SCRIPT_NAME'] as the haystack and you can self-discover a document's root directory for the purposes of locating global files automatically!

up
down

-5

[tab!] ¶

12 years ago

```
//
// string strtrmvistl(  string str, [int maxlen = 64],
//                      [bool right_justify = false],
//                      [string delimter = "<br>\n"])
//
// splits a long string into two chunks (a start and an end chunk)
// of a given maximum length and seperates them by a given delimeter.
// a second chunk can be right-justified within maxlen.
// may be used to 'spread' a string over two lines.
//

function strtrmvistl($str, $maxlen = 64, $right_justify = false, $delimter = "<br>\n") {
    if(($len = strlen($str = chop($str))) > ($maxlen = max($maxlen, 12))) {
        $newstr = substr($str, 0, $maxlen - 3);

        if($len > ($maxlen - 3)) {
            $endlen = min(($len - strlen($newstr)), $maxlen - 3);
            $newstr .= "..." . $delimter;

            if($right_justify)
                $newstr .= str_pad('', $maxlen - $endlen - 3, ' ');

            $newstr .= "..." . substr($str, $len - $endlen);
        }

        return($newstr);
    }

    return($str);
}
```

up

down

-3

navarr at gmail dot com ¶

11 years ago

```
stripos for PHP4.x

<?php
  function stripos($haystack,$needle) {
    return strpos(strtoupper($haystack),strtoupper($needle));
  }
?>
```

up

down

-7

Tomek Rychtyk ¶

5 years ago

```
Get the intersection of two strings using array_intersect

<?php
```

```
function string_intersect($string1, $string2)
{
    $array1 = $array2 = array();

    for($i = 0, $j = 0, $s1_len = strlen($string1), $s2_len = strlen($string2);($i < $s1_len) || ($j
< $s2_len); $i++, $j++) {
        if($i < $s1_len) {
            $array1[] = $string1[$i];
        }
        if($j < $s2_len) {
            $array2[] = $string2[$j];
        }
    }

    return implode('', array_intersect($array1, $array2));
}

?>
```

For more advanced comparison you can use array_uintersect as well.

add a note

- Strings
  - Introduction
  - Installing/Configuring
  - Predefined Constants
  - String Functions
  - Changelog