Registro de cambios »
« vsprintf

- Manual de PHP
- Referencia de funciones
- Procesamiento de texto
- Strings
- Funciones de strings

Change language: Spanish

Submit a Pull Request Report a Bug

wordwrap

(PHP 4 >= 4.0.2, PHP 5, PHP 7, PHP 8)

wordwrap — Ajusta un string hasta un número dado de caracteres

Descripción ¶

wordwrap(
  string `$str`,
  int `$width` = 75,
  string `$break` = "\n",
  bool `$cut` = **`false`**
): string

Ajusta un string hasta un número dado de caracteres utilizando un caracter de salto de string.

Parámetros ¶

`str`

El string de entrada.

`width`

El número de caracteres en el cual el string se verá envuelto.

`break`

La línea se rompe utilizando el parámetro opcional `break`.

`cut`

Si `cut` se establece en **`true`**, el string siempre será ajustado en o antes del `width` especificado. De tal forma que si se tiene una palabra que es más larga que el ancho dado, será dividida (ver segundo ejemplo). Cuando es **`false`** la función no divide la palabra incluso si `width` es menor que el ancho de la palabra.

Valores devueltos ¶

Devuelve el string dado ajustado a la longitud especificada.

Ejemplos ¶

Ejemplo #1 Ejemplo de wordwrap()

```
<?php
$text = "The quick brown fox jumped over the lazy dog.";
$newtext = wordwrap($text, 20, "<br />\n");

echo $newtext;
?>
```

El resultado del ejemplo sería:

```
The quick brown fox<br />
jumped over the lazy<br />
dog.
```

Ejemplo #2 Ejemplo de wordwrap()

```
<?php
$text = "A very long woooooooooooord.";
$newtext = wordwrap($text, 8, "\n", true);

echo "$newtext\n";
?>
```

El resultado del ejemplo sería:

```
A very
long
woooooooo
ooooord.
```

Ejemplo #3 Ejemplo de wordwrap()

```
<?php
$texto = "A very long wooooooooooooooooord. and something";
$nuevo_texto = wordwrap($texto, 8, "\n", false);

echo "$nuevo_texto\n";
?>
```

El resultado del ejemplo sería:

```
A very
long
wooooooooooooooooooord.
and
something
```

Ver también ¶

- nl2br() - Inserta saltos de línea HTML antes de todas las nuevas líneas de un string
- chunk_split() - Divide una cadena en trozos más pequeños

+ add a note

User Contributed Notes 23 notes

up
down
14

Alhadis ¶
7 years ago

For those interested in wrapping text to fit a width in *pixels* (instead of characters), you might find the following function useful; particularly for line-wrapping text over dynamically-generated images.

If a word is too long to squeeze into the available space, it'll hyphenate it as needed so it fits the container. This operates recursively, so ridiculously long words or names (e.g., URLs or this guy's signature - http://en.wikipedia.org/wiki/Wolfe+585,_Senior) will still keep getting broken off after they've passed the fourth or fifth lines, or whatever.

```
<?php

    /**
     * Wraps a string to a given number of pixels.
     *
     * This function operates in a similar fashion as PHP's native wordwrap function; however,
     * it calculates wrapping based on font and point-size, rather than character count. This
     * can generate more even wrapping for sentences with a consider number of thin characters.
     *
     * @static $mult;
     * @param string $text - Input string.
     * @param float $width - Width, in pixels, of the text's wrapping area.
     * @param float $size - Size of the font, expressed in pixels.
     * @param string $font - Path to the typeface to measure the text with.
     * @return string The original string with line-breaks manually inserted at detected wrapping
points.
     */
    function pixel_word_wrap($text, $width, $size, $font){

        #    Passed a blank value? Bail early.
        if(!$text) return $text;

        #    Check if imagettfbbox is expecting font-size to be declared in points or pixels.
        static $mult;
        $mult    =    $mult ?: version_compare(GD_VERSION, '2.0', '>=') ? .75 : 1;

        #    Text already fits the designated space without wrapping.
        $box    =    imagettfbbox($size * $mult, 0, $font, $text);
        if($box[2] - $box[0] / $mult < $width)    return $text;

        #    Start measuring each line of our input and inject line-breaks when overflow's
detected.
        $output        =    '';
        $length        =    0;

        $words        =    preg_split('/\b(?=\S)|(?=\s)/', $text);
        $word_count    =    count($words);
        for($i = 0; $i < $word_count; ++$i){

            #    Newline
            if(PHP_EOL === $words[$i])
                $length    =    0;

            #    Strip any leading tabs.
            if(!$length) $words[$i]    =    preg_replace('/^\t+/', '', $words[$i]);

            $box    =    imagettfbbox($size * $mult, 0, $font, $words[$i]);
```

```
            $m        =    $box[2] - $box[0] / $mult;

            #    This is one honkin' long word, so try to hyphenate it.
            if(($diff = $width - $m) <= 0){
                $diff    =    abs($diff);

                #    Figure out which end of the word to start measuring from. Saves a few extra
cycles in an already heavy-duty function.
                if($diff - $width <= 0)    for($s = strlen($words[$i]); $s; --$s){
                    $box    =    imagettfbbox($size * $mult, 0, $font, substr($words[$i], 0, $s) .
'-');

                    if($width > ($box[2] - $box[0] / $mult) + $size){
                        $breakpoint    =    $s;
                        break;
                    }
                }

                else{
                    $word_length    =    strlen($words[$i]);
                    for($s = 0; $s < $word_length; ++$s){
                        $box    =    imagettfbbox($size * $mult, 0, $font, substr($words[$i], 0,
$s+1) . '-');

                        if($width < ($box[2] - $box[0] / $mult) + $size){
                            $breakpoint    =    $s;
                            break;
                        }
                    }
                }

                if($breakpoint){
                    $w_l    =    substr($words[$i], 0, $s+1) . '-';
                    $w_r    =    substr($words[$i],     $s+1);

                    $words[$i]    =    $w_l;
                    array_splice($words, $i+1, 0, $w_r);
                    ++$word_count;
                    $box    =    imagettfbbox($size * $mult, 0, $font, $w_l);
                    $m        =    $box[2] - $box[0] / $mult;
                }
            }

            #    If there's no more room on the current line to fit the next word, start a new
line.
            if($length > 0 && $length + $m >= $width){
                $output    .=    PHP_EOL;
                $length    =    0;

                #    If the current word is just a space, don't bother. Skip (saves a weird-
looking gap in the text).
                if(' ' === $words[$i]) continue;
            }

            #    Write another word and increase the total length of the current line.
            $output    .=    $words[$i];
            $length +=    $m;
        }

        return $output;
```

```
    };

?>
```

up
down
16

julius ¶
10 years ago

Another solution to utf-8 safe wordwrap, unsing regular expressions.
Pretty good performance and works in linear time.

```
<?php
function utf8_wordwrap($string, $width=75, $break="\n", $cut=false)
{
  if($cut) {
    // Match anything 1 to $width chars long followed by whitespace or EOS,
    // otherwise match anything $width chars long
    $search = '/(.{1,'.$width.'})(?:\s|$)|(.{'.$width.'})/uS';
    $replace = '$1$2'.$break;
  } else {
    // Anchor the beginning of the pattern with a lookahead
    // to avoid crazy backtracking when words are longer than $width
    $pattern = '/(?=\s)(.{1,'.$width.'})(?:\s|$)/uS';
    $replace = '$1'.$break;
  }
  return preg_replace($search, $replace, $string);
}
?>
```

Of course don't forget to use preg_quote on the $width and $break parameters if they come from untrusted input.

up
down
13

Dave Lozier - dave at fusionbb.com ¶
16 years ago

If you'd like to break long strings of text but avoid breaking html you may find this useful. It seems to be working for me, hope it works for you. Enjoy. :)

```
<?php
    function textWrap($text) {
        $new_text = '';
        $text_1 = explode('>',$text);
        $sizeof = sizeof($text_1);
        for ($i=0; $i<$sizeof; ++$i) {
            $text_2 = explode('<',$text_1[$i]);
            if (!empty($text_2[0])) {
                $new_text .= preg_replace('#([^\n\r .]{25})#i', '\\1  ', $text_2[0]);
            }
            if (!empty($text_2[1])) {
                $new_text .= '<' . $text_2[1] . '>';
            }
        }
        return $new_text;
    }
?>
```

up
down
3

frans-jan at van-steenbeek dot R-E-M-O-V-E dot net ¶
17 years ago
Using wordwrap is usefull for formatting email-messages, but it has a disadvantage: line-breaks are often treated as whitespaces, resulting in odd behaviour including lines wrapped after just one word.

To work around it I use this:

```
<?php
function linewrap($string, $width, $break, $cut) {
  $array = explode("\n", $string);
  $string = "";
  foreach($array as $key => $val) {
   $string .= wordwrap($val, $width, $break, $cut);
   $string .= "\n";
  }
  return $string;
}
?>
```

I then use linewrap() instead of wordwrap()

hope this helps someone

up
down
2

michdingpayc ¶
4 months ago
A correction to ju1ius' utf-8 safe wordwrap from 10 years ago.
This version addresses issues where breaks were not being added to the first and last words in the input string.

```
<?php
function utf8_wordwrap($string, $width=75, $break="\n", $cut=false)
{
  if($cut) {
    // Match anything 1 to $width chars long followed by whitespace,
    // otherwise match anything $width chars long
    $search= '/(.{1,'.$width.'})(?:\s)|(.{'.$width.'})(?!$)/uS';
    $replace = '$1$2'.$break;
  } else {
    // Anchor the beginning of the pattern with a lookbehind
    // to avoid crazy backtracking when words are longer than $width
    $search= '/(?<=\s|^)(.{1,'.$width.'}\S*)(?:\s)/uS';
    $replace = '$1'.$break;
  }
  return preg_replace($search, $replace, $string);
}
?>
```

up
down
0

altin_bardhi at yahoo dot co dot uk ¶
11 years ago
Here I have come out with a possibly very useful wordwrap code snippet.

Apparently what this piece of code does is: it takes the entered text and looks for words longer than the defined ‘$chunk_length’ if it finds any, it splits the long words and then it

```
concatenates the whole string back to a new string with longer words separated by a dash character
in this case.

After it has accomplished this task it then inserts an HTML line break after a specified
‘$line_length’ (Depending on your containers width requirements)

<?php

//Start function explode_ wrap
function explode_wrap($text, $chunk_length, $line_length){

//Explode all the words separated by spaces in a string
$string_chunks = explode(' ', $text);

// Get each split word from the array $sring_chunks_array => key => value
foreach ($string_chunks as $chunk => $value) {

if(strlen($value) >= $chunk_length){

//Split the chunks/words which are longer than $chunk_length
$new_string_chunks[$chunk] = chunk_split($value, $chunk_length, ' - ');

}else {

//Do not split the normal length words
$new_string_chunks[$chunk] = $value;

}

}//End foreach loop

//Concatenate back the all the words
$new_text=implode(' ', $new_string_chunks);

return wordwrap($new_text, $line_length, '<br />');

}//End function

?>
```

up
down
0
Peter ¶
15 years ago

```
The main concern when you have a text in a cell is for long words that drags the cell margins.
This function will break words in a text that have more then $nr characters using the "-" char.

<?php
function processtext($text,$nr=10)
    {
        $mytext=explode(" ",trim($text));
        $newtext=array();
        foreach($mytext as $k=>$txt)
        {
            if (strlen($txt)>$nr)
            {
                $txt=wordwrap($txt, $nr, "-", 1);
            }
```

```
            $newtext[]=$txt;
        }
        return implode(" ",$newtext);
    }
?>
```

up
down
-1
$del=' at '; 'sanneschaap' dot $del dot 'gmail dot com' ¶
14 years ago

These functions let you wrap strings comparing to their actual displaying width of proportional font. In this case Arial, 11px. Very handy in some cases since CSS3 is not yet completely supported. 100 strings = ~5 ms

My old sheep word wrap function (posted at the bottom of this page, is kinda old dated and this one is faster and more accurate).

```
<?php
//the width of the biggest char @
$fontwidth = 11;

//each chargroup has char-ords that have the same proportional displaying width
$chargroup[0] = array(64);
$chargroup[1] = array(37,87,119);
$chargroup[2] = array(65,71,77,79,81,86,89,109);
$chargroup[3] = array(38,66,67,68,72,75,78,82,83,85,88,90);
$chargroup[4] = array(35,36,43,48,49,50,51,52,53,54,55,56,57,60,61,62,63,
69,70,76,80,84,95,97,98,99,100,101,103,104,110,111,112, 113,115,117,118,120,121,122,126);
$chargroup[5] = array(74,94,107);
$chargroup[6] = array(34,40,41,42,45,96,102,114,123,125);
$chargroup[7] = array(44,46,47,58,59,91,92,93,116);
$chargroup[8] = array(33,39,73,105,106,108,124);

//how the displaying width are compared to the biggest char width
$chargroup_relwidth[0] = 1; //is char @
$chargroup_relwidth[1] = 0.909413854;
$chargroup_relwidth[2] = 0.728241563;
$chargroup_relwidth[3] = 0.637655417;
$chargroup_relwidth[4] = 0.547069272;
$chargroup_relwidth[5] = 0.456483126;
$chargroup_relwidth[6] = 0.36589698;
$chargroup_relwidth[7] = 0.275310835;
$chargroup_relwidth[8] = 0.184724689;

//build fast array
$char_relwidth = null;
for ($i=0;$i<count($chargroup);$i++){
    for ($j=0;$j<count($chargroup[$i]);$j++){
        $char_relwidth[$chargroup[$i][$j]] = $chargroup_relwidth[$i];
    }
}

//get the display width (in pixels) of a string
function get_str_width($str){
    global $fontwidth,$char_relwidth;
    $result = 0;
    for ($i=0;$i<strlen($str);$i++){
        $result += $char_relwidth[ord($str[$i])];
```

```
    }
    $result = $result * $fontwidth;
    return $result;
}

//truncates a string at a certain displaying pixel width
function truncate_str_at_width($str, $width, $trunstr='...'){
    global $fontwidth,$char_relwidth;
    $trunstr_width = get_str_width($trunstr);
    $width -= $trunstr_width;
    $width = $width/$fontwidth;
    $w = 0;
    for ($i=0;$i<strlen($str);$i++){
        $w += $char_relwidth[ord($str[$i])];
        if ($w > $width)
            break;
    }
    $result = substr($str,0,$i).$trunstr;
    return $result;
    // texas is the reason rules at 10am :)
}
?>
```

up
down
-1

php at maranelda dot org ¶
14 years ago

Anyone attempting to write a text email client should be aware of the following:

```
<?php

$a = "some text that must wrap nice";

$a = wordwrap($a, 9);

echo $a;

//  some text
//  that must
//  wrap nice

$a = wordwrap($a, 9);

echo $a;

//  some text
//  that
//  must
//  wrap
//  nice

?>
```

Subsequent uses of wordwrap() on already wrapped text will take the end-of-line characters into account when working out line length, thus reading each line that just fit nicely the first time around as being one character too long the second. This can be a problem when preparing a text email that contains (eg.) a forwarded email which has already been word-wrapped.

Solutions below which explode() the text on end-of-lines and wordwrap() the resulting strings separately take care of this nicely.

up
down
-2

info at hsdn dot org ¶

11 years ago

Wordwrap with UTF-8 supports, returns as array.

```
<?php

function mb_wordwrap_array($string, $width)
{
    if (($len = mb_strlen($string, 'UTF-8')) <= $width)
    {
        return array($string);
    }

    $return = array();
    $last_space = FALSE;
    $i = 0;

    do
    {
        if (mb_substr($string, $i, 1, 'UTF-8') == ' ')
        {
            $last_space = $i;
        }

        if ($i > $width)
        {
            $last_space = ($last_space == 0) ? $width : $last_space;

            $return[] = trim(mb_substr($string, 0, $last_space, 'UTF-8'));
            $string = mb_substr($string, $last_space, $len, 'UTF-8');
            $len = mb_strlen($string, 'UTF-8');
            $i = 0;
        }

        $i++;
    }
    while ($i < $len);

    $return[] = trim($string);

    return $return;
}

?>
```

up
down
-2

Marcin Dobruk [zuku3000 at yahoo dot co dot uk] ¶

13 years ago

Word wrap from left to right (standard) and from right to left.

```
<?php
function myWordWrap ($string, $length=3, $wrap=',', $from='left') {
```

```
    if ($from=='left') $txt=wordwrap($string, $length, $wrap, true);
    if ($from=='right') {
        // string to array
        $arr_l=array();
        for ($a=0;strlen($string)>$a;$a++) $arr_l[$a]=$string{$a};
        // reverse array
        $arr_r=array_reverse($arr_l);
        // array to string
        $string_r='';
        foreach ($arr_r as $arr_line => $arr) $string_r.=$arr;
        // add wrap to reverse string
        $string_r=wordwrap($string_r, $length, $wrap, true);
        // reverse string to array
        $arr_r=array();
        for ($a=0;strlen($string_r)>$a;$a++) $arr_r[]=$string_r{$a};
        // reverse array again
        $arr_l=array_reverse($arr_r);
        // string with wrap
        $txt='';
        foreach ($arr_l as $arr_line => $arr) $txt.=$arr;
        }
    return $txt;
    }
?>
```

up
down
-2

ojs-hp at web dot de ¶

13 years ago

After I got some problems with my function to convert a BB-text into HTML. Long words didn't really fit into the layout and only wordwarp() also added breaks to words which would fit into the layout or destroy the other HTML-tags....
So this is my solution. Only words with strlen() >= 40 are edited with wordwarp().

```
<?php
function bb2html($bb) {
        $words= explode(' ', $bb); // string to array
    foreach ($words as $word) {
        $break = 0;
        for ($i = 0; $i < strlen($word); $i++) {
            if ($break >= 40) {
                $word= wordwrap($word, 40, '-<br>', true); //add <br> every 40 chars
                $break = 0;
            }
            $break++;

        }
        $newText[] = $word; //add word to array
    }
    $bb = implode(' ', $newText); //array to string
    return $bb;
}
?>
```

up
down
-3

maikuolan at gmail dot com ¶

9 years ago

(Re: kouber at php dot net).

Testing out your function, I can confirm that it works, and it works very well.

However, others that intend to use your function need to be aware that if they use it in conjunction with unverified data (such as raw user input from $_POST, $_GET, etcetera), they are creating potential attack vectors that can be exploited by hackers via script requests containing malicious code. This is because your function is using the preg_replace function in conjunction with the "e" flag (in order to allow the chunk_split bit to execute), which can allow execution of arbitrary code.

Solution: If there is any possibility that $str may contain unverified data (such as raw user input), ensure that the contents of $str is sanitized (such as by using htmlentities/htmlspecialchars/etc) prior to sending it to wrap($str,...).

Not a criticism; I intend to use your function, because I like it. However, just posting this as a note to other users that may not be aware of the importance of data sanitation.

up
down
-2

kozimbek at mail dot ru ¶

7 years ago

After searching and being tired of many non-working mb_wordwrap functions at many places, I finally created a really simple and working solution

```
<?php
function mb_wordwrap($string, $limit)
{
    $string = strip_tags($string); //Strip HTML tags off the text
    $string = html_entity_decode($string); //Convert HTML special chars into normal text
    $string = str_replace(array("\r", "\n"), "", $string); //Also cut line breaks
    if(mb_strlen($string, "UTF-8") <= $limit) return $string; //If input string's length is no more than cut length, return untouched
    $last_space = mb_strrpos(mb_substr($string, 0, $limit, "UTF-8"), " ", 0, "UTF-8"); //Find the last space symbol position

    return mb_substr($string, 0, $last_space, "UTF-8").' ...'; //Return the string's length substracted till the last space and add three points
}
?>
```

The function simply searches the last space symbol in the range and returns the string cut till that position. No iterations, no regular expressions and no buffer overload. Tested with large Russian texts and works perfectly.

up
down
-2

phil_marmotte at yahoo dot fr ¶

8 years ago

Another Word wrap from left or right :

```
public static function myWordWrap ($string, $length=3, $wrap=',', $from='left') {
        if ($from=='left') $txt=wordwrap($string, $length, $wrap, true);
            if ($from=='right') {
                $m = strlen($string)%$length;
                if ($m < strlen($string))
                    $txt = substr($string,0,$m).$wrap.wordwrap(substr($string,$m),$length, $wrap, true);
```

```
                else
                    $txt = $string;
            }

            return $txt;
        }
```

up
down
-2
joachim ¶
14 years ago
There seems to be a difference between php 5.1 and 5.2 in how wordwrap counts characters (all on Mac OSX 10.5.2):

```
/Applications/MAMP/bin/php5/bin/php --version
PHP 5.1.6 (cli) (built: Sep  8 2006 10:25:04)

/Applications/MAMP/bin/php5/bin/php -r 'echo wordwrap("In aller Freundschaft (50)_UT", 20) .
"\n";'
In aller
Freundschaft
(50)_UT

php --version
PHP 5.2.5 (cli) (built: Feb 20 2008 12:30:47)

php -r 'echo wordwrap("In aller Freundschaft (50)_UT", 20) . "\n";'
In aller
Freundschaft (50)_UT
```

up
down
-1
answers at clearcrescendo.com ¶
3 years ago
wordwrap() uses the break string as the line break detected, and the break inserted, so your text must be standardized to the line break you want in the wordwrap() output before using wordwrap, otherwise you will get line breaks inserted regardless of the location of existing line breaks in your text.

```
<?php
    $linebreak = '<br/>' . PHP_EOL;
    $width = 5;
    $standardized = preg_replace('/\r?\n/',$linebreak, "abc abc abc\nabc abc abc\r\nabc abc abc");
    echo 'Standardized EOL:', PHP_EOL, $standardized, PHP_EOL, PHP_EOL; // PHP_EOL for the command
line, use '<br/>' for HTML.
    echo "Wrapped at $width:", PHP_EOL, wordwrap( $standardized, 7, $linebreak), PHP_EOL;
?>

$ php -f test.php
Standardized EOL:
abc abc abc<br/>
abc abc abc<br/>
abc abc abc

Wrapped at 5:
abc abc<br/>
abc<br/>
abc abc<br/>
```

```
abc<br/>
abc abc<br/>
abc
```

up
down
-1
tuxedobob ¶
5 years ago

It should be noted that the behavior of the $break parameter is poorly explained.

If you specify the $break parameter, then *that string defines what the function considers a "newline"*.

Consider the following string:

```
$str = "Rumplestiltskin Schwartzmenikoff
1534 Gingerbread Lane
Black Forest, Germany";
```

You are trying to fit this address into a space that only allows for 22 characters, but you want it clear that you're continuing a previous line, so you want a space added. You might try this:

```
$str = wordwrap($str, 22, "\n>");
```

If you did that, you would end up with the following output:

```
"Rumplestiltskin
>Schwartzmenikoff
1534
>Gingerbread Lane
Black
>Forest, Germany"
```

This is because when you pass it a third parameter of "\n>", it assumes that entire string is a newline character. It's no longer using "\n". In your output, of course, \n is still a newline, so it appears to have extra lines.

If you're looking to wordwrap a multi-line string with something besides a newline character, make sure all existing linebreaks are already delineated with the string you pass to wordwrap().

up
down
-2
zac dot hester at gmail dot com ¶
7 years ago

I recently ran into the issue discussed by another contributor to this function (frans-jan at van-steenbeek dot R-E-M-O-V-E dot net). The problem appeared to be how wordwrap() was treating white space. Instead of writing my own version of wordwrap(), I discovered that the "break" parameter is not only used as the inserted string, but also used to detect the existing wrap delimiters (e.g. line endings). If you can manage to "normalize" the wrap delimiters in your original string, you don't need to try to work-around the function wrapping at seemingly odd places (like immediately after one short word). As one quick-and-dirty way to get wordwrap() to play nicely with most use-cases, I did this:

```
<?php
$break = strpos( $content, "\r" ) === false ? "\n" : "\r\n";
$content = wordwrap( $content, 78, $break );
?>
```

I also tend to normalize multi-line strings (if my OCD is acting up). You would typically perform this conversion _before_ sending it off to wordwrap().

```
<?php
//quick and simple, but clobbers old-style Mac line-endings
$content = str_replace( "\r", '', $content );

//slower, but works with everything
$content = preg_replace( "/(\r\n|\r)/", "\n", $content );

//now, wordwrap() will behave exactly as expected
$content = wordwrap( $content, 78, "\n" );
?>
```

up
down
-4

bruceboughton @ google mail ¶

17 years ago

I found that wordwrap deletes the spaces it wraps on. If you want to break up a string which doesn't consist of words, you may find this behaviour undesirable, as I did when trying to wordwrap a Regular Expression to 80 characters (for display along with test string, matches, etc.).

To preserve the spaces and still achieve a consistent cut length, you need to replace spaces with a suitable one-character replacement. I chose the ASCII non-printing character SUB (ASCII #26; some old telephone code meaning substitute):

```
<?php
$regex= str_replace(' ', chr(26), $regex);
$regex= wordwrap($regex, 80, '<br />', TRUE);
$regex= str_replace(chr(26), ' ', $regex);
?>
```

(Of course, you need to replace 80 with your column length and '
' with your break string)

up
down
-7

nbenitezl[arroba]gmail[dot]com ¶

11 years ago

Hi, this function is like wordwrap but it ignores html tags, it works like wordwrap when called with fourth parameter as true. It's based on a function I find here but improved to closer match the output of wordwrap (i.e. removed spaces at start of line) and also to improve performance.

Hope it can be useful for you :-)

```
<?php
function htmlwrap(&$str, $maxLength, $char='<br />'){
    $count = 0;
    $newStr = '';
    $openTag = false;
    $lenstr = strlen($str);
    for($i=0; $i<$lenstr; $i++){
        $newStr .= $str{$i};
        if($str{$i} == '<'){
            $openTag = true;
            continue;
        }
        if(($openTag) && ($str{$i} == '>')){
            $openTag = false;
```

```
            continue;
        }
        if(!$openTag){
            if($str{$i} == ' '){
                if ($count == 0) {
                    $newStr = substr($newStr,0, -1);
                    continue;
                } else {
                    $lastspace = $count + 1;
                }
            }
            $count++;
            if($count==$maxLength){
                if ($str{$i+1} != ' ' && $lastspace && ($lastspace < $count)) {
                    $tmp = ($count - $lastspace)* -1;
                    $newStr = substr($newStr,0, $tmp) . $char . substr($newStr,$tmp);
                    $count = $tmp * -1;
                } else {
                    $newStr .= $char;
                    $count = 0;
                }
                $lastspace = 0;
            }
        }
    }

    return $newStr;
}
?>
```

up
down
-13

Matt at newbiewebdevelopment dot idk ¶

13 years ago

My version of multibyte wordwrap

```
<?php
function mb_wordwrap($string, $width=75, $break="\n", $cut = false) {
    if (!$cut) {
        $regexp = '#^(?:[\x00-\x7F]|[\xC0-\xFF][\x80-\xBF]+){'.$width.',}\b#U';
    } else {
        $regexp = '#^(?:[\x00-\x7F]|[\xC0-\xFF][\x80-\xBF]+){'.$width.'}#';
    }
    $string_length = mb_strlen($string,'UTF-8');
    $cut_length = ceil($string_length / $width);
    $i = 1;
    $return = '';
    while ($i < $cut_length) {
        preg_match($regexp, $string,$matches);
        $new_string = $matches[0];
        $return .= $new_string.$break;
        $string = substr($string, strlen($new_string));
        $i++;
    }
    return $return.$string;
}
```

```
$mb_string = "こんにちは";//Hello in Japanese
$cut_mb_string = mb_wordwrap($mb_string,1," ",true); //こ ん に ち は
print($cut_mb_string);
?>
```

up
down
-3
simbiat at bk dot ru ¶
6 years ago

Some people use wordwrap as long text cutter. I have a slightly different approach for that. The function does not wrap the text, but returns the position where to cut. It preserves HTML tags (does not count them) and strips br tags and whitespaces at the end. I am looking into a way prevent HTML tags corruption which can happen at this time and also support BB tags like code and quote, so that they won't be counted as well. When I do, I'll post an updated version of the code.

```
<?php
function txtcut($string, $length) {
    $tag = false;
    $chars = 0;
    $position = 0;
    $split = str_split($string);
    foreach ($split as $char) {
        $position++;
        if ($char == "<") {
            $tag = true;
            continue;
        }
        if ($char == ">") {
            $tag = false;
            continue;
        }
        if ($tag == true) {
            continue;
        }
        $chars++;
        if ($chars >= $length) {
            if ($char == " " || $char == "\r" || $char == "\n") {
                $position--;
                break;
            }
        }
    }
    if ($position == strlen($string)) {
        return strlen($string);
    } else {
        $string = substr($string, 0, $position);
        if (substr($string, -4, 4) == "<br>") {
            $string = rtrim(substr($string, 0, -4));
        }
        return strlen($string);
    }
}
?>
```

+ add a note

- Funciones de strings
  - addcslashes

- addslashes
- bin2hex
- chop
- chr
- chunk_split
- convert_uudecode
- convert_uuencode
- count_chars
- crc32
- crypt
- echo
- explode
- fprintf
- get_html_translation_table
- hebrev
- hex2bin
- html_entity_decode
- htmlentities
- htmlspecialchars_decode
- htmlspecialchars
- implode
- join
- lcfirst
- levenshtein
- localeconv
- ltrim
- md5_file
- md5
- metaphone
- money_format
- nl_langinfo
- nl2br
- number_format
- ord
- parse_str
- print
- printf
- quoted_printable_decode
- quoted_printable_encode
- quotemeta
- rtrim
- setlocale
- sha1_file
- sha1
- similar_text
- soundex
- sprintf
- sscanf
- str_contains
- str_ends_with
- str_getcsv
- str_ireplace
- str_pad
- str_repeat
- str_replace
- str_rot13
- str_shuffle
- str_split