Funciones de strings
addcslashes
addslashes
bin2hex
convert_ uuencode
count_ chars
rtrim
setlocale
similar_ text
soundex
sprintf

substr

Change language: Spanish

Edit Report a Bug

(PHP 4, PHP 5)

Descripción ¶ ¶

```
string substr ( string $string , int $start [, int $length ] )
```

Devuelve una parte del `string` definida por los parámetros `start` y `length`.

Feedback & Ideas

Parámetros ¶ ¶

`string`
La cadena de entrada. Debe ser de almenos de un caracter.

`start`
Si `start` no es negativo, la cadena devuelta comenzará en el `start` de la posición del `string` empezando desde cero. Por ejemplo, en la cadena '***abcdef***', el caracter en la posición ***0*** es '***a***', el caracter en la posición ***2*** es '***c***', y así sucesivamente.
Si `start` es negativo, la cadena devuelta empezará en `start` contando desde el final de `string`.
Si la longitud del `string` es menor o igual a `start`, la función devolverá FALSE.

Ejemplo #1 Usando un `start` negativo

```
<?php
$rest = substr("abcdef", -1);    // devuelve "f"
$rest = substr("abcdef", -2);    // devuelve "ef"
$rest = substr("abcdef", -3, 1); // devuelve "d"
?>
```

`length`
Si se especifica el `length` y es positivo, la cadena devuelta contendrá como máximo de caracteres de la cantidada dada por `length` que comienza en `start` (dependiedo de la longitud del `string`).
Si se especifica `length` es negativo, entonces ese número de caracteres se omiten al final del `string` (después de la posición inicial se ha calculado a `start` es negativo). Si `start` indica la posición de su truncamiento o más allá, se devolverá false.
Si se omite el `length`, la subcadena empezará por `start` hasta el final de la cadena donde será devuelta.
Si se especifica `length` y es ***0***, FALSE o NULL devolverá una cadena vacía.

Ejemplo #2 Usando un `length` negativo

```
<?php
$rest = substr("abcdef", 0, -1);  // devuelve "abcde"
$rest = substr("abcdef", 2, -1);  // devuelve "cde"
$rest = substr("abcdef", 4, -4);  // devuelve false
$rest = substr("abcdef", -3, -1); // devuelve "de"
?>
```

Valores devueltos ¶ ¶

Devuelve la parte extraída del string, o FALSE en caso de error o un `string` vacío.

Historial de cambios ¶ ¶

Versión	Descripción
5.2.2 - 5.2.6	Si el parámetro **start** indica una posición negativa de truncamiento o más allá, se devolverá false. E obtienen la cadena desde el principio.

str_replace

Ejemplos ¶ ¶

Ejemplo #3 Uso básico de substr()

```
<?php
echo substr('abcdef', 1);     // bcdef
echo substr('abcdef', 1, 3);  // bcd
echo substr('abcdef', 0, 4);  // abcd
echo substr('abcdef', 0, 8);  // abcdef
echo substr('abcdef', -1, 1); // f

// El acceso a caracteres específicos en una cadena
// se puede conseguir usando "corchetes"
$string = 'abcdef';
echo $string[0];                 // a
echo $string[3];                 // d
echo $string[strlen($string)-1]; // f

?>
```

Ejemplo #4 Comportamiento de casting de substr()

```
<?php
class apple {
    public function __toString() {
        return "green";
    }
}

echo "1) ".var_export(substr("pear", 0, 2), true).PHP_EOL;
echo "2) ".var_export(substr(54321, 0, 2), true).PHP_EOL;
echo "3) ".var_export(substr(new apple(), 0, 2), true).PHP_EOL;
echo "4) ".var_export(substr(true, 0, 1), true).PHP_EOL;
echo "5) ".var_export(substr(false, 0, 1), true).PHP_EOL;
echo "6) ".var_export(substr("", 0, 1), true).PHP_EOL;
echo "7) ".var_export(substr(1.2e3, 0, 4), true).PHP_EOL;
?>
```

El resultado del ejemplo sería:

```
1) 'pe'
2) '54'
3) 'gr'
4) '1'
5) false
6) false
7) '1200'
```

Errores/Excepciones ¶ ¶

Devuelve `FALSE` en caso de error.

```
<?php
var_dump(substr('a', 1)); // bool(false)
?>
```

Ver también ¶ ¶

- strrchr() - Encuentra la última aparición de un caracter en un string
- substr_replace() - Reemplaza el texto dentro de una porción de un string
- preg_match() - Realiza una comparación con una expresión regular
- trim() - Elimina espacio en blanco (u otro tipo de caracteres) del inicio y el final de la cadena
- mb_substr() - Obtiene parte de una cadena de caracteres

- wordwrap() - Ajusta un string hasta un número dado de caracteres
- Acceso a cadenas y modificación por caracter

User Contributed Notes 58 notes

add a note

5 Bradley from California ¶ *7 years ago*

```
Add on to (a function originally written by) "Matias from Argentina": str_format_number function.

Just added handling of $String shorter then $Format by adding a side to start the fill and a string length to
the while loop.

<?php
function str_format_number($String, $Format, $Start = 'left'){
    //If we want to fill from right to left incase string is shorter then format
    if ($Start == 'right') {
        $String = strrev($String);
        $Format = strrev($Format);
    }
    if($Format == '') return $String;
    if($String == '') return $String;
    $Result = '';
    $FormatPos = 0;
    $StringPos = 0;
    while ((strlen($Format) - 1) >= $FormatPos && strlen($String) > $StringPos) {
        //If its a number => stores it
        if (is_numeric(substr($Format, $FormatPos, 1))) {
            $Result .= substr($String, $StringPos, 1);
            $StringPos++;
            //If it is not a number => stores the caracter
        } else {
            $Result .= substr($Format, $FormatPos, 1);
        }
        //Next caracter at the mask.
        $FormatPos++;
    }
    if ($Start == 'right') $Result = strrev($Result);
    return $Result;
}
?>
```

5 mar dot czapla at gmail dot com ¶ *5 years ago*

```
Here we have gr8 function which simply convert ip address to a number using substr with negative offset.
You can need it if you want to compare some IP addresses converted to a numbers.
For example when using ip2country, or eliminating same range of ip addresses from your website :D

<?php

function ip2no($val)
{
    list($A,$B,$C,$D)    =    explode(".",$val);
    return
        substr("000".$A,-3).
        substr("000".$B,-3).
        substr("000".$C,-3).
        substr("000".$D,-3);
}

$min        =    ip2no("10.11.1.0");
$max        =    ip2no("111.11.1.0");
$visitor    =    ip2no("105.1.20.200");

if($min<$visitor && $visitor<$max)
    {    echo 'Welcome !';    }
else
    {    echo 'Get out of here !';    }

?>
```

4 Andreas Bur (andreas dot buro at gmail dot com) ¶ *4 years ago*

```
For getting a substring of UTF-8 characters, I highly recommend mb_substr
```

```
<?php
        $utf8string = "cakeæøå";

        echo substr($utf8string,0,5);
        // output cake#
        echo mb_substr($utf8string,0,5,'UTF-8');
        //output cakeæ
?>
```

3 Cristianlf ¶ *3 years ago*

```
I needed a function like lpad from oracle, or right from SQL
 then I use this code :

<?php
function right($string,$chars)
{
    $vright = substr($string, strlen($string)-$chars,$chars);
    return $vright;

}

    echo right('0r0j4152',4);
?>

Result:
 4152
------------------------------------------------
This function is really simple, I just wanted to share, maybe helps someone out there.

regards,
```

3 fanfatal at fanfatal dot pl ¶ *8 years ago*

```
Hmm ... this is a script I wrote, whitch is very similar to substr, but it isn't takes html and bbcode for
counting and it takes portion of string and show avoided (html & bbcode) tags too ;]
Specially usefull for show part of serach result included html and bbcode tags

<?php

/**
 * string csubstr ( string string, int start [, int length] )
 *
 * @author FanFataL
 * @param string string
 * @param int start
 * @param [int length]
 * @return string
 */
function csubstr($string, $start, $length=false) {
    $pattern = '/(\[\w+[^\]]*?\]|\[\/\w+\]|<\w+[^>]*?>|<\/\w+>)/i';
    $clean = preg_replace($pattern, chr(1), $string);
    if(!$length)
        $str = substr($clean, $start);
    else {
        $str = substr($clean, $start, $length);
        $str = substr($clean, $start, $length + substr_count($str, chr(1)));
    }
    $pattern = str_replace(chr(1),'(.*?)',preg_quote($str));
    if(preg_match('/'.$pattern.'/is', $string, $matched))
        return $matched[0];
    return $string;
}

?>

Using this is similar to simple substr.

Greatings ;]
...
```

2 steve at unicycle dot co dot nz ¶ *8 years ago*

```
To quickly trim an optional trailing slash off the end of a path name:

if (substr( $path, -1 ) == '/') $path = substr( $path, 0, -1 );
```

▲ 3 ▼ vnonov at gmail dot com / Viktor Nonov ¶ *3 years ago*

```
<?php

//removes string from the end of other

function removeFromEnd($string, $stringToRemove) {
    $stringToRemoveLen = strlen($stringToRemove);
    $stringLen = strlen($string);

    $pos = $stringLen - $stringToRemoveLen;

    $out = substr($string, 0, $pos);

    return $out;
}

$string = 'picture.jpg.jpg';
$string = removeFromEnd($string, '.jpg');
?>
```

▲ 3 ▼ Petez ¶ *6 years ago*

I wanted to work out the fastest way to get the first few characters from a string, so I ran the following experiment to compare substr, direct string access and strstr:

```
<?php
/* substr access */
beginTimer();
for ($i = 0; $i < 1500000; $i++){
    $opening = substr($string,0,11);
    if ($opening == 'Lorem ipsum'){
        true;
    }else{
        false;
    }
}
$endtime1 = endTimer();

/* direct access */
beginTimer();
for ($i = 0; $i < 1500000; $i++){
    if ($string[0] == 'L' && $string[1] == 'o' && $string[2] == 'r' && $string[3] == 'e' && $string[4] == 'm'
&& $string[5] == ' ' && $string[6] == 'i' && $string[7] == 'p' && $string[8] == 's' && $string[9] == 'u' &&
$string[10] == 'm'){
        true;
    }else{
        false;
    }
}
$endtime2 = endTimer();

/* strstr access */
beginTimer();
for ($i = 0; $i < 1500000; $i++){
    $opening = strstr($string,'Lorem ipsum');
    if ($opening == true){
        true;
    }else{
        false;
    }
}
$endtime3 = endTimer();

echo $endtime1."\r\n".$endtime2."\r\n".$endtime3;
?>
```

The string was 6 paragraphs of Lorem Ipsum, and I was trying match the first two words. The experiment was run 3 times and averaged. The results were:

```
(substr) 3.24
(direct access) 11.49
(strstr) 4.96
```

(With standard deviations 0.01, 0.02 and 0.04)

THEREFORE substr is the fastest of the three methods for getting the first few letters of a string.

4 slow at acedsl dot com ¶ *2 years ago*

```
Anyone coming from the Python world will be accustomed to making substrings by using a "slice index" on a
string.  The following function emulates basic Python string slice behavior. (A more elaborate version could
be made to support array input as well as string, and the optional third "step" argument.)

<?php

function py_slice($input, $slice) {
    $arg = explode(':', $slice);
    $start = intval($arg[0]);
    if ($start < 0) {
        $start += strlen($input);
    }
    if (count($arg) === 1) {
        return substr($input, $start, 1);
    }
    if (trim($arg[1]) === '') {
        return substr($input, $start);
    }
    $end = intval($arg[1]);
    if ($end < 0) {
        $end += strlen($input);
    }
    return substr($input, $start, $end - $start);
}

print py_slice('abcdefg', '2') . "\n";
print py_slice('abcdefg', '2:4') . "\n";
print py_slice('abcdefg', '2:') . "\n";
print py_slice('abcdefg', ':4') . "\n";
print py_slice('abcdefg', ':-3') . "\n";
print py_slice('abcdefg', '-3:') . "\n";

?>

The $slice parameter can be a single character index, or a range separated by a colon. The start of the range
is inclusive and the end is exclusive, which may be  counterintuitive. (Eg, py_slice('abcdefg', '2:4') yields
'cd' not 'cde'). A negative range value means to count from the end of the string instead of the beginning.
Both the start and end of the range may be omitted; the start defaults to 0 and the end defaults to the total
length of the input.

The output from the examples:
c
cd
cdefg
abcd
abcd
efg
```

5 pugazhenthi k ¶ *10 months ago*

```
<?Php

### SUB STRING  BY WORD USING substr() and strpos()  #####

### THIS SCRIPT WILL RETURN PART OF STRING  WITHOUT WORD BREAK ###

$description = ‘your description here your description here your description here your description here your
description here your description here your description hereyour description here your description here’  //
your description here .

$no_letter = 30 ;

if(strlen($desctiption) > 30 )
{
    echo substr($description,0,strpos($description,’ ‘,30));            //strpos to find ‘ ‘ after 30
characters.
}
else {
    echo $description;
}

?>
```

2 pheagey at gmail dot com ¶ *1 year ago*

```
Using a 0 as the last parameter for substr().

As per examples
<?php $var = substr($var, 4); ?>

works no problem. However
<?php $var = substr($var, 4, 0); ?>

will get you nothing. Just a quick heads up
```

▲ 2 ▼ frank at jkelloggs dot dk ¶ *8 years ago*

```
Regarding the utf8_substr function from lmak: The pattern '/./u' doesn't match newline characters. This means
that the substring from 0 to the total length of the string will miss the number of characters in the end
matching the number of newlines in the string. To fix this one can add the s modifier (PCRE_DOTALL) in the
pattern:

<?php
function utf8_substr($str,$start)
{
   preg_match_all("/./su", $str, $ar);

   if(func_num_args() >= 3) {
       $end = func_get_arg(2);
       return join("",array_slice($ar[0],$start,$end));
   } else {
       return join("",array_slice($ar[0],$start));
   }
}
?>
```

▲ 1 ▼ nikolai dot wuestemann at t-online dot de ¶ *2 years ago*

```
If you want to have a string BETWEEN two strings, just use this function:

<?php
function get_between($input, $start, $end)
{
  $substr = substr($input, strlen($start)+strpos($input, $start), (strlen($input) - strpos($input,
$end))*(-1));
  return $substr;
}

//Example:

$string = "123456789";
$a = "12";
$b = "9";

echo get_between($string, $a, $b);

//Output:
//345678
?>
```

▲ 1 ▼ sajjad at sajjad dot biz ¶ *4 years ago*

```
Substring utf-8 strings!
very simple!

<?php
function substru($str,$from,$len){
    return preg_replace('#^(?:[\x00-\x7F]|[\xC0-\xFF][\x80-\xBF]+){0,'. $from .'}'.'((?:[\x00-\x7F]|[\xC0-
\xFF][\x80-\xBF]+){0,'. $len .'}).*#s','$1', $str);
}
?>
```

▲ 1 ▼ Anonymous ¶ *5 years ago*

```
Split a string to an array of strings specified by an array of lengths:

<?php
function split_by_lengths($inString, $arrayLengths)
{
    $output = array();
    foreach ($arrayLengths as $oneLength)
    {
        $output[] = substr($inString, 0, $oneLength);
        $inString = substr($inString, $oneLength);
    }
    return ($output);
}
?>
split_by_lengths('teststringtestteststring', array(4,6,4,4,6)) returns:
array('test','string','test','test','string')

Don't use it on user input without some error handling!
```

▲ 1 ▼ Anonymous ¶ *5 years ago*

I've used the between, after, before, etc functions that biohazard put together for years and they work great. I've also added to it a new function that I use a lot and thought others might like it as well. It uses his before/after functions so they are required to use it.

```
<?php
$example_html = "<p>test1 Test2</p><title>hi there</title><p>Testing</p>";
$paragraph_text = multi_between('<p>', '</p>', $example_html);

//Prints an arry of:
//Array ( [1] => test1 Test2 [2] => Testing )
print_r($paragraph_text);

function multi_between($this, $that, $inthat)
{
   $counter = 0;
   while ($inthat)
   {
      $counter++;
      $elements[$counter] = before($that, $inthat);
      $elements[$counter] = after($this, $elements[$counter]);
      $inthat = after($that, $inthat);
   }
   return $elements;
}
//Get the help functions from biohazard's post below.
?>
```

▲ 1 ▼ morgangalpin att gmail dotty com ¶ *6 years ago*

Adding the $limit parameter introduced a bug that was not present in the original. If $limit is small or negative, a string with a length exceeding the limit can be returned. The $limit parameter should be checked. It takes slightly more processing, but it is dwarfed in comparison to the use of strlen().

```
<?php
  function short_name($str, $limit)
  {
    // Make sure a small or negative limit doesn't cause a negative length for substr().
    if ($limit < 3)
    {
      $limit = 3;
    }

    // Now truncate the string if it is over the limit.
    if (strlen($str) > $limit)
    {
      return substr($str, 0, $limit - 3) . '...';
    }
    else
    {
      return $str;
    }
  }
?>
```

▲ 1 ▼ ivanhoe011 at gmail dot com ¶ *8 years ago*

If you need just a single character from the string you don't need to use substr(), just use curly braces notation:

```
<?php
    // both lines will output the 3rd character
    echo substr($my_string, 2, 1);
    echo $my_string{2};
```

```
?>

curly braces syntax is faster and more readable IMHO..
```

1 rob NOSPAM at clancentric dot net ¶ *8 years ago*

```
I have developed a function with a similar outcome to jay's

Checks if the last character is or isnt a space. (does it the normal way if it is)
It explodes the string into an array of seperate works, the effect is... it chops off anything after and
including the last space.

<?php
function limit_string($string, $charlimit)
{
    if(substr($string,$charlimit-1,1) != ' ')
    {
        $string = substr($string,'0',$charlimit);
        $array = explode(' ',$string);
        array_pop($array);
        $new_string = implode(' ',$array);

        return $new_string.'...';
    }
    else
    {
        return substr($string,'0',$charlimit-1).'...';
    }
}
?>
```

1 woutermb at gmail dot com ¶ *8 years ago*

```
Well this is a script I wrote, what it does is chop up long words with malicious meaning into several parts.
This way, a chat in a table will not get stretched anymore.

<?php

function text($string,$limit=20,$chop=10){

$text = explode(" ",$string);
while(list($key, $value) = each($text)){
    $length = strlen($value);
    if($length >=20){
        for($i=0;$i<=$length;$i+=10){
            $new .= substr($value, $i, 10);
            $new .= " ";
        }
         $post .= $new;
    }
    elseif($length <=15){
        $post .= $value;
    }
    $post .= " ";
}
return($post);
}

// for example, this would return:
$output = text("Well this text doesn't get cut up, yet thisssssssssssssssssssssss one does.", 10, 5);

echo($output); // "Well this text doesn't get cup up, yet thiss sssss sssss sssss sssss sss one does."
?>

I hope it was useful.. :)
```

0 nanhe dot kumar at gmail dot com ¶ *2 months ago*

```
<?php
$string="NanheKumar";
echo substr($string,0,5); //Nanhe
echo substr($string,5); //Kumar
echo substr($string,-4); //umar
echo substr($string,-4,2); //um
echo substr($string,2,6); //umnheKum
?>
```

0 fengdingbo at gmail dot com ¶ *4 months ago*

```
<?php
$a = 'abc';
$php = 'ok123';
$str = microtime(TRUE);
for($i=0;$i<=1000000;$i++)
    $a = substr($php,1,1);
echo microtime(true)-$str,"\n";

$str = microtime(TRUE);
for($i=0;$i<=1000000;$i++)
    $a = $php[1];
echo microtime(true)-$str;
?>
output:
0.26694822311401
0.10447096824646
```

0 biohazard dot ge at gmail dot com ¶ *4 months ago*

```
may be by following functions will be easier to extract the needed sub parts from a string:

<?php
after ('@', 'biohazard@online.ge');
//returns 'online.ge'
//from the first occurrence of '@'

before ('@', 'biohazard@online.ge');
//returns 'biohazard'
//from the first occurrence of '@'

between ('@', '.', 'biohazard@online.ge');
//returns 'online'
//from the first occurrence of '@'

after_last ('[', 'sin[90]*cos[180]');
//returns '180]'
//from the last occurrence of '['

before_last ('[', 'sin[90]*cos[180]');
//returns 'sin[90]*cos['
//from the last occurrence of '['

between_last ('[', ']', 'sin[90]*cos[180]');
//returns '180'
//from the last occurrence of '['
?>

here comes the source:

<?php

    function after ($this, $inthat)
    {
        if (!is_bool(strpos($inthat, $this)))
        return substr($inthat, strpos($inthat,$this)+strlen($this));
    };

    function after_last ($this, $inthat)
    {
        if (!is_bool(strrevpos($inthat, $this)))
        return substr($inthat, strrevpos($inthat, $this)+strlen($this));
    };

    function before ($this, $inthat)
    {
        return substr($inthat, 0, strpos($inthat, $this));
    };

    function before_last ($this, $inthat)
    {
        return substr($inthat, 0, strrevpos($inthat, $this));
    };

    function between ($this, $that, $inthat)
    {
        return before ($that, after($this, $inthat));
    };

    function between_last ($this, $that, $inthat)
    {
     return after_last($this, before_last($that, $inthat));
    };

// use strrevpos function in case your php version does not include it
function strrevpos($instr, $needle)
{
    $rev_pos = strpos (strrev($instr), strrev($needle));
    if ($rev_pos===false) return false;
    else return strlen($instr) - $rev_pos - strlen($needle);
};
?>
```

▲ 0 ▼ leon weidauer ¶ *2 years ago*

```
When using a value of a wrong type as second parameter , substr() does not return FALSE but NULL although the
docs say, it should return FALSE on error.

Prior to PHP 5.3, substr() tries to cast the second parameter to int and doesn't throw any errors. Since PHP
5.3 a warning is thrown.
```

▲ 0 ▼ magickey ¶ *3 years ago*

```
Simple UTF-8 Multibyte solution (without mb_substr)

<?php
  $string="texto en español";
  echo substr($string,0,14); //Outputs: texto en espa�
?>

<?php
  $string="texto en español";
  echo utf8_encode(substr(utf8_decode($string),0,14)); //Outputs: texto en españ
?>
```

▲ 0 ▼ joseph dot davidson dot 707 at gmail dot com ¶ *3 years ago*

```
Using substr() to examine characters of a string without altering the string.

<?php
$string = 'This is my string';
$length = strlen($string);
$myChar = 'm';

for($i = 0; $i < $length; $i++) {

    $showString_i = substr($string, $i, 1);
    if($myChar == $showString_i) return $i;
}
?>

can also examine subs.
```

▲ 0 ▼ biner(gf) ¶ *4 years ago*

```
<?php
//substring without words breaking

$str = "aa bb ccc ddd ee fff gg hhh iii";

echo substr(($str=wordwrap($str,$,'$$')),0,strpos($str,'$$'));
?>
```

▲ 0 ▼ kaj dot strom at kapsi dot fi ¶ *4 years ago*

```
One thing to keep in mind when using string indexes and UTF-8 is that string indexes are NOT multi-byte safe.

<?php
$string = 'äää1';
echo $string[3];
?>
```

```
Outputs:
¤

When it logically should output "1". This is not a bug, as PHP 'normal' string functions are not intended to
be multi-byte safe. This can be solved by using this function

<?php
/**
 *
 * @param string $string String to "search" from
 * @param int $index Index of the letter we want.
 * @return string The letter found on $index.
 */
function charAt($string, $index){
    if($index < mb_strlen($string)){
        return mb_substr($string, $index, 1);
    }
    else{
        return -1;
    }
}
?>
```

0 gkhelloworld at gmail dot com ¶ *4 years ago*

```
Shortens the filename and its expansion has seen.

<?php
$file = "Hellothisfilehasmorethan30charactersandthisfayl.exe";

function funclongwords($file)
{
if (strlen($file) > 30)
{
$vartypesf = strrchr($file,".");
$vartypesf_len = strlen($vartypesf);
$word_l_w = substr($file,0,15);
$word_r_w = substr($file,-15);
$word_r_a = substr($word_r_w,0,-$vartypesf_len);

return $word_l_w."...".$word_r_a.$vartypesf;
}
else
return $file;
}
// RETURN: Hellothisfileha...andthisfayl.exe
?>
```

0 link ¶ *4 years ago*

```
I created some functions for entity-safe splitting+lengthcounting:

<?php
function strlen_entities($text)
{
    preg_match_all(
        '/((?:&(?:#[0-9]{2,}|[a-z]{2,});)|(?:[^&])|'.
        '(?:&(?!\w;)))s',$text,$textarray);
    return count($textarray[0]);
}
function substr_entities($text,$start,$limit=0)
{
    $return = '';
    preg_match_all(
        '/((?:&(?:#[0-9]{2,}|[a-z]{2,});)|(?:[^&])|'.
        '(?:&(?!\w;)))s',$text,$textarray);
    $textarray = $textarray[0];
    $numchars = count($textarray)-1;
    if ($start>=$numchars)
        return false;
    if ($start<0)
    {
        $start = ($numchars)+$start+1;
    }
    if ($start>=0)
    {
        if ($limit==0)
        {
            $end=$numchars;
        }
        elseif ($limit>0)
        {
            $end = $start+($limit-1);
        }
        else
        {
            $end = ($numchars)+$limit;
        }

        for ($i=$start;$i<=$end;$i++)
        {
            $return .= $textarray[$i];
        }
        return $return;
    }
}
?>
```

▲ 0 ▼ NULL_byte ¶ *5 years ago*

```
<?php

function insert_substr($str, $pos, $substr) {
    $part1 = substr($str, 0, -$pos);
    $part2 = substr($str, -$pos);
    return $part1.$substr.$part2;
}

?>
```

▲ 0 ▼ post [at] jannik - zappe [dot] de ¶ *5 years ago*

```
Just a little function to cut a string by the wanted amount. Works in both directions.

<?php
function cutString($str, $amount = 1, $dir = "right")
{
  if(($n = strlen($str)) > 0)
  {
    if($dir == "right")
    {
      $start = 0;
      $end = $n-$amount;
    } elseif( $dir == "left") {
      $start = $amount;
      $end = $n;
    }

    return substr($str, $start, $end);
  } else return false;
}
?>

Enjoy ;)
```

▲ 0 ▼ kriskra at gmail dot com ¶ *5 years ago*

```
The javascript charAt equivalent in php of felipe has a little bug. It's necessary to compare the type
(implicit) aswell or the function returns a wrong result:
<?php
function charAt($str,$pos) {
    return (substr($str,$pos,1) !== false) ? substr($str,$pos,1) : -1;
}
?>
```

▲ 0 ▼ highstrike at gmail dot com ¶ *5 years ago*

```
Because i didnt see a function that would cut a phrase from a text (article or whatever) no matter where,
front/middle/end and add ... + keeping the words intact, i wrote this:

Usage:
- The parameter $value if array will need the whole text and the portion you want to start from, a string. EG:
cuttext(array($text, $string), 20). If the string is "have" and is near the beginning of the text, the
function will cut like "I have a car ...", if the string is in the middle somewhere it will cut like "... if
```

```
you want to have your own car ..." and if its somewhere near the end it will cut like "... and you will have
one."
- The $length parameter is self explanatory.

Note: if you have just a string "127hh43h2h52312453jfks2" and you want to cut it, just use the function like
so: cuttext($string, 10) and it will cut it like "127hh43h2h..."

<?php

/////////////////////////////////////////////////////////
// Function:         cuttext
// Description: Cuts a string and adds ...

function cuttext($value, $length)
{
    if(is_array($value)) list($string, $match_to) = $value;
    else { $string = $value; $match_to = $value{0}; }

    $match_start = stristr($string, $match_to);
    $match_compute = strlen($string) - strlen($match_start);

    if (strlen($string) > $length)
    {
        if ($match_compute < ($length - strlen($match_to)))
        {
            $pre_string = substr($string, 0, $length);
            $pos_end = strrpos($pre_string, " ");
            if($pos_end === false) $string = $pre_string."...";
            else $string = substr($pre_string, 0, $pos_end)."...";
        }
        else if ($match_compute > (strlen($string) - ($length - strlen($match_to))))
        {
            $pre_string = substr($string, (strlen($string) - ($length - strlen($match_to))));
            $pos_start = strpos($pre_string, " ");
            $string = "...".substr($pre_string, $pos_start);
            if($pos_start === false) $string = "...".$pre_string;
            else $string = "...".substr($pre_string, $pos_start);
        }
        else
        {
            $pre_string = substr($string, ($match_compute - round(($length / 3))), $length);
            $pos_start = strpos($pre_string, " "); $pos_end = strrpos($pre_string, " ");
            $string = "...".substr($pre_string, $pos_start, $pos_end)."...";
            if($pos_start === false && $pos_end === false) $string = "...".$pre_string."...";
            else $string = "...".substr($pre_string, $pos_start, $pos_end)."...";
        }

        $match_start = stristr($string, $match_to);
        $match_compute = strlen($string) - strlen($match_start);
    }

    return $string;
}

?>
```

0 ijavier aka(not imatech) igjav ¶ *6 years ago*

```
<?php
/*
    An advanced substr but without breaking words in the middle.
    Comes in 3 flavours, one gets up to length chars as a maximum, the other with length chars as a minimum up
to the next word, and the other considers removing final dots, commas and etcteteras for the sake of beauty
(hahaha).
   This functions were posted by me some years ago, in the middle of the ages I had to use them in some
corporations incorporated, with the luck to find them in some php not up to date mirrors. These mirrors are
rarely being more not up to date till the end of the world... Well, may be am I the only person that finds
usef not t bre word in th middl?

Than! (ks)

This is the calling syntax:

    snippet(phrase,[max length],[phrase tail])
    snippetgreedy(phrase,[max length before next space],[phrase tail])

*/
```

```
function snippet($text,$length=64,$tail="...") {
    $text = trim($text);
    $txtl = strlen($text);
    if($txtl > $length) {
        for($i=1;$text[$length-$i]!=" ";$i++) {
            if($i == $length) {
                return substr($text,0,$length) . $tail;
            }
        }
        $text = substr($text,0,$length-$i+1) . $tail;
    }
    return $text;
}

// It behaves greedy, gets length characters ore goes for more

function snippetgreedy($text,$length=64,$tail="...") {
    $text = trim($text);
    if(strlen($text) > $length) {
        for($i=0;$text[$length+$i]!=" ";$i++) {
            if(!$text[$length+$i]) {
                return $text;
            }
        }
        $text = substr($text,0,$length+$i) . $tail;
    }
    return $text;
}

// The same as the snippet but removing latest low punctuation chars,
// if they exist (dots and commas). It performs a later suffixal trim of spaces

function snippetwop($text,$length=64,$tail="...") {
    $text = trim($text);
    $txtl = strlen($text);
    if($txtl > $length) {
        for($i=1;$text[$length-$i]!=" ";$i++) {
            if($i == $length) {
                return substr($text,0,$length) . $tail;
            }
        }
        for(;$text[$length-$i]=="," || $text[$length-$i]=="." || $text[$length-$i]==" ";$i++) {;}
        $text = substr($text,0,$length-$i+1) . $tail;
    }
    return $text;
}

/*
echo(snippet("this is not too long to run on the column on the left, perhaps, or perhaps yes, no idea") .
"<br>");
echo(snippetwop("this is not too long to run on the column on the left, perhaps, or perhaps yes, no idea") .
"<br>");
echo(snippetgreedy("this is not too long to run on the column on the left, perhaps, or perhaps yes, no
idea"));
*/
?>
```

▲ 0 ▼ persisteus at web dot de ¶ *6 years ago*

Here is also a nice (but a bit slow) alternative for colorizing an true color image:

```
<?php
// $colorize = hexadecimal code in String format, f.e. "10ffa2"
// $im = the image that have to be computed

$red = hexdec(substr($colorize, 0, 2));
$green = hexdec(substr($colorize, 2, 2));
$blue = hexdec(substr($colorize, 4, 2));

$lum_c = floor(($red*299 + $green*587 + $blue*144) / 1000);

for ($i = 0; $i < $lum_c; $i++)
{
  $r = $red * $i / $lum_c;
  $g = $green * $i / $lum_c;
  $b = $blue * $i / $lum_c;
  $pal[$i] = $r<<16 | $g<<8 | $b;
}
$pal[$lum_c] = $red<<16 | $green<<8 | $blue;
for ($i = $lum_c+1; $i < 255; $i++)
{
  $r = $red + (255-$red) * ($i-$lum_c) / (255-$lum_c);
  $g = $green + (255-$green) * ($i-$lum_c) / (255-$lum_c);
  $b = $blue + (255-$blue) * ($i-$lum_c) / (255-$lum_c);
  $pal[$i] = $r<<16 | $g<<8 | $b;
}

$sy = imagesy($im);
$sx = imagesx($im);
for($y=0;$y<$sy;$y++)
{
  for($x=0;$x<$sx;$x++)
  {
    $rgba = imagecolorat($im, $x, $y);
    $a = ($rgba & 0x7F000000) >> 24;
    $r = ($rgba & 0xFF0000) >> 16;
    $g = ($rgba & 0x00FF00) >> 8;
    $b = ($rgba & 0x0000FF);

    $lum = floor(($r*299+$g*587+$b*144)/1000);

    imagesetpixel($im, $x, $y, $a<<24 | $pal[$lum]);
  }
}
?>
```

0 egingell at sisna dot com ¶ *7 years ago*

```
<?php

/**
 * string substrpos(string $str, mixed $start [[, mixed $end], boolean $ignore_case])
 *
 * If $start is a string, substrpos will return the string from the position of the first occuring $start to
$end
 *
 * If $end is a string, substrpos will return the string from $start to the position of the first occuring
$end
 *
 * If the first character in (string) $start or (string) $end is '-', the last occuring string will be used.
 *
 * If $ignore_case is true, substrpos will not care about the case.
 * If $ignore_case is false (or anything that is not (boolean) true, the function will be case sensitive.
 *        Both of the above: only applies if either $start or $end are strings.
 *
 * echo substrpos('This is a string with 0123456789 numbers in it.', 5, '5');
 *        // Prints 'is a string with 01234';
 *
 * echo substrpos('This is a string with 0123456789 numbers in it.', '5', 5);
 *        // Prints '56789'
 *
 * echo substrpos('This is a string with 0123456789 numbers in it and two strings.', -60, '-string')
 *        // Prints 's is a string with 0123456789 numbers in it and two '
 *
 * echo substrpos('This is a string with 0123456789 numbers in it and two strings.', -60, '-STRING', true)
 *        // Prints 's is a string with 0123456789 numbers in it and two '
 *
 * echo substrpos('This is a string with 0123456789 numbers in it and two strings.', -60, '-STRING', false)
 *        // Prints 's is a string with 0123456789 numbers in it and two strings.'
 *
 * Warnings:
 *        Since $start and $end both take either a string or an integer:
 *            If the character or string you are searching $str for is a number, pass it as a quoted string.
 *        If $end is (integer) 0, an empty string will be returned.
 *        Since this function takes negative strings ('-search_string'):
 *            If the string your using in $start or $end is a '-' or begins with a '-' escape it with a '\'.
 *            This only applies to the *first* character of $start or $end.
 */

// Define stripos() if not defined (PHP < 5).
if (!is_callable("stripos")) {
    function stripos($str, $needle, $offset = 0) {
        return strpos(strtolower($str), strtolower($needle), $offset);
    }
}

function substrpos($str, $start, $end = false, $ignore_case = false) {
    // Use variable functions
    if ($ignore_case === true) {
        $strpos = 'stripos'; // stripos() is included above in case it's not defined (PHP < 5).
    } else {
        $strpos = 'strpos';
    }

    // If end is false, set it to the length of $str
    if ($end === false) {
        $end = strlen($str);
    }

    // If $start is a string do what's needed to make it an integer position for substr().
    if (is_string($start)) {
        // If $start begins with '-' start processing until there's no more matches and use the last one
found.
        if ($start{0} == '-') {
            // Strip off the '-'
            $start = substr($start, 1);
            $found = false;
            $pos = 0;
            while(($curr_pos = $strpos($str, $start, $pos)) !== false) {
                $found = true;
                $pos = $curr_pos + 1;
            }
            if ($found === false) {
                $pos = false;
            } else {
                $pos -= 1;
            }
        } else {
            // If $start begins with '\-', strip off the '\'.
            if ($start{0} . $start{1} == '\-') {
                $start = substr($start, 1);
            }
            $pos = $strpos($str, $start);
        }
        $start = $pos !== false ? $pos : 0;
    }

    // Chop the string from $start to strlen($str).
    $str = substr($str, $start);

    // If $end is a string, do exactly what was done to $start, above.
    if (is_string($end)) {
        if ($end{0} == '-') {
            $end = substr($end, 1);
            $found = false;
            $pos = 0;
            while(($curr_pos = strpos($str, $end, $pos)) !== false) {
                $found = true;
                $pos = $curr_pos + 1;
            }
            if ($found === false) {
                $pos = false;
            } else {
                $pos -= 1;
            }
        } else {
            if ($end{0} . $end{1} == '\-') {
                $end = substr($end, 1);
            }
            $pos = $strpos($str, $end);
        }
        $end = $pos !== false ? $pos : strlen($str);
    }

    // Since $str has already been chopped at $start, we can pass 0 as the new $start for substr()
    return substr($str, 0, $end);
}

?>
```

0 mr at bbp dot biz ¶ *7 years ago*

Here's a little addon to the html_substr function posted by fox.

Now it counts only chars outside of tags, and doesn't cut words.

Note: this will only work in xhtml strict/transitional due to the checking of "/>" tags and the requirement of quotations in every value of a tag. It's also only been tested with the presence of br, img, and a tags, but it should work with the presence of any tag.

```php
<?php
function html_substr($posttext, $minimum_length = 200, $length_offset = 20, $cut_words = FALSE, $dots = TRUE)
{

    // $minimum_length:
    // The approximate length you want the concatenated text to be


    // $length_offset:
    // The variation in how long the text can be in this example text
    // length will be between 200 and 200-20=180 characters and the
    // character where the last tag ends

    // Reset tag counter & quote checker
    $tag_counter = 0;
    $quotes_on = FALSE;
    // Check if the text is too long
    if (strlen($posttext) > $minimum_length) {
        // Reset the tag_counter and pass through (part of) the entire text
        $c = 0;
        for ($i = 0; $i < strlen($posttext); $i++) {
            // Load the current character and the next one
            // if the string has not arrived at the last character
            $current_char = substr($posttext,$i,1);
            if ($i < strlen($posttext) - 1) {
                $next_char = substr($posttext,$i + 1,1);
            }
            else {
                $next_char = "";
            }
            // First check if quotes are on
            if (!$quotes_on) {
                // Check if it's a tag
                // On a "<" add 3 if it's an opening tag (like <a href...)
                // or add only 1 if it's an ending tag (like </a>)
                if ($current_char == '<') {
                    if ($next_char == '/') {
                        $tag_counter += 1;
                    }
                    else {
                        $tag_counter += 3;
                    }
                }
                // Slash signifies an ending (like </a> or ... />)
                // substract 2
                if ($current_char == '/' && $tag_counter <> 0) $tag_counter -= 2;
                // On a ">" substract 1
                if ($current_char == '>') $tag_counter -= 1;
                // If quotes are encountered, start ignoring the tags
                // (for directory slashes)
                if ($current_char == '"') $quotes_on = TRUE;
            }
            else {
                // IF quotes are encountered again, turn it back off
                if ($current_char == '"') $quotes_on = FALSE;
            }

            // Count only the chars outside html tags
            if($tag_counter == 2 || $tag_counter == 0){
                $c++;
            }

            // Check if the counter has reached the minimum length yet,
            // then wait for the tag_counter to become 0, and chop the string there
            if ($c > $minimum_length - $length_offset && $tag_counter == 0 && ($next_char == ' ' || $cut_words
== TRUE)) {
                $posttext = substr($posttext,0,$i + 1);
                if($dots){
                    $posttext .= '...';
                }
                return $posttext;
            }
        }
    }
    return $posttext;
}

?>
```

▲ 0 ▼ php_net at thomas dot trella dot de ¶ *8 years ago*

```
I needed to cut a string after x chars at a  html converted utf-8 text (for example Japanese text like
&#23344;&#35632;&#24368;&#33072;&#27440;&#32591;).
The problem was, the different length of the signs, so I wrote the following function to handle that.
Perhaps it helps.

<?php

function html_cutstr ($str, $len)
{
    if (!preg_match('/\&#[0-9]*;.*/i', $str))
    {
        $rVal = strlen($str, $len);
        break;
    }

    $chars = 0;
    $start = 0;
    for($i=0; $i < strlen($str); $i++)
    {
        if ($chars >= $len)
        break;

        $str_tmp = substr($str, $start, $i-$start);
        if (preg_match('/\&#[0-9]*;.*/i', $str_tmp))
        {
            $chars++;
            $start = $i;
        }
    }
    $rVal = substr($str, 0, $start);
    if (strlen($str) > $start)
    $rVal .= " ...";
    return $rVal;
}
?>
```

▲ 0 ▼ andrewmclagan at gmail dot com ¶ *8 years ago*

```
Hi there here is a little function i wrote to limit the number of lines in a string, i could not find anything
else like it out there

<?php
function lineLimiter ($string = "", $max_lines = 1) {

         $string = ereg_replace("\n", "##", $string);

         $totalLines = (substr_count($string, '##') + 1);

         $string = strrev($string);

         $stringLength = strlen($string);

         while ($totalLines > $max_lines) {
             $pos = 0;
             $pos = strpos ( $string, "##") + 2;
             //$pos = $pos - $stringLength;
             $string = substr($string, $pos);
             $totalLines--;
         }
         $string = strrev($string);
          $string = ereg_replace("##", "\n", $string);
         return $string;
    }
```

```
?>
```

0 Anonymous ¶ *10 years ago*

```
If you want to substring the middle of a string with another and keep the words intact:

<?php
/**
 * Reduce a string by the middle, keeps whole words together
 *
 * @param string $string
 * @param int $max (default 50)
 * @param string $replacement (default [...])
 * @return string
 * @author david at ethinkn dot com
 * @author loic at xhtml dot ne
 * @author arne dot hartherz at gmx dot net
 */

function strMiddleReduceWordSensitive ($string, $max = 50, $rep = '[...]') {
    $strlen = strlen($string);

    if ($strlen <= $max)
        return $string;

    $lengthtokeep = $max - strlen($rep);
    $start = 0;
    $end = 0;

    if (($lengthtokeep % 2) == 0) {
        $start = $lengthtokeep / 2;
        $end = $start;
    } else {
        $start = intval($lengthtokeep / 2);
        $end = $start + 1;
    }

    $i = $start;
    $tmp_string = $string;
    while ($i < $strlen) {
        if ($tmp_string[$i] == ' ') {
            $tmp_string = substr($tmp_string, 0, $i) . $rep;
            $return = $tmp_string;
        }
        $i++;
    }

    $i = $end;
    $tmp_string = strrev ($string);
    while ($i < $strlen) {
        if ($tmp_string[$i] == ' ') {
            $tmp_string = substr($tmp_string, 0, $i);
            $return .= strrev ($tmp_string);
        }
        $i++;
    }
    return $return;
    return substr($string, 0, $start) . $rep . substr($string, - $end);
}

echo strMiddleReduceWordSensitive ('ABCDEEF GHIJK LLKJHKHKJHKL HGHFK sdfasdfsdafsdf sadf asdf sadf sad s', 30)
. "\n";
// Returns: ABCDEEF GHIJK[...]asdf sadf sad s (33 chrs)
echo strMiddleReduceWordSensitive ('ABCDEEF GHIJK LLKJHKHKJHKL HGHFK sdfasdfsdafsdf sadf asdf sadf sad s', 30,
'...') . "\n";
// Returns: ABCDEEF GHIJK...asdf sadf sad s (32 chrs)
?>
```

-1 maxim at inbox dot ru ¶ *1 year ago*

```
Here is a recursion function to get parts of passed string which are in a char. Func looks pretty, and works
fast, tell me please if you find more opt way.
<?php
$s = "info= &make&,endvcc &new& &another&info";
echo str_cut($s,"&",",");
//output:
//make,new,another
```

```
function str_cut($s,$a,$d="")
{
    $f=strpos($s,$a)+1;
    $l=strpos($s,$a,$f);
    $out= substr($s,$f,$l-$f);
                if
    (strpos($s,$a,$l+1)!==false)
    {$s=substr($s,$l+1);$out.=$d.str_cut($s,$a,$d);}
return $out;
}
?>

it is possible to output in array, you have to returns a var as an array.
Also you can add an extra needle, which would be compare with the end of strings :
replace to
$l=strpos($s,$b,$f)-$f;
and dont forget to pass into recursion call $b value, but i did like that
$b=func_get_arg(func_num_args()-1);
care! the last argument should be $a, because $b getting last arg. like that:
function str_cut($s,$d="",$a)
but i think this method of getting $b not the best way for perfomance.
Sorry for my English.
```

-1 mr.davin ¶ *5 years ago*

```
Simple use of substr to determine possession:

<?php
function possessive ($word) {
    return  $word.(substr($word, -1) == 's' ? "'" : "'s");
}

// Davis => Davis'
// Paul => Paul's
?>
```

-1 jeff dot swain at pcmmllc dot com ¶ *3 years ago*

```
I noticed a slight issue when parsing out long strings using the substr function.

Here is my string: $merge = "UPDATE AssistanceRequest SET RequestorID = '4301' WHERE RequestorID IN (
'4535','6222','4865','5137','4893')"

To parse out the WHERE portion I used:
$whereClause = substr($merge, strpos($merge,'WHERE', (strlen($merge) - strpos($merge,'WHERE'))));
Normally the function returned: $whereClause = "WHERE RequestorID IN ( '4535','6222','4865','5137','4893')"

This $whereClause gives me the WHERE clause to modify the MSSQL database records being manipulated. So that
when I used $whereClause as the WHERE clause to create subsequent SQL, I used the following syntax:
$setDeleteFlag = "UPDATE AssistanceRequestor SET bIsDirty = 'DELETE' " . $whereClause;

This should have returned: $setDeleteFlag = "UPDATE AssistanceRequestor SET bIsDirty = 'DELETE' WHERE
RequestorID IN ( '4535','6222','4865','5137','4893')"

As long as the length of the original $merge string was less than 104 characters, the $setDeleteFlag sql came
out correctly. However, when the length of the original $merge string exceeded 104 characters, I got this
returned:

$setDeleteFlag = "UPDATE AssistanceRequestor SET bIsDirty = 'DELETE' UPDATE AssistanceRequestor SET bIsDirty =
'DELETE' WHERE RequestorID IN ( '4535','6222','4865','5137','4893')"

The result was that the bIsDirty field for every record in the database was set to 'DELETE'. I fixed it by
breaking apart the substr to create the original $whereClause so that it looked like this:

$wherePosition = strpos($merge,'WHERE');
$whereClause = substr($merge, $wherePosition, strlen($merge) - $wherePosition);
$setDeleteFlag = "UPDATE AssistanceRequestor SET bIsDirty = 'DELETE' " . $whereClause;

I do have to note that I run PHP 5.x on my development server, while I think the production host is still on
4.x. I did not seem to have an issue in development, but I don't think I tested strings longer than 104
characters. Maybe this issue has been corrected in version 5.x.
```

-1 Jarrod Nettles (jarrod at squarecrow dot com) ¶ *3 years ago*

```
I've seen numerous requests over the years from people trying to put together templating systems using XML
parsers or regular expressions - you can create a simple template system with the following class. It could
```

```
easily be expanded to take advantage of parameters, conditionals, etc.

<?php

class Template
{
    const OPEN_BRACKET = "{";
    const CLOSE_BRACKET = "}";

    public static function inject(array $source, $template)
    {
        $ob_size = strlen(self::OPEN_BRACKET);
        $cb_size = strlen(self::CLOSE_BRACKET);

        $pos = 0;
        $end = strlen($template);

        while($pos <= $end)
        {
            if($pos_1 = strpos($template, self::OPEN_BRACKET, $pos))
            {
                if($pos_1)
                {
                    $pos_2 = strpos($template, self::CLOSE_BRACKET, $pos_1);

                    if($pos_2)
                    {
                        $return_length = ($pos_2-$cb_size) - $pos_1;

                        $var = substr($template, $pos_1+$ob_size, $return_length);

                        $template = str_replace(self::OPEN_BRACKET.$var.self::CLOSE_BRACKET, $source[$var],
$template);

                        $pos = $pos_2 + $cb_size;
                    }
                    else
                    {
                        throw new exception("Incorrectly formed template - missing closing bracket. Please
check your syntax.");
                        break;
                    }
                }
            }
            else
            {
                //exit the loop
                break;
            }
        }

        return $template;
    }

}

//array of values to inject into the template
$array = array("NAME" => "John Doe",
                "DOB"    => "12/21/1986",
                "ACL" => "Super Administrator");

//template using '{' and '}' to signify variables
$template = "This is your template, {NAME}. You were born on {DOB} and you are a {ACL} on this system.";

echo Template::inject($array, $template);
?>
```

-1 eallik at hotmail dot com ¶ *7 years ago*

```
Be careful when comparing the return value of substr to FALSE. FALSE may be returned even if the output is a
valid string.

substr("0", 0); // equals "0", comparision with FALSE evaluates to true, because "0" == 0 == FALSE
```

0 greg at apparel dot com ¶ *14 hours ago*

```
Coming to PHP from classic ASP I am used to the Left() and Right() functions built into ASP so I did a quick
```

```
PHPversion. hope these help someone else making the switch

function left($str, $length) {
    return substr($str, 0, $length);
}

function right($str, $length) {
    return substr($str, -$length);
}
```

▲ 0 ▼ Quicker ¶ *2 years ago*

```
If you need to parse utf-8 strings char by char, try this one:

<?php
     $utf8marker=chr(128);
     $count=0;
     while(isset($string{$count})){
       if($string{$count}>=$utf8marker) {
         $parsechar=substr($string,$count,2);
         $count+=2;
       } else {
         $parsechar=$string{$count};
         $count++;
       }
       /* do what you like with parsechar ... , eg.:*/  echo $parsechar."<BR>\r\n";
     }
?>

- it works without mb_substr
- it is fast, because it grabs characters based on indexes  when possible and avoids any count and split
functions
```

▲ 0 ▼ kaysar in ymail in com ¶ *4 years ago*

```
Drop extensions of a file (even from a file location string)

<?php

$filename = "c:/some dir/abc defg. hi.jklmn";

echo substr($filename, 0, (strlen ($filename)) - (strlen (strrchr($filename,'.'))));

?>

output: c:/some dir/abc defg. hi

Hope it may help somebody like me.. (^_^)
```

▲ 0 ▼ webmaster at oehoeboeroe dot nl ¶ *4 years ago*

```
You might expect substr('123456', 6) to return an empty string. Instead it returns boolean FALSE.

This behavior should be mentioned in the Return Values section of the manual. Instead it is only mentioned in
the Parameters section.

If you need an empty string instead of a boolean FALSE you should typecast the result to a string.

<?php
$a = substr('123456', 6);              // equivalent to $a = FALSE
$a = (string) substr('123456', 6);   // equivalent to $a = '';
?>
```

▲ 0 ▼ link ¶ *4 years ago*

```
And as always there is bound to be a bug:

<?php
function strlen_entities($text)
{
    preg_match_all(
        '/((?:&(?:#[0-9]{2,}|[a-z]{2,});)|(?:[^&])|'.
        '(?:&(?!\w;)))s',$text,$textarray);
    return count($textarray[0]);
}
function substr_entities($text,$start,$limit=0)
{
    $return = '';
    preg_match_all(
        '/((?:&(?:#[0-9]{2,}|[a-z]{2,});)|(?:[^&])|'.
        '(?:&(?!\w;)))s',$text,$textarray);
    $textarray = $textarray[0];
    $numchars = count($textarray)-1;
    if ($start>=$numchars)
        return false;
    if ($start<0)
    {
        $start = ($numchars)+$start+1;
    }
    if ($start>=0)
    {
        if ($limit==0)
        {
            $end=$numchars;
        }
        elseif ($limit>0)
        {
            $end = $start+($limit-1);
        }
        else
        {
            $end = ($numchars)+$limit;
        }

        for ($i=$start;($i<=$end && isset($textarray[$i]));$i++)
        {
            $return .= $textarray[$i];
        }
        return $return;
    }
}
?>
```

▲ 0 ▼ Robert Chapin ¶ *6 years ago*

All the references to "curly braces" on this page appear to be obsolete.

According to http://us.php.net/manual/en/language.types.string.php

"Using square array-brackets is preferred because the {braces} style is deprecated as of PHP 6."

Robert Chapin
Chapin Information Services

▲ 0 ▼ feedback at realitymedias dot com ¶ *7 years ago*

This function can replace substr() in some situations you don't want to cut right in the middle of a word.
strtrim will cut between words when it is possible choosing the closest possible final string len to return.
the maxoverflow parameter lets you choose how many characters can overflow past the maxlen parameter.

```
<?php

function strtrim($str, $maxlen=100, $elli=NULL, $maxoverflow=15) {
    global $CONF;

    if (strlen($str) > $maxlen) {

        if ($CONF["BODY_TRIM_METHOD_STRLEN"]) {
            return substr($str, 0, $maxlen);
        }

        $output = NULL;
        $body = explode(" ", $str);
        $body_count = count($body);

        $i=0;

        do {
            $output .= $body[$i]." ";
            $thisLen = strlen($output);
            $cycle = ($thisLen < $maxlen && $i < $body_count-1 && ($thisLen+strlen($body[$i+1])) <
$maxlen+$maxoverflow?true:false);
            $i++;
        } while ($cycle);
        return $output.$elli;
    }
    else return $str;
}

?>
```

▲ 0 ▼ bleakwind at msn dot com ¶ *8 years ago*

```
This returns the portion of str specified by the start and length parameters..
It can performs multi-byte safe on number of characters. like mb_strcut() ...

Note:
1.Use it like this bite_str(string str, int start, int length [,byte of on string]);
2.First character's position is 0. Second character position is 1, and so on...
3.$byte is one character length of your encoding, For example: utf-8 is "3", gb2312 and big5 is "2"...you can
use the function strlen() get it...
Enjoy it :) ...

--- Bleakwind
QQ:940641
http://www.weaverdream.com

PS:I'm sorry my english is too poor... :(

<?php
// String intercept By Bleakwind
// utf-8:$byte=3 | gb2312:$byte=2 | big5:$byte=2
function bite_str($string, $start, $len, $byte=3)
{
    $str     = "";
    $count   = 0;
    $str_len = strlen($string);
    for ($i=0; $i<$str_len; $i++) {
        if (($count+1-$start)>$len) {
            $str  .= "...";
            break;
        } elseif ((ord(substr($string,$i,1)) <= 128) && ($count < $start)) {
            $count++;
        } elseif ((ord(substr($string,$i,1)) > 128) && ($count < $start)) {
            $count = $count+2;
            $i     = $i+$byte-1;
        } elseif ((ord(substr($string,$i,1)) <= 128) && ($count >= $start)) {
            $str  .= substr($string,$i,1);
            $count++;
        } elseif ((ord(substr($string,$i,1)) > 128) && ($count >= $start)) {
            $str  .= substr($string,$i,$byte);
            $count = $count+2;
            $i     = $i+$byte-1;
        }
    }
    return $str;
}

// Test
$str = "123456����123456����123456��ɖ����";
for($i=0;$i<30;$i++){
    echo "<br>".bite_str($str,$i,20);
}
?>
```

▲ 0 ▼ vitalic#pisem.net ¶ *8 years ago*

```
Split $string after each $pos, by $space
Example: <?php spaceStr('1836254','-',3); ?>
Would return '183-625-4';

<?php
function spaceStr($string,$space,$pos)
{
       $cpos=$pos;
       while ($cpos<strlen($string))
       {
         $string=substr($string,0,$cpos).$space.substr($string,$cpos);
         $cpos+=strlen($space)+$pos;
       };
       return $string;
}

?>
```

▲ -1 ▼ smapan atsign gmail dot com ¶ *2 months ago*

```
/**
Simple function for get text between text
@param string $input   The input string. Must be one character or longer.
@param string $start    The start string. Must be one character or longer.
@param string $start    The end string. Must be one character or longer.
@Returns the extracted part of string betweent start end end string; or FALSE on failure, or an empty string.
*/

function get_text_between($input, $start, $end)
{
    $a = strpos($input, $start);
    $b = strlen($start);
    return substr($input, $a, strpos($input, $end, $a + $b) - $a + $b*2);
}
```

▲ -2 ▼ benny at bennyborn dot de ¶ *4 years ago*

If you need a word-sensitive and also html-tags aware version of substr, this one should do the job. It works fine for me

```
<?php
/**
* word-sensitive substring function with html tags awareness
* @param text The text to cut
* @param len The maximum length of the cut string
* @returns string
**/
function substrws( $text, $len=180 ) {

    if( (strlen($text) > $len) ) {

        $whitespaceposition = strpos($text," ",$len)-1;

        if( $whitespaceposition > 0 )
            $text = substr($text, 0, ($whitespaceposition+1));

        // close unclosed html tags
        if( preg_match_all("|<([a-zA-Z]+)>|",$text,$aBuffer) ) {

            if( !empty($aBuffer[1]) ) {

                preg_match_all("|</([a-zA-Z]+)>|",$text,$aBuffer2);

                if( count($aBuffer[1]) != count($aBuffer2[1]) ) {

                    foreach( $aBuffer[1] as $index => $tag ) {

                        if( empty($aBuffer2[1][$index]) || $aBuffer2[1][$index] != $tag)
                            $text .= '</'.$tag.'>';
                    }
                }
            }
        }
    }

    return $text;
}
?>
```

▲ -2 ▼ Anonymous ¶ *5 years ago*

I wrote this simple function to limit the middle characters of a string to a specified length.

```
<?php
$input = "hello world"
echo(limitchrmid($imput,10)) // hel ... rld

//limit chars middle
function limitchrmid($value,$lenght){
    if (strlen($value) >= $lenght ){
        $lenght_max = ($lenght/2)-3;
        $start = strlen($value)- $lenght_max;
        $limited = substr($value,0,$lenght_max);
        $limited.= " ... ";
        $limited.= substr($value,$start,$lenght_max);
    }
    else{
        $limited = $value;
    }
    return $limited;
}
?>
```

add a note