str_replace

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

str_replace — Reemplaza todas las apariciones del string buscado con el string de reemplazo

Descripción ¶

str_replace(
  mixed `$search`,
  mixed `$replace`,
  mixed `$subject`,
  int `&$count` = ?
): mixed

Esta función devuelve un string o un array con todas las apariciones de `search` en `subject` reemplazadas con el valor dado de `replace`.

Si no se necesitan reglas complicadas de reemplazo (como expresiones regulares), se puede utilizar siempre esta función en lugar de preg_replace().

Parámetros ¶

Si `search` y `replace` son arrays, entonces **str_replace()** toma un valor de cada array y lo utiliza para buscar y reemplazar en `subject`. Si `replace` tiene menos valores que `search`, entonces un string vacío es usado para el resto de los valores de reemplazo. Si `search` es un array y `replace` es un string, entonces este string de reemplazo es usado para cada valor de `search`. Sin embargo, lo contrario no tendría sentido.

Si `search` o `replace` son arrays, sus elementos son procesados del primero al último.

`search`

El valor a ser buscado, también conocida como la *aguja*. Un array puede ser utilizado para designar varias agujas.

`replace`

El valor de reemplazo que sustituye los valores encontrados de `search`. Un array puede ser utilizado para designar reemplazos múltiples.

`subject`

El string o array sobre el que se busca y se sustituye, también conocido como el *pajar*.

Si `subject` es un array, entonces la búsqueda y reemplazo se realiza con cada entrada de `subject` y el valor devuelto también es un array.

`count`

Si es pasado, con este se establece el número de reemplazos realizados.

Valores devueltos ¶

Esta función devuelve un string o un array con los valores sustituidos.

Ejemplos ¶

Ejemplo #1 Ejemplos básicos de str_replace()

```php
<?php
// Produce: <body text='black'>
$bodytag = str_replace("%body%", "black", "<body text='%body%'>");

// Produce: Hll Wrld f PHP
$vowels = array("a", "e", "i", "o", "u", "A", "E", "I", "O", "U");
```

```php
$onlyconsonants = str_replace($vowels, "", "Hello World of PHP");

// Produce: You should eat pizza, beer, and ice cream every day
$phrase  = "You should eat fruits, vegetables, and fiber every day.";
$healthy = array("fruits", "vegetables", "fiber");
$yummy   = array("pizza", "beer", "ice cream");

$newphrase = str_replace($healthy, $yummy, $phrase);

// Produce: 2
$str = str_replace("ll", "", "good golly miss molly!", $count);
echo $count;
?>
```

Ejemplo #2 Ejemplos de trampas potenciales con str_replace()

```php
<?php
// Orden del reemplazo
$str     = "Line 1\nLine 2\rLine 3\r\nLine 4\n";
$order   = array("\r\n", "\n", "\r");
$replace = '<br />';

// Procesa primero \r\n así no es convertido dos veces.
$newstr = str_replace($order, $replace, $str);

// La salida es F debido a que A es reemplazada con B, entonces B es reemplazada con C, y así sucesivamente...
// Finalmente E se reemplaza con F, debido a los reemplazos de izquierda a derecha.
$search  = array('A', 'B', 'C', 'D', 'E');
$replace = array('B', 'C', 'D', 'E', 'F');
$subject = 'A';
echo str_replace($search, $replace, $subject);

// La salida es: apearpearle pear
// Por la misma razón mencionada arriba
$letters = array('a', 'p');
$fruit   = array('apple', 'pear');
$text    = 'a p';
$output  = str_replace($letters, $fruit, $text);
echo $output;
?>
```

Notas ¶

> **Nota**: Esta función es segura binariamente.

Precaución

Trampa del orden de reemplazo

Debido a que **str_replace()** reemplaza de izquierda a derecha, podría sustituir a un valor previamente insertado al hacer sustituciones múltiples. Ver también los ejemplos de este documento.

> **Nota**:
>
> Esta función es sensible a mayúsculas y minúsculas. Utilice Use str_ireplace() para reemplazos insensibles a mayúsculas y minúsculas.

Ver también ¶

- str_ireplace() - Versión insensible a mayúsculas y minúsculas de str_replace
- substr_replace() - Reemplaza el texto dentro de una porción de un string
- preg_replace() - Realiza una búsqueda y sustitución de una expresión regular
- strtr() - Convierte caracteres o reemplaza substrings

+ add a note

User Contributed Notes 34 notes

up
down
257
nikolaz dot tang at hotmail dot com ¶
12 years ago

A faster way to replace the strings in multidimensional array is to json_encode() it, do the str_replace() and then json_decode() it, like this:

```
<?php
function str_replace_json($search, $replace, $subject){
     return json_decode(str_replace($search, $replace,  json_encode($subject)));

}
?>
```

This method is almost 3x faster (in 10000 runs.) than using recursive calling and looping method, and 10x simpler in coding.

Compared to:

```
<?php
function str_replace_deep($search, $replace, $subject)
{
    if (is_array($subject))
    {
        foreach($subject as &$oneSubject)
            $oneSubject = str_replace_deep($search, $replace, $oneSubject);
        unset($oneSubject);
        return $subject;
    } else {
        return str_replace($search, $replace, $subject);
    }
}
?>
```

up
down
62
moostende at gmail dot com ¶
11 years ago

Note that this does not replace strings that become part of replacement strings. This may be a problem when you want to remove multiple instances of the same repetative pattern, several times in a row.

If you want to remove all dashes but one from the string '-aaa----b-c-----d--e---f' resulting in '-aaa-b-c-d-e-f', you cannot use str_replace. Instead, use preg_replace:

```
<?php
$challenge = '-aaa----b-c-----d--e---f';
echo str_replace('--', '-', $challenge).'<br>';
echo preg_replace('/--+/', '-', $challenge).'<br>';
?>
```

This outputs the following:

```
-aaa--b-c---d-e--f
-aaa-b-c-d-e-f
```

up
down
48
Alberto Lepe ¶
13 years ago

Be careful when replacing characters (or repeated patterns in the FROM and TO arrays):

For example:

```
<?php
$arrFrom = array("1","2","3","B");
$arrTo = array("A","B","C","D");
$word = "ZBB2";
echo str_replace($arrFrom, $arrTo, $word);
?>
```

I would expect as result: "ZDDB"
However, this return: "ZDDD"
(Because B = D according to our array)

To make this work, use "strtr" instead:

```
<?php
$arr = array("1" => "A","2" => "B","3" => "C","B" => "D");
$word = "ZBB2";
echo strtr($word,$arr);
?>
```

This returns: "ZDDB"

up
down
36

Wes Foster ¶

12 years ago

Feel free to optimize this using the while/for or anything else, but this is a bit of code that allows you to replace strings found in an associative array.

For example:

```
<?php
$replace = array(
'dog' => 'cat',
'apple' => 'orange'
'chevy' => 'ford'
);

$string = 'I like to eat an apple with my dog in my chevy';

echo str_replace_assoc($replace,$string);

// Echo: I like to eat an orange with my cat in my ford
?>
```

Here is the function:

```
<?php
function strReplaceAssoc(array $replace, $subject) {
   return str_replace(array_keys($replace), array_values($replace), $subject);
}
?>
```

[Jun 1st, 2010 - EDIT BY thiago AT php DOT net: Function has been replaced with an updated version sent by ljelinek AT gmail DOT com]

up
down
6

ravenswd at gmail dot com ¶

4 years ago

This is what happens when the search and replace arrays are different sizes:

```
<?php
$search  = array('a', 'b', 'c', 'd', 'e');
$replace = array('A', 'B', 'C');
$subject = 'abcdefg';
echo str_replace($search, $replace, $subject);
// result: 'ABCfg'

$search  = array('a', 'b', 'c');
$replace = array('A', 'B', 'C', 'D', 'E');
$subject = 'abcdefg';
echo str_replace($search, $replace, $subject);
// result: 'ABCdefg'
?>
```

No warning or error is generated in either of these cases.

up
down
13

David Holt ¶

6 years ago

Be aware that if you use this for filtering & sanitizing some form of user input, or remove ALL instances of a string, there's another gotcha to watch out for:

```
// Remove all double characters
$string="1001011010";
$string=str_replace(array("11","00"),"",$string);
// Output: "110010"

$string="<ht<html>ml> Malicious code </<html>html> etc";
$string=str_replace(array("<html>","</html>"),"",$string);
// Output: "<html> Malicious code </html> etc"
```

up
down
9

michael dot moussa at gmail dot com ¶

13 years ago

As previous commentators mentioned, when $search contains values that occur earlier in $replace, str_replace will factor those previous replacements into the process rather than operating solely on the original string. This may produce unexpected output.

Example:

```
<?php
$search = array('A', 'B', 'C', 'D', 'E');
$replace = array('B', 'C', 'D', 'E', 'F');
$subject = 'ABCDE';

echo str_replace($search, $replace, $subject); // output: 'FFFFFF'
?>
```

In the above code, the $search and $replace should replace each occurrence in the $subject with the next letter in the alphabet. The expected output for this sample is 'BCDEF'; however, the actual output is 'FFFFF'.

To more clearly illustrate this, consider the following example:

```
<?php
$search = array('A', 'B', 'C', 'D', 'E');
$replace = array('B', 'C', 'D', 'E', 'F');
$subject = 'A';

echo str_replace($search, $replace, $subject); // output: 'F'
?>
```

Since 'A' is the only letter in the $search array that appears in $subject, one would expect the result to be 'B'; however, replacement number $n does *not* operate on $subject, it operates on $subject after the previous $n-1 replacements have been completed.

The following function utilizes array_combine and strtr to produce the expected output, and I believe it is the most efficient way to perform the desired string replacement without prior replacements affecting the final result.

```
<?php
/**
* When using str_replace(...), values that did not exist in the original string (but were put there by
previous
* replacements) will be replaced continuously.  This string replacement function is designed replace the
values
* in $search with those in $replace while not factoring in prior replacements.  Note that this function will
* always look for the longest possible match first and then work its way down to individual characters.
*
* The "o" in "stro_replace" represents "original", indicating that the function operates only on the
original string.
*
* @param array $search list of strings or characters that need to be replaced
* @param array $replace list of strings or characters that will replace the corresponding values in $search
* @param string $subject the string on which this operation is being performed
*
* @return string $subject with all substrings in the $search array replaced by the values in the $replace
array
*/
function stro_replace($search, $replace, $subject)
{
    return strtr( $subject, array_combine($search, $replace) );
}

$search = array('A', 'B', 'C', 'D', 'E');
$replace = array('B', 'C', 'D', 'E', 'F');
$subject = 'ABCDE';

echo stro_replace($search, $replace, $subject); // output: 'BCDEF'
?>
```

Some other examples:

```
<?php
$search = array(' ', '&');
$replace = array(' ', '&amp;');
$subject = 'Hello & goodbye!';

// We want to replace the spaces with   and the ampersand with &amp;
echo str_replace($search, $replace, $subject); // output: "Hello&amp;nbsp&amp;&amp;nbspgoodbye!" - wrong!

echo stro_replace($search, $replace, $subject); // output: "Hello &amp; goodbye!" - correct!

/*
    Note:  Run the above code in the CLI or view source on your web browser - the replacement strings for
stro_replace are HTML entities which the browser interprets.
*/
?>

<?php
$search = array('ERICA', 'AMERICA');
$replace = array('JON', 'PHP');
$subject = 'MIKE AND ERICA LIKE AMERICA';

// We want to replace the name "ERICA" with "JON" and the word "AMERICA" with "PHP"
echo str_replace($search, $replace, $subject); // output: "MIKE AND JON LIKE AMJON", which is not correct
```

```
echo stro_replace($search, $replace, $subject); // output: "MIKE AND JON LIKE PHP", which is correct
?>
```

up
down
8
mrrehbein at gmail dot com ¶
7 years ago

nikolaz dot tang at hotmail dot com's solution of using json_encode/decode is interesting, but a couple of issues to be aware of with it.

```
<?php
// From: nikolaz dot tang at hotmail dot com's post
function str_replace_json($search, $replace, $subject){
     return json_decode(str_replace($search, $replace,  json_encode($subject)));
}
?>
```

json_decode will return objects, where arrays are probably expected. This is easily remedied by adding 2nd parameter 'true' to json_decode.

$search and $replace could contain strings that match json encoding, which will either change the structure returned by this method, or break the json.

ie:

```
<?php
var_dump(str_replace_json('":"', '","', ['this' => 'stuff']));
var_dump(str_replace_json('this":"', 'this" : "thing", "with":"', ['this' => 'stuff']));
?>
```

up
down
12
jay_knows_(all)uk at hotmail dot com ¶
11 years ago

This strips out horrible MS word characters.

Just keep fine tuning it until you get what you need, you'll see ive commented some out which caused problems for me.

There could be some that need adding in, but its a start to anyone who wishes to make their own custom function.

```
<?php

function msword_conversion($str)
{
$str = str_replace(chr(130), ',', $str);    // baseline single quote
$str = str_replace(chr(131), 'NLG', $str);  // florin
$str = str_replace(chr(132), '"', $str);    // baseline double quote
$str = str_replace(chr(133), '...', $str);  // ellipsis
$str = str_replace(chr(134), '**', $str);   // dagger (a second footnote)
$str = str_replace(chr(135), '***', $str);  // double dagger (a third footnote)
$str = str_replace(chr(136), '^', $str);    // circumflex accent
$str = str_replace(chr(137), 'o/oo', $str); // permile
$str = str_replace(chr(138), 'Sh', $str);   // S Hacek
$str = str_replace(chr(139), '<', $str);    // left single guillemet
// $str = str_replace(chr(140), 'OE', $str);   // OE ligature
$str = str_replace(chr(145), "'", $str);    // left single quote
$str = str_replace(chr(146), "'", $str);    // right single quote
// $str = str_replace(chr(147), '"', $str);    // left double quote
// $str = str_replace(chr(148), '"', $str);    // right double quote
$str = str_replace(chr(149), '-', $str);    // bullet
$str = str_replace(chr(150), '--', $str);    // endash
$str = str_replace(chr(151), '--', $str);   // emdash
```

```
// $str = str_replace(chr(152), '~', $str);    // tilde accent
// $str = str_replace(chr(153), '(TM)', $str); // trademark ligature
$str = str_replace(chr(154), 'sh', $str);   // s Hacek
$str = str_replace(chr(155), '>', $str);    // right single guillemet
// $str = str_replace(chr(156), 'oe', $str);   // oe ligature
$str = str_replace(chr(159), 'Y', $str);    // Y Dieresis
$str = str_replace('°C', '&deg;C', $str);    // Celcius is used quite a lot so it makes sense to add this in
$str = str_replace('£', '&pound;', $str);
$str = str_replace("'", "'", $str);
$str = str_replace('"', '"', $str);
$str = str_replace('–', '&ndash;', $str);

return $str;
}

?>
```

up
down
7

pjcdawkins at googlemail dot com ¶

12 years ago

Here's a deep replace function allowing multi-dimensional arrays in $search, $replace and $subject. The keys and other structure of $subject are preserved.

```
<?php
// Auxiliary function:
function _replaceWithAnything($search,$replace,$subject){
  if(!is_array($search) || !is_array($replace)){
    $search=array($search);
    $replace=array($replace);
  }
  $match=array_search($subject,$search,true);
  if($match!==false && array_key_exists($match,$replace))
    $subject=$replace[$match];
  return $subject;
}

// Main function:
function deepReplace($search,$replace,$subject){
  if(!is_array($subject))
    return _replaceWithAnything($search,$replace,$subject);
  foreach($subject as &$val){
    if(is_array($val)){
      $val=deepReplace($search,$replace,$val);
      continue;
    }
    $val=_replaceWithAnything($search,$replace,$val);
  }
  return $subject;
}
?>
```

up
down
9

jbarnett at jmbelite dot com ¶

12 years ago

Might be worth mentioning that a SIMPLE way to accomplish Example 2 (potential gotchas) is to simply start your "replacements" in reverse.

So instead of starting from "A" and ending with "E":

```
<?php
$search  = array('A', 'B', 'C', 'D', 'E');
$replace = array('B', 'C', 'D', 'E', 'F');
```

```
// replaces A to B, B to C, C to D, D to E, E to F (makes them all F)
// start from "E" and end with "A":

$search  = array('E', 'D', 'C', 'B', 'A');
$replace = array('F', 'E', 'D', 'C', 'B');
// replaces E to F, D to E, C to D, B to C, A to B (prevents from
// multiple replacements of already replaced values)
?>
```

So basically start from the "end" and put the replacements in an order where the "replaced value" won't equal a value that exists later in the "search array".

up
down
6

matt wheaton ¶
16 years ago

As an effort to remove those Word copy and paste smart quotes, I've found that this works with UTF8 encoded strings (where $text in the following example is UTF8). Also the elipsis and em and en dashes are replaced.

There is an "invisible" character after the â€ for the right side double smart quote that doesn't seem to display here. It is chr(157).

```
<?php
  $find[] = 'â€œ';  // left side double smart quote
  $find[] = 'â€ ';  // right side double smart quote
  $find[] = 'â€˜';  // left side single smart quote
  $find[] = 'â€™';  // right side single smart quote
  $find[] = 'â€¦';  // elipsis
  $find[] = 'â€”';  // em dash
  $find[] = 'â€“';  // en dash

  $replace[] = '"';
  $replace[] = '"';
  $replace[] = "'";
  $replace[] = "'";
  $replace[] = "...";
  $replace[] = "-";
  $replace[] = "-";

  $text = str_replace($find, $replace, $text);
?>
```

up
down
4

Ing. Mirko Plazotta ¶
8 years ago

```
<?php
// a very beatiful way to do multiple replacements is this one, using just one array
$replaceThis = Array(
'old word' => 'new word',
'was' => 'it',
'past' => 'future',
);

$originalText = "every old word was a thing of the past...";
$replacedText = str_replace(array_keys($replaceThis), $replaceThis, $originalText);
echo $replacedText;
?>
```

up
down
2

karst at onlinq dot nl ¶
8 years ago

"If search is an array and replace is a string, then this replacement string is used for every value of search. The converse would not make sense, though. "

I think one important (and not at all vaguely theoretical) use-case is completely ignored here. Take, for example, the way the PDO handles parameter replacement.

If we have the following query:
"SELECT * FROM my_table WHERE (id = ? AND my_column = ? AND other_column = ?);"
The "?"s should be replaced by each successive variable in a $parameters array. That is EXACTLY the use case for "search" being a value and "replace" being an array.

Considering that this is not only a real-world example but also part of a core PHP functionality I find it very strange that it's dismissed so easily here.

up
down
1

viundan at gmail dot com ¶

6 years ago

Decision to avoid problem "it might replace a previously inserted value when doing multiple replacements. See also the examples in this document."

$urls - array of urls i want to replace with tag <a> and urls could be similar
http://abc.com/parameter/
http://abc.com/

```
// at first sort by length to have longest firstly
usort($urls,'sortByLen');

$replaces=[];

// replace all urls with unique
foreach($urls as $url){
    $replace = '__REPLACE' . uniqid() . '__';
    $text = str_replace($url,$replace, $text);
    $replaces[$replace] = '<a href="' . $url . '">' . $url . '</a>';
}

foreach($replaces as $key => $replace){
    $text = str_replace($key,$replace, $text);
}

--------------

function sortByLen($a,$b){
    return strlen($b)-strlen($a);
}
```

Hope it will help others like me

up
down
2

apmuthu at usa dot net ¶

12 years ago

If we have a html template that contains placeholders in curly braces that need to be replaced in runtime, the following function will do it using str_replace:

```
<?php

function parse_template($filename, $data) {
// example template variables {a} and {bc}
// example $data array
// $data = Array("a" => 'one', "bc" => 'two');
    $q = file_get_contents($filename);
    foreach ($data as $key => $value) {
```

```
        $q = str_replace('{'.$key.'}', $value, $q);
    }
    return $q;
}

?>
```

up
down
3

Denzel Morris ¶

10 years ago

Maybe obvious to veteran PHP programmers but less so to novice PHP programmers is the fact that this is invalid:

```
<?php
str_replace($search, $replace, $subject, 1);
?>
```

At a glance it appears to be a reasonable request, until you realize that the fourth parameter must be a variable in order to be passed as a reference. A replacement:

```
<?php
str_replace($search, $replace, $subject, $temp = 1);
// or
$temp = 1;
str_replace($search, $replace, $subject, $temp);
?>
```

up
down
3

kriscraig at php dot net ¶

10 years ago

```
<?php

/**
* Convert foreign 8859-1 characters into HTML entities.
*
* @param string $str
*   The string being parsed.
*
* @return string
*   The converted string.
*/
public static function convert_chars_to_entities( $str )
{
    $str = str_replace( 'À', '&#192;', $str );
    $str = str_replace( 'Á', '&#193;', $str );
    $str = str_replace( 'Â', '&#194;', $str );
    $str = str_replace( 'Ã', '&#195;', $str );
    $str = str_replace( 'Ä', '&#196;', $str );
    $str = str_replace( 'Å', '&#197;', $str );
    $str = str_replace( 'Æ', '&#198;', $str );
    $str = str_replace( 'Ç', '&#199;', $str );
    $str = str_replace( 'È', '&#200;', $str );
    $str = str_replace( 'É', '&#201;', $str );
    $str = str_replace( 'Ê', '&#202;', $str );
    $str = str_replace( 'Ë', '&#203;', $str );
    $str = str_replace( 'Ì', '&#204;', $str );
    $str = str_replace( 'Í', '&#205;', $str );
    $str = str_replace( 'Î', '&#206;', $str );
    $str = str_replace( 'Ï', '&#207;', $str );
    $str = str_replace( 'Ð', '&#208;', $str );
    $str = str_replace( 'Ñ', '&#209;', $str );
    $str = str_replace( 'Ò', '&#210;', $str );
    $str = str_replace( 'Ó', '&#211;', $str );
    $str = str_replace( 'Ô', '&#212;', $str );
    $str = str_replace( 'Õ', '&#213;', $str );
```

```
    $str = str_replace( 'Ö', '&#214;', $str );
    $str = str_replace( '×', '&#215;', $str );  // Yeah, I know.  But otherwise the gap is confusing.  --
Kris
    $str = str_replace( 'Ø', '&#216;', $str );
    $str = str_replace( 'Ù', '&#217;', $str );
    $str = str_replace( 'Ú', '&#218;', $str );
    $str = str_replace( 'Û', '&#219;', $str );
    $str = str_replace( 'Ü', '&#220;', $str );
    $str = str_replace( 'Ý', '&#221;', $str );
    $str = str_replace( 'Þ', '&#222;', $str );
    $str = str_replace( 'ß', '&#223;', $str );
    $str = str_replace( 'à', '&#224;', $str );
    $str = str_replace( 'á', '&#225;', $str );
    $str = str_replace( 'â', '&#226;', $str );
    $str = str_replace( 'ã', '&#227;', $str );
    $str = str_replace( 'ä', '&#228;', $str );
    $str = str_replace( 'å', '&#229;', $str );
    $str = str_replace( 'æ', '&#230;', $str );
    $str = str_replace( 'ç', '&#231;', $str );
    $str = str_replace( 'è', '&#232;', $str );
    $str = str_replace( 'é', '&#233;', $str );
    $str = str_replace( 'ê', '&#234;', $str );
    $str = str_replace( 'ë', '&#235;', $str );
    $str = str_replace( 'ì', '&#236;', $str );
    $str = str_replace( 'í', '&#237;', $str );
    $str = str_replace( 'î', '&#238;', $str );
    $str = str_replace( 'ï', '&#239;', $str );
    $str = str_replace( 'ð', '&#240;', $str );
    $str = str_replace( 'ñ', '&#241;', $str );
    $str = str_replace( 'ò', '&#242;', $str );
    $str = str_replace( 'ó', '&#243;', $str );
    $str = str_replace( 'ô', '&#244;', $str );
    $str = str_replace( 'õ', '&#245;', $str );
    $str = str_replace( 'ö', '&#246;', $str );
    $str = str_replace( '÷', '&#247;', $str );  // Yeah, I know.  But otherwise the gap is confusing.  --
Kris
    $str = str_replace( 'ø', '&#248;', $str );
    $str = str_replace( 'ù', '&#249;', $str );
    $str = str_replace( 'ú', '&#250;', $str );
    $str = str_replace( 'û', '&#251;', $str );
    $str = str_replace( 'ü', '&#252;', $str );
    $str = str_replace( 'ý', '&#253;', $str );
    $str = str_replace( 'þ', '&#254;', $str );
    $str = str_replace( 'ÿ', '&#255;', $str );

    return $str;
}
?>
```

up
down
2

ressing1 at gmail dot com ¶

1 year ago

```
To remove all characters from string $b that exist in string $a:

$a="ABC";
$b="teAsBtC";
echo str_replace(str_split($a),'',$b);

Output: test

To remove all characters from string $b that don't exist in string $a:

$a="ABC";
```

```
$b="teAsBtC";
echo str_replace(str_split(str_replace(str_split($a),'',$b)),'',$b);

Output: ABC
```

up
down
1

mbullard at accuvista dot co dot uk ¶

11 years ago

```
Insert space after comma.

If you have a form that stores results in a database field as comma separated values, when you display this
data you can use the following to insert a space after each comma:

<?php
$find[] = ',';
$replace[] = '&#44; ';
$text = str_replace($find, $replace, $row_rsRecordset['Field']);
print_r($text);
?>

Notes:
1) To get round the Replacement Order Gotcha, the comma is also replaced with its code equivalent: &#44;
2) You can adapt the $replace section to suit your needs: swap out the   code with <br/> or replace
comma and space with  &middot;  etc.
```

up
down
0

Mohsin Ali (puregenius02 at gmail dot com) ¶

6 years ago

```
$myString = “It was the best of mine it was the worst of mine,”;
// Displays “It was the best of bananas, it was the worst of bananas,”
echo str_replace( “mine”, “bananas”, $myString );
If you want to know how many times the search string was replaced, pass in a variable as an optional
fourth argument. After the function runs, this variable holds the number of replacements:
$myString = “It was the best of mine, it was the worst of mine,”;
// Displays “It was the best of bananas, it was the worst of bananas,”
echo str_replace( “mine”, “bananas”, $myString, $num ) . “ < br/ > ”;
// Displays “The text was replaced 2 times.”
echo “The text was replaced $num times. < br/ > ”;
```

up
down
0

markem at sim1 dot us ¶

8 years ago

```
I was working with MySQL and displaying the title to things on the web page.  I'd written a script to ensure
single and double quotes were removed from the title.  I used

    $title = str_replace( "'", "", $title );

and

     $title = str_replace( '"', "", $title );

But still the single and double quotes continued.  So I wrote a bit of code to print out each character
separated by a dash.  Like so:

     for( $i=0; $i<strlen($title); $i++ ){
         echo "$i-";
         }

     echo "<br>\n";

This displayed:
```

```
     m-y-c-o-m-p-a-n-y- b-b-&-#-3-9-;-s
```

Which made me go "Oh! I get it."

The MySQL function real_escape_string modifies the single quotes to be ' and double quotes as "
These still show up as single and double quotes under HTML and most importantly -

```
     JAVASCRIPT sees the &#34; and &#39; as actual single or double
     quotes.  So if you are passing arguments to a function you have
     to get rid of them or else you will get an error on trying to call
     a given function.  Example:

     <a href="javascript:func1('mycompany bbs&#39;s")'">

     becomes

     <a href="javascript:func1('mycompany bbs's');">
```

Which then will give you an error because there is a single quote inside of the single quoted string.
HOWEVER, the

```
     $title = str_replace( "'", "", $title );
```

WILL NOT FIND a single quote. Instead, you have to do this:

```
     $title = str_replace( "&#39;", "'", $title );
```

and

```
     $title = str_relace( "&#34;", '"', $title );
```

(Or you could just get rid of them.)

So remember! If you are trying to remove single and double quotes and are using MySQL and MySQL's real_escape_string() function that you might be having single and double quotes hanging around which are defined as ' and " but which show up as single and double quotes as well as causing problems in your Javascripts.

up
down
-2

cc at cc dot com ¶

10 years ago

I found a pretty low tech solution to avoid the "gotcha" without worrying about the array order of how things are replaced. I could not "order" the replacement array easily because it was being read from a database table.

Anyway if you add an identifiable token to each replaced word, then just filter this out at the very end, no nested search terms are found. I just dynamically add the %% after the first char of each word before pumping it into the str_ireplace function.

```
$find = array("as1", "as2", "as3", "flex");
$replace = array("<a href = \"#as1\">A%%uto S%%entry R%%ev. A%%</a>", "<a href = \"#as2\">A%%uto S%%entry E%%xp</a>", "<a href = \"#as3\">A%%uto S%%entry f%%lex</a>", "<a style = \"color: red;\" href = \"#flex\">f%%lex</a>");
$text = str_ireplace($find, $replace, $text);
echo str_ireplace("%%", "", $text);
```

In this case I am using %% as my token as this is an unlikely char combo for me.

up
down
-2

jefrey at forteras dot tech ¶

4 years ago

NEVER USE this function to protect against SQL Injection.

It may sound ridiculous but I've seen a couple of developers doing so.

It's interesting that these developers use str_replace (let's ignore the fact that they don't even use the str_ireplace which is case-insensitive) to remove common SQL commands such as "SELECT" or "DROP" from user-entered inputs.

A funny thing to note is that:

```
<?php
$input = "SELSELECTECT";
echo str_replace("SELECT", null, $input); // = SELECT
?>
```

Yeah you could loop it, but str_replace was never meant to be used this way. There are proper ways to protect against SQL Injections, such as using prepared statements (placeholders).

up
down
-3

borasahin at gmail dot com ¶
7 years ago

jSON Turkish Characters Problem - (PHP < 5.4 for example)

```
<?php
function json_decode_tr($json){
       $json_char =
array("u00e7","u0131","u00fc","u011f","u00f6","u015f","u0130","u011e","u00dc","u00d6","u015e","u00c7");
       $turkish = array("ç","ı","ü","ğ","ö","ş","İ","Ğ","Ü","Ö","Ş","Ç");
       $result = str_replace($json_char, $turkish, $json);
       return json_decode($json);
    }
?>
```

up
down
-4

christof dot rieger at r-tron dot de ¶
10 years ago

In many countries the numeric format is 1.000,33 in english it is 1,000.33

This function converts numeric arguments always into the PHP confirm numeric format. If only one seperator is into the numericstring so it is interpreted as the decimalpoint.

```
function dp($zahl)
{
  if ((strpos($zahl,".") > "-1") | (strpos($zahl,",") > "-1")) {
    if ((strpos($zahl,".") > "-1") & (strpos($zahl,",") > "-1")) {
      if (strpos($zahl,".") > strpos($zahl,",")){
          return str_replace(",","",$zahl);
    } else {
          return str_replace(",",".",str_replace(".","",$zahl));
      }
  } else {
      if (strpos($zahl,".") > "-1") {
        if (strpos($zahl,".") == strrpos($zahl,".")) {
           return $zahl;
      } else {
          return str_replace(".","",$zahl);
        }
    } else {
        if (strpos($zahl,",") == strrpos($zahl,",")) {
          return str_replace(",",".",$zahl);
      } else {
          return str_replace(",","",$zahl);
        }
    } }
```

```
} else {
    return $zahl;
} }
```

up
down
-4
flame2000 at mail dot ru ¶
6 years ago

```
Replace chars in multi-byte string.
In example, replacing 'f'=>'b', 'o'=>'e', 't'=>'r' and etc.

<?php
function mb_chars_replace($from, $to, $subj, $delSymb='_') {
     $nsubj='';
     preg_match_all('/(.)/u', $subj, $subj);$subj=$subj[1];
     if (!is_array($from)) {preg_match_all('/(.)/u', $from, $from);$from=$from[1];}
     if (!is_array($to)) {preg_match_all('/(.)/u', $to, $to);$to=$to[1];}
     if (count($from)!==count($to)) return false;

     foreach($subj as $s) {
       foreach($from as $k=>$f) {
         if($s===$f) {
                    $s=$to[$k];
                    break;
         }
       }
       if($s!==$delSymb) $nsubj.=$s;
     }
    return $nsubj;
}

//examples:

$from="fotber, ";
$to="berfot+_";

// or

$from=array("f","o","t","b","e","r",",","," ");
$to=array("b","e","r","f","o","t","+","_");

//out:
echo mb_chars_replace($from,$to,"foot, beer"); //beer+foot
?>
```

up
down
-3
ASchmidt at Anamera dot net ¶
4 years ago

```
Escaping strings with control characters, quotes and backslashes for subsequent use in MySQL commands.

MySQL has documented a number of backslash escape sequences that need to be used to pass certain values in
SQL commands: https://dev.mysql.com/doc/refman/5.7/en/string-literals.html

It's crucial to escape existing backslashes first to prevent double-escaping, before escaping the various
control sequences:

<?php
$result = str_replace(
                array( '\\',    "\0",   "'",    "\x8" /* BS */, "\n",   "\r",   "\t",   "\x1A" /* Ctrl+Z */
),
                array( '\\\\',  '\\0',  '\\''', '\\b',          '\\n',  '\\r',  '\\t',  '\\Z' ),
                $value );
?>
```

This code is NOT intended to protect against SQL insertions, it's intended to PRESERVE string content correctly, if it contains control characters.

up
down
-3

vuabid at hotmail dot com ¶
5 years ago

Consider this while using str_replace function when $search and $replace are arrays.

```
$search = array( 'login_reactivate', 'login_reactivate_date' );
$replace = array( 'login reactivate status', 'login reactivate date' );
$subject = "fname, email, login_reactivate, login_reactivate_date";

$returnValue = str_replace( $search, $replace, $subject );

$returnValue will be:
fname, email, login reactivate status, login reactivate status_date
```

You can see we are expecting "login_reactivate_date" to be replaced with "login reactivate date" but it will replace to "login reactivate status_date"

up
down
-3

christian dot reinecke at web dot de ¶
12 years ago

If you need to replace a string in another, but only once but still in all possible combinations (f.e. to replace "a" with "x" in "aba" to get array("xba", "abx")) you can use this function:

```
<?php
function getSingleReplaceCombinations($replace, $with, $inHaystack)
{
    $splits = explode($replace, $inHaystack);
    $result = array();
    for ($i = 1, $ix = count($splits); $i < $ix; ++$i) {
        $previous = array_slice($splits, 0, $i);
        $next     = array_slice($splits, $i);

        $combine  = array_pop($previous) . $with . array_shift($next);
        $result[] = implode($replace, array_merge($previous, array($combine), $next));
    }
    return $result;
}
var_dump(getSingleReplaceCombinations("a", "x", "aba")); // result as mentioned above
?>
```

It may not be the best in performance, but it works.

up
down
-2

Oyedele Hammed Horlah - itz dot harmid at gmail dot com ¶
5 years ago

this is a simple function to replace all newlines to
 tags.

```
\r\n - windows line break
\n - linux line break
\r - mac line break

<?php
function nl_to_br($str) {
return str_replace(array("\r\n","\n","\r"), "<br/>", $str);
}
echo nl_to_br('Hello world\n I am Oyedele Hammed Horlah'); // => Hello World <br> I am Oyedele Hammed Horlah
?>
```

Enjoy

up
down
-2
hyperzlib at outlook dot com ¶
2 years ago
Use str_replace to remove all dashes but one from the string '-aaa----b-c-----d--e---f' (resulting is: '-aaa-b-c-d-e-f')

```
<?php
$challenge = '-aaa----b-c-----d--e---f';
do $challenge = str_replace('--', '-', $challenge, $count); while($count);
echo $challenge;
?>
```

up
down
-8
Anonymous ¶
9 years ago
@moostende at gmail dot com
If you want to remove all dashes but one from the string '-aaa----b-c-----d--e---f' resulting in '-aaa-b-c-d-e-f', you CAN use str_replace !

```
<?php
function foo($str)
{
    do {
        $str = str_replace("--", "-", $str, $count);
    } while ($count > 0);
    return $str;
}
echo foo("-aaa----b-c-----d--e---f");
?>
```

This outputs the following:
-aaa-b-c-d-e-f

up
down
-6
iskus1981 at gmail dot com ¶
2 years ago

```
bracesChecker($str) {
    while($str) {
        $str = str_replace(['()','[]','{}'], '', $str, $count);
        if (!$str) return true;
        if (!$count && $str) return false;
    }
}
$str = '[}]{(){}[][}()';
echo bracesChecker($str);
//false
$str = '([]{})[{}[]{[]}([{[]}])';
echo bracesChecker ($str);
//true
```

+ 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
- str_starts_with
- str_word_count
- strcasecmp
- strchr
- strcmp
- strcoll
- strcspn
- strip_tags
- stripcslashes
- stripos
- stripslashes
- stristr
- strlen