Keyboard Shortcuts

?
: This help

j
: Next menu item

k
: Previous menu item

g p
: Previous man page

g n
: Next man page

G
: Scroll to bottom

g g
: Scroll to top

g h
: Goto homepage

g s
: Goto search
(current page)

/
: Focus search box

stripslashes »

« stripcslashes

- PHP Manual
- Function Reference
- Text Processing
- Strings
- String Functions

Change language: English

Edit Report a Bug

stripos

(PHP 5, PHP 7)

stripos — Find the position of the first occurrence of a case-insensitive substring in a string

Description ¶

mixed **stripos** (string `$haystack` , string `$needle` [, int `$offset` = 0])

Find the numeric position of the first occurrence of `needle` in the `haystack` string.

Unlike the strpos(), **stripos()** is case-insensitive.

Parameters ¶

`haystack`

The string to search in.

`needle`

Note that the `needle` may be a string of one or more characters.

If `needle` is not a string, it is converted to an integer and applied as the ordinal value of a character.

`offset`

If specified, search will start this number of characters counted from the beginning of the string. If the offset is negative, the search will start this number of characters counted from the end of the string.

Return Values ¶

Returns the position of where the needle exists relative to the beginnning of the `haystack` string (independent of offset). Also note that string positions start at 0, and not 1.

Returns **`FALSE`** if the needle was not found.

Warning

This function may return Boolean **`FALSE`**, but may also return a non-Boolean value which evaluates to **`FALSE`**. Please read the section on Booleans for more information. Use the === operator for testing the return value of this function.

Changelog ¶

Version	Description
7.1.0	Support for negative `offsets` has been added.

Examples ¶

Example #1 stripos() examples

```
<?php
$findme    = 'a';
$mystring1 = 'xyz';
$mystring2 = 'ABC';

$pos1 = stripos($mystring1, $findme);
$pos2 = stripos($mystring2, $findme);

// Nope, 'a' is certainly not in 'xyz'
if ($pos1 === false) {
    echo "The string '$findme' was not found in the string '$mystring1'";
}

// Note our use of ===.  Simply == would not work as expected
// because the position of 'a' is the 0th (first) character.
if ($pos2 !== false) {
    echo "We found '$findme' in '$mystring2' at position $pos2";
}
?>
```

Notes ¶

> **Note**: This function is binary-safe.

See Also ¶

- mb_stripos() - Finds position of first occurrence of a string within another, case insensitive
- strpos() - Find the position of the first occurrence of a substring in a string
- strrpos() - Find the position of the last occurrence of a substring in a string
- strripos() - Find the position of the last occurrence of a case-insensitive substring in a string
- stristr() - Case-insensitive strstr
- substr() - Return part of a string
- str_ireplace() - Case-insensitive version of str_replace.

add a note

User Contributed Notes 6 notes

up
down
25
emperorshishire at gmail dot com ¶
8 years ago

I found myself needing to find the first position of multiple needles in one haystack. So I wrote this little function:

```
<?php
function multineedle_stripos($haystack, $needles, $offset=0) {
    foreach($needles as $needle) {
```

```
        $found[$needle] = stripos($haystack, $needle, $offset);
    }
    return $found;
}

// It works as such:
$haystack = "The quick brown fox jumps over the lazy dog.";
$needle = array("fox", "dog", ".", "duck")
var_dump(multineedle_stripos($haystack, $needle));
/* Output:
   array(3) {
     ["fox"]=>
     int(16)
     ["dog"]=>
     int(40)
     ["."]=>
     int(43)
     ["duck"]=>
     bool(false)
   }
*/
?>
```

up
down
3
spam at kleppinger dot com ¶
2 years ago

Regarding the function by spam at wikicms dot org

It is very bad practice to use the same function name as an existing php function but have a different output format. Someone maintaining the code in the future is likely to be very confused by this. It will also be hard to eradicate from a codebase because the naming is identical so each use of stripos() would have to be analyzed to see how it is expecting the output format (bool or number/bool).

Calling it string_found() or something like that would make a lot more sense for long-term use.

up
down
-1
Ian Macdonald ¶
2 years ago

Regarding the === note, it might be worth clarifying that the correct tests for a binary found/not found condition are !==false to detect found, and ===false to detect not found.

up
down
-9
steve at opilo dot net ¶
4 years ago

A handy function if you need to adjust layout based on whether or not a string contains descending letters:

```
<?php function containsDescenders($text) {
    $descenders = array("g","j","p","q","y");
    foreach ($descenders as $letter) {
        if (stripos($text,$letter) !== false) {
            return true;
        }
    }
```

```
    return false;
} ?>
```

up
down
-10
grf at post dot cz ¶
10 years ago

this would to work with any language, i hope.
tested on czech (eastern europe) lang.

```
<?php
/**************************************
*    SAFE HIGHLIGHT
**************************************/
/**
* function finds and encase every string in a $needleArr array with
* strings $shearLft (from the left side) and $shearRgt (guess from which
* side).
* already encased needles are IGNORED for any other step, so order
* of needles in $needleArr is pretty important.
*
* function is searching needles in case-insensitive mode,
* but case in the subject is saved.
*
* can you do it better? so, do it.
*
* @param array $needleArr array of needles
* @param string $shearLft left shear
* @param string $shearRgt right shear
* @param string $subject subject
* @param string $encoding encoding ('utf-8' is default)
*
* @author griffin
*/
function safeHighlight( $needleArr, $shearLft, $shearRgt, $subject, $encoding = 'utf-8')
{

    // encoding
    $e = $encoding;

    // oh, no needles
    if( !is_array( $needleArr))
        return $subject;

    // empty keys throw-off, only unique, reindex
    $nA = array_values(
            array_unique(
                array_diff( $needleArr, array(''))
            )
        );

    // needle count
    if( !($nC = count( $nA)))
        return $subject; // nothing to hl

    // shear length
    if( !(($rLL = mb_strlen( $rL = $shearLft, $e))
    + ($rRL = mb_strlen( $rR = $shearRgt, $e))))
```

```
        return $subject; // no shears

    // subject length
    if( !($sL = mb_strlen( $s = $subject, $e)))
        return null; // empty subject

    // subject in lowercase (we need to aviod
    // using mb_stripos due to PHP version)
    $sW = mb_strtolower( $s, $e);

    // masking ~ 0=not changed, 1=changed
    $m = str_repeat( '0', $sL);

    // loop for each needle
    for( $n=0; $n<$nC; $n++)
    {

        // needle string loWercase
        $nW = mb_strtolower( $nA[ $n], $e);

        $o = 0; // offset
        $nL = mb_strlen( $nW, $e); // needle length

        // search needle
        while( false !== ($p = mb_strpos( $sW, $nW, $o, $e)))
        {
            // oh hurrey, needle found on $p position

            // is founded needle already modified? (in full-length)
            for( $q=$p; $q<($p+$nL); $q++)
                if( $m[ $q])
                {
                    // ai, caramba. already modified, jump over
                    $o+= $nL;

                    // continue for while() loop - not for for() loop!
                    continue 2;
                }

            // explode subject and mask into three parts
            // partA|needle|partB
            $sE[0] = mb_substr( $s, 0, $p, $e);
            $sE[1] = mb_substr( $s, $p, $nL, $e);
            $sE[2] = mb_substr( $s, $p+$nL, $sL-$p-$nL, $e);

            // mask
            // partA|partB (needle not needed)
            $mE[0] = mb_substr( $m, 0, $p, $e);
            $mE[1] = mb_substr( $m, $p+$nL, $sL-$p-$nL, $e);

            // apply shears
            $sE[1] = $rL.$sE[1].$rR;

            // update sunject length
            $sL+= $rLL + $rRL;

            // update mask
            $m = $mE[0] . str_repeat( '1', $rLL + $nL + $rRL) . $mE[1];
```

```
            // implode into a subject
            $s = implode( $sE);

            // update lowercase subject
            $sW = mb_strtolower( $s, $e);

            // increase offset
            $o+= $rLL + $nL + $rRL;

            // end of string reached
            if( $o>=$sL)
                break;

        } // while()

    } // for( $n=0; $n<$nC; $n++)

    // oouu yeaaa, kick the subject out of the function
    return $s;

} // function safeHighlight()
/**************************************
*    END: SAFE HIGHLIGHT
**************************************/
?>
```

up
down
-13
spam at wikicms dot org ¶
4 years ago

If you like using ternary operator, I wrote simple example how to use stripos function.
Also, in my example I add "How to use namespaces" for wide knowledges for newbies.

```
<?php
namespace My;

//You can be free using core functions in your NameSpaces (My)
function stripos($haystack, $needle) {
    //To call core function (from global NS) you should add backslash only - \func
    return (FALSE === \stripos($haystack, $needle)) ? FALSE : TRUE;
}

var_dump(stripos($haystack = 'John knows English language.', $needle = 'john')); //TRUE
var_dump(stripos($haystack = 'Sara knows English language too.', $needle = 'john')); //FALSE
?>
```

add a note

- String Functions
  - addcslashes
  - addslashes
  - bin2hex
  - chop
  - chr
  - chunk_split
  - convert_cyr_string
  - convert_uudecode
  - convert_uuencode

- count_chars
- crc32
- crypt
- echo
- explode
- fprintf
- get_html_translation_table
- hebrev
- hebrevc
- 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_getcsv
- str_ireplace
- str_pad
- str_repeat
- str_replace
- str_rot13
- str_shuffle
- str_split
- str_word_count
- strcasecmp
- strchr
- strcmp
- strcoll
- strcspn
- strip_tags
- stripcslashes