Posted by Ubaidullah in PHP Tutorials and Resources | 0 Comments

20 Handy PHP String Functions

Tweet

0

+1

PHP is the widely used open source language especially for developing web applications. Strings in PHP play very important role. In this tutorial, You will find about 20 handy and most used PHP functions.

1. addcslashes()

This string function returns a string also with backslashes before characters and characters are listed as parameter.

Syntax:

```
<?php addcslashes($str, characters); ?>
```

Example:

```
<?php
$mystr = "How are you.";
echo addcslashes($mystr, 'r');
?>
```

Output:

```
How a\re you.
```

2. convert_uudecode()

This string function decodes a string that is uuencoded (in simple words it decodes a string which is encrypted by some security methods).

Syntax:

```
<?php
convert_uudecode($encrypted_string);
```

```
?>
```

Example:

```
<?php
$encrypted_string = "+2&]W($%R92!9;W4` `";

echo convert_uudecode($encrypted_string);
?>
```

Output:

```
How Are You
```

3. convert_uuencode()

This string function encodes a string.

Syntax:

```
<?php
convert_uuencode($encrypted_string);
?>
```

Example:

```
<?php
$encrypted_string = "How Are You";

echo convert_uuencode($encrypted_string);
?>
```

Output:

```
+2&]W($%R92!9;W4` `
```

4. count_chars()

This string function returns how many times a character (ASCII) appears in a string depending on the modes we specified.

By default, mode is 0 that returns an array with Key (ASCII Value) and Value (Number of Occurrences.

Mode 1 returns an array with same functionality as Mode 2, but it only lists appearance of characters which are greater than zero.

Mode 2 returns occurrence which are equal to zero.

Mode 3 returns string with difference characters used.

Mode 4 generates string of all unused characters.

Syntax:

```
count_chars($str, mode);
```

Example:

```
<?php
$mystr = "Hello Dear, How are you";

print_r(count_chars($mystr, 0));
print_r(count_chars($mystr, 1));
print_r(count_chars($mystr, 2));

echo count_chars($mystr, 3);
echo count_chars($mystr, 4);

?>
```

5. echo()

This function helps to output more than one strings. Parameters in this function are optional.

Syntax:

```
echo(string)
```

6. explode()

explode() explodes or break a string into an array. We can easily understand the functionality of this function by the name “explode”. explode() has three parameters, first 2 are required and third is optional.

Syntax:

```
explode(string_separator, $str, limit);
```

Example:

```
<?php

$mystr = "Hello Dear, How are you";

print_r(explode(" ", $mystr));

?>
```

Output:

```
Array ( [0] => Hello [1] => Dear, [2] => How [3] => are [4] => you )
```

7. implode()

implode() is a reverse of explode(), means it joins an array into a sring. It has two parameters, first one is optional and second one is required.

Syntax:

```
imlode(string_separator, array);
```

Example:

```
<?php

$arr = array("Hello", "Dear", "How", "Are", "You");

echo implode(" ", $arr);

?>
```

Output:

```
Hello Dear How Are You
```

8. md5()

md5() function calculates Message-Digest Algorithm (md5) hash of a string. It has two parameters, first one is required and second one is optional which was included in PHP 5.0.

Syntax:

```
md5($str, raw);
```

Example:

```
<?php

$mystr = "How Are You";

echo md5($mystr);

?>
```

Output:

```
9e227bb366c119c7f27a7115f0136f42
```

9. str_replace()

If you want to replace some specified characters with other characters in PHP, str_replace() is the best choice. It has many rules like:

1. If we want to search a string in an array, it returns as an array.
2. Find and Replace is performed with each array element as we want to search a string in an array.
3. If we are finding an array and also replacing with it with an array, an empty string will be used as replace.
4. In case, replace is a string and find is an array, then it uses replace sting as find value.

Syntax:

```
str_replace(find, replace, string, count);
```

Example:

```
<?php
echo str_replace("How", "Dear", "How Di");
?>
```

Output:

```
How Dear
```

10. str_split()

str_split() works like explode() in some manner, it is used for splitting a string into an array.

Syntax:

```
str_split($mystring, length);
```

Example:

```
<?php
print_r(str_split("What"));
?>
```

Output:

```
Array
(
[0] => W
[1] => h
[2] => a
[3] => T
)
```

11. str_word_count()

This function is used for counting the words in a string.

Syntax:

```
str_word_count(string, return, char);
```

Example:

```
<?php
echo str_word_count("What are you doing");
?>
```

Output:

```
4
```

12. strcmp()

This case-sensitive string function is used for comparing two strings. It returns “0″ if both strings are equal, “<0″ if first string is less than second string and “>0″ if first string is greater than second string.

Syntax:

```
strcmp(first string, second string);
```

Example:

```
<?php
echo strcmp("What Are You Doing", "What I Should Do");
?>
```

Output:

```
-1
```

13. strlen()

strlen() calculates the length of string based on extra spaces and characters used in a string.

Syntax:

```
strlen($str);
```

Example:

```
<?php
echo strlen("Oh, No!");
?>
```

Output:

```
7
```

14. strrpos()

strrpos() is used to find the position of the last occurrence string inside another string.

Syntax:

```
strrpos(strong, find, start)
```

Example:

```
<?php
echo strrpos("Where are you going?", "go");
?>
[code]

<strong>Output:</strong>

[code type="php"]14
```

15. strstr()

strstr() finds the first occurrence of a string in a string, if found it returns at the matching point else it returns 0.

Syntax:

```
strstr();
```

Example:

```
<?php
echo strstr('Hi What', 'Hi');
?>
```

Output:

```
Hi What
```

16. strtolower()

This function converts a string to lowercase (It converts only letters from A-Z or a-z).

Syntax:

```
strtolower($str);
```

Example:

```
<?php
echo strtolower("Now WhaT are");
?>
```

Output:

```
now what are
```

17. strtoupper()

This function converts a string to uppercase letters.

Syntax:

```
strtoupper($str)
```

Example:

```
<?php
echo strtoupper("Yes");
?>
```

Output:

```
YES
```

18. substr()

This function split up a part from a string according to given criteria specified in the parameters.

Syntax:

```
substr($str, start, length)
```

Example:

```
<?php
echo substr("Where I Can Find You", 7);
?>
```

Output:

```
Can Find You
```

19. trim()

This function erases predefined characters and whitespaces from every side of a string. Predefined characters are like: \0, \t, \n, \xob, \r etc.

Syntax:

```
trim($str, chars)
```

Example:

```
<?php
$mystr = "Hey dude, I'm going with you";
echo trim($mystr);
?>
```

20. wordwrap()

If you want to cut a string after it reaches to a specific length, wordwrap() is the best choice for achieving this.

Syntax:

```
wordwrap($str, width, break, cut)
```

Example:

```
<?php

$str = "Check the example: Supercalifragulistic";

echo wordwrap($str,9);
?>
```

Output:

Browser will display out like this

```
Check the example: Supercalifragulistic
```

But after checking the page source, output will be something like below:

```
Check the
example:
Supercalifragulistic
```