# PHP

# PHP String Functions

# PHP String Functions [1]

**Refer Part I String handling functions also**

# str_ireplace() [1]

- The str_ireplace() function replaces some characters with some other characters in a string.
- This function works by the following rules:
  - If the string to be searched is an array, it returns an array
  - If the string to be searched is an array, find and replace is performed with every array element
  - If both find and replace are arrays, and replace has fewer elements than find, an empty string will be used as replace
  - If find is an array and replace is a string, the replace string will be used for every find value
- **Note:** This function is case-insensitive. Use the str_replace() function to perform a case-sensitive search.
- **Example 1**

# str_ireplace() [1]

## Syntax

```
str_ireplace(find,replace,string,count)
```

| Parameter | Description |
|---|---|
| *find* | Required. Specifies the value to find |
| *replace* | Required. Specifies the value to replace the value in *find* |
| *string* | Required. Specifies the string to be searched |
| *count* | Optional. A variable that counts the number of replacements |

# str_pad() [1]

- The str_pad() function pads a string to a new length. **Example 2**

## Syntax

```
str_pad(string,length,pad_string,pad_type)
```

| Parameter | Description |
|---|---|
| *string* | Required. Specifies the string to pad |
| *length* | Required. Specifies the new string length. If this value is less than the original length of the string, nothing will be done |
| *pad_string* | Optional. Specifies the string to use for padding. Default is whitespace |
| *pad_type* | Optional. Specifies what side to pad the string.<br><br>Possible values:<br><br>• STR_PAD_BOTH - Pad to both sides of the string. If not an even number, the right side gets the extra padding<br>• STR_PAD_LEFT - Pad to the left side of the string<br>• STR_PAD_RIGHT - Pad to the right side of the string. This is default |

# str_repeat() [1]

- **Example 3**

The str_repeat() function repeats a string a specified number of times.

## Syntax

```
str_repeat(string,repeat)
```

| Parameter | Description |
|---|---|
| *string* | Required. Specifies the string to repeat |
| *repeat* | Required. Specifies the number of times the string will be repeated. Must be greater or equal to 0 |

# str_shuffle()  [1]

- Randomly shuffles all characters in a string. **Example 3**

## Syntax

```
str_shuffle(string)
```

| Parameter | Description |
| --- | --- |
| *string* | Required. Specifies the string to shuffle |

# str_split()[1]

- Splits a string into an array. **Example 4**

## Syntax

```
str_split(string,length)
```

| Parameter | Description |
|---|---|
| *string* | Required. Specifies the string to split |
| *length* | Optional. Specifies the length of each array element. Default is 1 |

## Technical Details

| | |
|---|---|
| **Return Value:** | If length is less than 1, the str_split() function will return FALSE. If length is larger than the length of string, the entire string will be returned as the only element of the array. |
| **PHP Version:** | 5+ |

# strcasecmp()[1]

- **Example 4**

The strcasecmp() function compares two strings.

**Tip:** The strcasecmp() function is binary-safe and case-insensitive.

**Tip:** This function is similar to the strncasecmp() function, with the difference that you can specify the number of characters from each string to be used in the comparison with strncasecmp().

## Syntax

```
strcasecmp(string1,string2)
```

**Return Value:** This function returns:

- 0 - if the two strings are equal
- <0 - if string1 is less than string2
- >0 - if string1 is greater than string2

**PHP Version:** 4+

| Parameter | Description |
|---|---|
| *string1* | Required. Specifies the first string to compare |
| *string2* | Required. Specifies the second string to compare |

# strcmp()[1]

- **Example 5**

The strcmp() function compares two strings.

**Note:** The strcmp() function is binary-safe and case-sensitive.

**Tip:** This function is similar to the strncmp() function, with the difference that you can specify the number of characters from each string to be used in the comparison with strncmp().

## Syntax

```
strcmp(string1,string2)
```

**Return Value:** This function returns:

- 0 - if the two strings are equal
- <0 - if string1 is less than string2
- >0 - if string1 is greater than string2

| Parameter | Description |
|---|---|
| *string1* | Required. Specifies the first string to compare |
| *string2* | Required. Specifies the second string to compare |

# strchr()[1]

- **Example 5**

The strchr() function searches for the first occurrence of a string inside another string.

This function is an alias of the strstr() function.

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

**Note:** This function is case-sensitive. For a case-insensitive search, use stristr() function.

## Syntax

```
strchr(string,search,before_search);
```

| Parameter | Description |
|---|---|
| *string* | Required. Specifies the string to search |
| *search* | Required. Specifies the string to search for. If this parameter is a number, it will search for the character matching the ASCII value of the number |
| *before_search* | Optional. A boolean value whose default is "false". If set to "true", it returns the part of the string before the first occurrence of the *search* parameter. |

# strip_tags()[1]

- **Example 6**

The strip_tags() function strips a string from HTML, XML, and PHP tags.

**Note:** HTML comments are always stripped. This cannot be changed with the allow parameter.

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

## Syntax

```
strip_tags(string,allow)
```

| Parameter | Description |
|---|---|
| *string* | Required. Specifies the string to check |
| *allow* | Optional. Specifies allowable tags. These tags will not be removed |

# stripos()[1]

- **Example 6**

## Definition and Usage

The stripos() function finds the position of the first occurrence of a string inside another string.

**Note:** The stripos() function is case-insensitive.

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

Related functions:

- strripos() - Finds the position of the last occurrence of a string inside another string (case-insensitive)
- strpos() - Finds the position of the first occurrence of a string inside another string (case-sensitive)
- strrpos() - Finds the position of the last occurrence of a string inside another string (case-sensitive)

# stripos()[1]

- **Example 6**

## Syntax

```
stripos(string,find,start)
```

| Parameter | Description |
|---|---|
| *string* | Required. Specifies the string to search |
| *find* | Required. Specifies the string to find |
| *start* | Optional. Specifies where to begin the search |

## Technical Details

| | |
|---|---|
| **Return Value:** | Returns the position of the first occurrence of a string inside another string, or FALSE if the string is not found. **Note:** String positions start at 0, and not 1. |
| **PHP Version:** | 5+ |

# strripos()[1]

- The strripos() function finds the position of the last occurrence of a string inside another string.
- **Note:** The strripos() function is case-insensitive.
- **Example 6**

```
strripos(string,find,start)
```

| Parameter | Description |
|---|---|
| *string* | Required. Specifies the string to search |
| *find* | Required. Specifies the string to find |
| *start* | Optional. Specifies where to begin the search |

# strrpos()[1]

- The strrpos() function finds the position of the last occurrence of a string inside another string.
- **Note:** The strrpos() function is case-sensitive.
- **Example 6**

## Syntax

```
strrpos(string,find,start)
```

| Parameter | Description |
|---|---|
| *string* | Required. Specifies the string to search |
| *find* | Required. Specifies the string to find |
| *start* | Optional. Specifies where to begin the search |

# substr() [1]

- The substr() function returns a part of a string.
- **Note:** If the start parameter is a negative number and length is less than or equal to start, length becomes 0.
- **Example 7**

```
substr(string,start,length)
```

| Parameter | Description |
|---|---|
| *string* | Required. Specifies the string to return a part of |
| *start* | Required. Specifies where to start in the string<br>• A positive number - Start at a specified position in the string<br>• A negative number - Start at a specified position from the end of the string<br>• 0 - Start at the first character in string |
| *length* | Optional. Specifies the length of the returned string. Default is to the end of the string.<br>• A positive number - The length to be returned from the start parameter<br>• Negative number - The length to be returned from the end of the string |

# substr_compare() [1]

- The substr_compare() function compares two strings from a specified start position.
- **Tip:** This function is binary-safe and optionally case-sensitive.
- **Example 8**

# substr_compare() [1]

```
substr_compare(string1,string2,startpos,length,case)
```

| Parameter | Description |
|---|---|
| *string1* | Required. Specifies the first string to compare |
| *string2* | Required. Specifies the second string to compare |
| *startpos* | Required. Specifies where to start comparing in string1. If negative, it starts counting from the end of the string |
| *length* | Optional. Specifies how much of string1 to compare |
| *case* | Optional. A boolean value that specifies whether or not to perform a case-sensitive compare:<br>• FALSE - Default. Case-sensitive<br>• TRUE - Case-insensitive |

**Return Value:** This function returns:

- 0 - if the two strings are equal
- <0 - if string1 (from startpos) is less than string2
- >0 - if string1 (from startpos) is greater than string2

If length is equal or greater than length of string1, this function returns FALSE.

# substr_count() [1]

- The substr_count() function counts the number of times a substring occurs in a string.
- **Note:** The substring is case-sensitive.
- **Note:** This function does not count overlapped substrings.
- **Note:** This function generates a warning if the start parameter plus the length parameter is greater than the string length.
- **Example 9**

# substr_count() [1]

```
substr_count(string,substring,start,length)
```

| Parameter | Description |
| --- | --- |
| *string* | Required. Specifies the string to check |
| *substring* | Required. Specifies the string to search for |
| *start* | Optional. Specifies where in string to start searching |
| *length* | Optional. Specifies the length of the search |

# substr_replace() [1]

- The substr_replace() function replaces a part of a string with another string.
- **Note:** If the start parameter is a negative number and length is less than or equal to start, length becomes 0.
- **Note:** This function is binary-safe.
- **Example 10**

# substr_replace() [1]

`substr_replace(string,replacement,start,length)`

| Parameter | Description |
| --- | --- |
| *string* | Required. Specifies the string to check |
| *replacement* | Required. Specifies the string to insert |
| *start* | Required. Specifies where to start replacing in the string<br>• A positive number - Start replacing at the specified position in the string<br>• Negative number - Start replacing at the specified position from the end of the string<br>• 0 - Start replacing at the first character in the string |
| *length* | Optional. Specifies how many characters should be replaced. Default is the same length as the string.<br>• A positive number - The length of string to be replaced<br>• A negative number - How many characters should be left at end of string after replacing<br>• 0 - Insert instead of replace |

# wordwrap() [1]

- **Example 11**

The wordwrap() function wraps a string into new lines when it reaches a specific length.

**Note:** This function may leave white spaces at the beginning of a line.

## Syntax

```
wordwrap(string,width,break,cut)
```

| Parameter | Description |
|---|---|
| *string* | Required. Specifies the string to break up into lines |
| *width* | Optional. Specifies the maximum line width. Default is 75 |
| *break* | Optional. Specifies the characters to use as break. Default is "\n" |
| *cut* | Optional. Specifies whether words longer than the specified width should be wrapped:<br>• FALSE - Default. No-wrap<br>• TRUE - Wrap |

# strtolower() [1]

- The strtolower() function converts a string to lowercase.
- **Note:** This function is binary-safe.

- Example:
- <?php
  echo strtolower
  ("Hello WORLD.");
?>
- Output:

  hello world.

```
strtolower(string)
```

| Parameter | Description |
|---|---|
| *string* | Required. Specifies the string to convert |

## Technical Details

| | |
|---|---|
| **Return Value:** | Returns the the lowercased string |
| **PHP Version:** | 4+ |

# strtoupper() [1]

- The strtoupper() function converts a string to uppercase.
- **Note:** This function is binary-safe.

- Example:
- <?php
  echo strtoupper
  ("Hello WORLD!");
- ?>
- Output:
  HELLO WORLD!

```
strtoupper(string)
```

| Parameter | Description |
|---|---|
| *string* | Required. Specifies the string to convert |

# trim() [1]

- The trim() function removes whitespace and other predefined characters from both sides of a string.
- Related functions:
  - ltrim() - Removes whitespace or other predefined characters from the left side of a string
  - rtrim() - Removes whitespace or other predefined characters from the right side of a string

- **Example 12**

# trim() [1]

`trim(string,charlist)`

| Parameter | Description |
| --- | --- |
| *string* | Required. Specifies the string to check |
| *charlist* | Optional. Specifies which characters to remove from the string. If omitted, all of the following characters are removed:<br>• "\0" - NULL<br>• "\t" - tab<br>• "\n" - new line<br>• "\x0B" - vertical tab<br>• "\r" - carriage return<br>• " " - ordinary white space |

# strstr() [1]

- The strstr() function searches for the first occurrence of a string inside another string.
- **Note:** This function is binary-safe.
- **Note:** This function is case-sensitive. For a case-insensitive search, use stristr() function.
- **Example 12**

# strstr() [1]

```
strstr(string,search,before_search)
```

| Parameter | Description |
| --- | --- |
| *string* | Required. Specifies the string to search |
| *search* | Required. Specifies the string to search for. If this parameter is a number, it will search for the character matching the ASCII value of the number |
| *before_search* | Optional. A boolean value whose default is "false". If set to "true", it returns the part of the string before the first occurrence of the *search* parameter. |

# stristr() [1]

- The stristr() function searches for the first occurrence of a string inside another string.
- **Note:** This function is binary-safe.
- **Note:** This function is case-insensitive. For a case-sensitive search, use strstr() function.
- **Example 12**

# stristr() [1]

`stristr(string,search,before_search)`

| Parameter | Description |
|---|---|
| *string* | Required. Specifies the string to search |
| *search* | Required. Specifies the string to search for. If this parameter is a number, it will search for the character matching the ASCII value of the number |
| *before_search* | Optional. A boolean value whose default is "false". If set to "true", it returns the part of the string before the first occurrence of the *search* parameter. |

# ucfirst() [1]

- **Example 13**

The ucfirst() function converts the first character of a string to uppercase.

Related functions:

- lcfirst() - converts the first character of a string to lowercase
- ucwords() - converts the first character of each word in a string to uppercase
- strtoupper() - converts a string to uppercase
- strtolower() - converts a string to lowercase

## Syntax

```
ucfirst(string)
```

| Parameter | Description |
|---|---|
| *string* | Required. Specifies the string to convert |

# lcfirst() [1]

- **Example 13**

The lcfirst() function converts the first character of a string to lowercase.

Related functions:

- ucfirst() - converts the first character of a string to uppercase
- ucwords() - converts the first character of each word in a string to uppercase
- strtoupper() - converts a string to uppercase
- strtolower() - converts a string to lowercase

## Syntax

```
lcfirst(string)
```

| Parameter | Description |
|---|---|
| *string* | Required. Specifies the string to convert |

# ucwords() [1]

- **Example 13**

The ucwords() function converts the first character of each word in a string to uppercase.

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

Related functions:

- ucfirst() - converts the first character of a string to uppercase
- lcfirst() - converts the first character of a string to lowercase
- strtoupper() - converts a string to uppercase
- strtolower() - converts a string to lowercase

## Syntax

```
ucwords(string)
```

| Parameter | Description |
|---|---|
| *string* | Required. Specifies the string to convert |

# implode()

- **Example 14**

The implode() function returns a string from the elements of an array.

**Note:** The implode() function accept its parameters in either order. However, for consistency with explode(), you should use the documented order of arguments.

**Note:** The separator parameter of implode() is optional. However, it is recommended to always use two parameters for backwards compatibility.

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

## Syntax

```
implode(separator,array)
```

| Parameter | Description |
|---|---|
| *separator* | Optional. Specifies what to put between the array elements. Default is "" (an empty string) |
| *array* | Required. The array to join to a string |

# explode() [1]

- The explode() function breaks a string into an array.
- **Note:** The "separator" parameter cannot be an empty string.
- **Note:** This function is binary-safe.

- **Example 14**

# explode() [1]

```
explode(separator,string,limit)
```

| Parameter | Description |
| --- | --- |
| *separator* | Required. Specifies where to break the string |
| *string* | Required. The string to split |
| *limit* | Optional. Specifies the number of array elements to return.<br><br>Possible values:<br><br>• Greater than 0 - Returns an array with a maximum of *limit* element(s)<br>• Less than 0 - Returns an array except for the last *-limit* elements()<br>• 0 - Returns an array with one element |

# Binary Safe in PHP

- In PHP, Some functions are marked as binary safe functions. It means that the functions works correctly even when you pass binary data. Ex: A string containing non-ascii bytes, null bytes etc..
- To say more cleanly, A non binary safe function might be based on null terminated strings, When it sees any null character in the strings these functions ignores anything after it.

```
$str1="web";
$str2="webx00Development";

echo strcoll($str1,$str2); // gives 0, treats both strings are equal ( Non-binary safe functions )

echo strcmp($str1,str2); // gives less than 0, which means $str1 less than $str2 ( Binary safe function )
```

# References

1. https://www.w3schools.com/php/

Thank you….