**Strings in PHP**

Nowdocs are to single-quoted strings what heredocs are to double-quoted strings. A nowdoc is specified similarly to a heredoc, but no parsing is done inside a nowdoc. The construct is ideal for embedding PHP code or other large blocks of text without the need for escaping.

In other words:

$foo = 'bar';

$here = <<<HERE

I'm here , $foo !

HERE;

$now = <<<'NOW'

I'm now , $foo !

NOW;

$here is "I'm here , bar !", while $now is "I'm now , $foo !".

|  |  |
| --- | --- |
| **Function** | **Purpose** |
| **Addcslashes**  Syntax : string **addcslashes** (string, charlist);  Parameter :   * *string* –  This is a **Required** parameter. which needs to be escaped. * *charlist* – This is a **Required** parameter. It can have one characters or collection of characters. | **addcslashes function will add** backslash before the specified characters in **charlist** parameter for the **string**.  **Example:**  echo addcslashes("Hi from Subodh ","o");  **Output**  “Hi fr\om Sub\odh “.  echo addcslashes($strExample,'A..Z')."<br>";  echo addcslashes($strExample,'a..z')."<br>";  \Hi from \Subodh H\i \f\r\o\m S\u\b\o\d\h |
| **Addslashes**  Syntax : string **addslashes**(string); | addslashes function will add backslash ( \ ) before the predefined characters.  The predefined characters are:   * *single quote* (‘) * *double quote* (“) * *backslash* (\) * *NULL*   ***Example***  **echo** addslashes('Hi "from" Subodh.');  **output**  Hi \”from\” Subodh |
| bin2hex  string **bin2hex** (string); | **bin2hex function**will converts a string of ASCII characters to hexadecimal values. The string can be converted back using the pack() function. |
| chop or rtrim  string **chop** (string, charlist);  string –  String to check.  charlist – Optional. This is optional parameter, If any value is provided, then specified characters will be removed from right of a string.  The following characters are removed if the charlist parameter is empty:   * “\0” – NULL * “\t” – tab * “\n” – new line * “\r” – carriage return * ” ” – ordinary white space * “\x0B” – vertical tab | **chop function**will remove predefined characters or specified characters in charlist if any from the right end of a string. This function is an alias of- rtrim() function.  **example**  echo chop(“hello “);  **output**  hello  echo chop(“hello”,”0”);  output  hell |
| chr  string **chr** (ascii); | returns a character from the specified ASCII value.  The ASCII value can be specified in   * Decimal : Decimal values are defined by a leading 0x. * Octal : Octal values are defined by a leading 0 (Zero) * Hex : hex values are defined by a leading 0x.   **Example**  **echo chr(65)**  **echo chr(050)**  **echo chr(0x50)**  **output**  **A**  **(**  **P** |
| chunk\_split  string **chunk\_split(**string, chunklength, end**)** ;  **parameter**  string –  String to be chunked.  chunklength – This is Optional parameter,  String will be chunked in chunklength size. Chunk Default size is 76.  end – This is Optional parameter. End will defines what to be placed at the end of each chunk. Default is \r\n | returns smaller chunks(pieces) of a string in chunklength parameter.  **Example**  echo chunk\_split("Hi All",1,"..");  output  H..i.. ..A..l..l.. |
| ltrim  string **ltrim** (string, charlist);  string –  String to check.  charlist – Optional. This is optional parameter, If any value is provided, then specified characters will be removed from left of a string.  The following characters are removed if the charlist parameter is empty:   * “\0” – NULL * “\t” – tab * “\n” – new line * “\r” – carriage return * ” ” – ordinary white space * “\x0B” – vertical tab | **ltrim function**will remove predefined characters or specified characters in charlist if any from the left end of a string.  **Example**  echo ltrim(“ hello“);  output  hello  echo ltrim(“hello”,”h”);  output  ell0 |
| **join**  **join** ( separator, array );  Parameter :   * *separator*– Optional. This is an optional parameter. By default it is an empty string (“”). * *array*–  Collection of elements to join a string | join() function will glue(join) array elements in a string. join()  is an alias of the PHP implode() function. |
| **str\_ireplace**  **str\_ireplace** ( find, replace, string, count );  **str\_replace**  **str\_replace** ( find, replace, string, count );  Parameter :   * *find* – This is a **Required** parameter. It sets values to find. * *replace* – This is a **Required** parameter. It sets values to replace. * *string*– This is a **Required** parameter. It sets string to search. * *count* – This is an **Optional** parameter. If passed, this will be set to the number of replacements performed. | str\_ireplace() function will replace all occurrences of the search string with the replacement string. It is case – insensitive function.  str\_replace() function will replace all occurrences of the search string with the replacement string. It is case – sensitive function.  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   **Example**  $strExample = "Hi from subodh!";  echo str\_ireplace("Hi","Hello",$strExample);  output  Hello from subodh! |
| **str\_pad**  **str\_pad** (string, length, pad\_string, pad\_type );  Parameter :   * *string*– This is a **Required** parameter. It is the input string on which function will be performed. * *length* – This is a **Required** parameter. It sets value of new length of a input string parameter. if the value of *pad\_length* is negative, less than, or equal to the length of the input string, no padding takes place, and input will be returned. * *pad\_string*– This is a **Required** parameter. It sets string to be padded with input string. **Default** padded string is **whitespace**. * *pad\_type* – This is an **Optional** parameter. It can be Specifies what side to pad the string.   + ***STR\_PAD\_RIGHT***– Padding is done to the right side of the string. This is **default** value.   + ***STR\_PAD\_LEFT***– Padding is done to the left side of the string   + ***STR\_PAD\_BOTH***– Padding is done to both sides of the string. If not an even number, the right side gets the extra padding. | str\_pad() function will pad a string to a certain length with another string.  **Example**  $strExample = "Hi from Subodh.";  echo str\_pad($strExample,23,"!");  output  Hi from Subodh.!!!!!!!! |
| **str\_repeat**  **str\_repeat** (string, repeat );  Parameter :   * *string*– This is a **Required** parameter. It is the input string which will be repeated. * *repeat*– This is a **Required** parameter. It will set how many times a input string will be repeated. It should be greater than or equal to 0. If its value is set to 0, the function will return an empty string. | str\_repeat() function repeats a string a specified number of times..  **Example**  echo str\_repeat("!",10);  output  !!!!!!!!!! |
| **str\_split()**  **str\_split** ( string, length );   * *string*– This is a **Required** parameter. It is the input string on which function will be performed. * *length* – This is an **Optional** parameter. It sets the length of each array element. **Default length** value is **1**.  Output : This will return an array of string.  it also depends on the length parameter value if,   * length < 1, It will return **FALSE**. * length > larger than the length of string, the entire string will be returned as the only element of the array. | str\_split() function will breaks a string into an array.  Example  $strExample = "Hi from Subodh";  print\_r(str\_split($strExample));  Array ( [0] => H [1] => i [2] => [3] => f [4] => r [5] => o [6] => m [7] => [8] => S [9] => u [10] => b [11] => o [12] => d [13] =>h) |
| **str\_word\_count()**  **str\_word\_count** (string, returnformat, characterlist );   * *string*– This is a **Required** parameter. It is the input string on which function will be performed. * *returnformat* – This is a **Optional**parameter. It tells us about the return value of the str\_word\_count() function. Below are the values which can be pass in this parameter. **0** is the **default** value.   + ***0***– It is the **default value**. This will return number of words count in a string.   + ***1***– This will return an array with the words from the string.   + ***2***– returns an associative array, where the key is the numeric position of the word inside the string and the value is the actual word itself * *characterlist* – This is an **Optional** parameter. We can set special characters, which will be consider in word count.  Output : This will return a number or an array, depending upon the returnformat parameter value. | str\_word\_count() function will counts the number of words in a string.  Example  $strExample = "Hi from Subodh";  echo str\_word\_count($strExample)  output  3  $strExample = "Hi from Subodh";  echo str\_word\_count($strExample,1)  output  Array ( [0] => Hi [1] => from [2] => Subodh)  $strExample = "Hi from Subodh";  echo str\_word\_count($strExample,2)  output  Array ( [0] => Hi [3] => from [8] => Subodh ) |
| **strcasecmp()**  **strcasecmp** ( string1, string2 );   * *string1*– This is a **Required** parameter. It is the first string which will be compared. * *string2*– This is a **Required** parameter. It is the second string which will be compared.   Return values in this function are:   * 0 – if the two strings are equal. * < 0 – if string1 is less than string2. * > 0 – if string1 is greater than string2. | strcasecmp() function will compare two strings. (case insensitive match)  Example  echo strcasecmp("Hi from Subodh.","hI FROM SUBODH.");  output:  0 |
| **strchr()**  **strchr** ( string, search, before\_search );   * *string*– This is a **Required** parameter. This is the string which is to be searched. * *search*– This is a **Required** parameter. It contains the value to be seacrhed. If we pass this parameter as a number, then it will search for the character matching the ASCII value of the number. * *before\_search –* This is an **Optional** parameter. A Boolean value, default is set to “false”.  If set to “true”, it returns the part of the string before the first occurrence of the search parameter.  Output : It will returns below values:   * the portion of string after the first occurrence is found in the string * FALSE if the search is not found in the string. | strchr() function will find the first occurrence of search in a string. This function is an alias of the strstr() function.  Example:  echo strchr("Hi from Subodh.","from");  // case-sensitive comparison  Output  from Subodh.  echo strchr("Hi from Subodh.",102);// ASCII value of f is 102.  Output  from Subodh.  echo "Using before\_search 3rd parameter in the function<br />";  echo strchr("Hi from tutorialmines.","tutorialmines",true); // case-sensitive comparison  echo "<br/>";  echo strchr("Hi from tutorialmines.",116,true);// ASCII value of t is 116.  Using before\_search 3rd parameter in the function Hi from Hi from |
| **strtolower()**  **strtolower** ( string ); | strtolower() function will take string as input and converts all letter to lowercase letter, if that character is alphabetic.  Example:  $strExample = "An EXAMPLE of a strtolower() function.";  echo strtolower($strExample);  output  an example of a strtolower() function. |
| **strtoupper()**  **strtoupper** ( string ); | Strtolower() function will take string as input and converts all letter to uppercase letter, if that character is alphabetic.  Example:  $strexample = "an example of a strtoupper() function.";  echo strtoupper($strexample);  Output  AN EXAMPLE OF A STRTOLOWER() FUNCTION. |
| * **strops() - The strpos() function 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() - Finds the position of the first occurrence of a string inside another string (case-insensitive)** * **strripos() - Finds the position of the last occurrence of a string inside another string (case-insensitive)**   **Parameter**  **String: Required. Specifies the string to search**  **Find: Required. Specifies the string to find**  **start : Optional. Specifies where to begin the search**  **Return value:**  **Position or FALSE** | Example  echo strrpos("I love php, I love php too!","php");  output  19  echo strpos("I love php, I love php too!","php");  output  7 |