|  |  |  |
| --- | --- | --- |
|  | **Basic String Functions** |  |
|  | This page mentions some basic string functions, and demonstrates them with code.  Topics on this page:  [Copying a substring in a string (substr)](http://www.trans4mind.com/personal_development/phpTutorial/strings.htm#o1substr)  [Finding the position of a substring in a string (strpos)](http://www.trans4mind.com/personal_development/phpTutorial/strings.htm#o2strpos)  [When strpos doesn't find the string](http://www.trans4mind.com/personal_development/phpTutorial/strings.htm#o3When_the_string_isn%27t_found)  [Boolean false zero with strpos](http://www.trans4mind.com/personal_development/phpTutorial/strings.htm#o4Boolean_false_and_zero)  [Finding the last occurrence of a stubstring in a string](http://www.trans4mind.com/personal_development/phpTutorial/strings.htm#o5FindingTheLastOccurrenceOfAStringInAString)  [strrpos for PHP5](http://www.trans4mind.com/personal_development/phpTutorial/strings.htm#o6strrpos)  [How to reverse a string (strrev)](http://www.trans4mind.com/personal_development/phpTutorial/strings.htm#o61HowToReverseAString)  [How to find the length of a string (strlen)](http://www.trans4mind.com/personal_development/phpTutorial/strings.htm#o62HowToFindTheLengthOfAString)  [How to find the last occurrence of a substring in a string (custom method)](http://www.trans4mind.com/personal_development/phpTutorial/strings.htm#o7HowToFindTheLastOccurrenceOfAStringInAString)  [Characters in a String](http://www.trans4mind.com/personal_development/phpTutorial/strings.htm#o8CharactersInStrings)  [Changing a Character to Character Code (ord) and vice versa (chr)](http://www.trans4mind.com/personal_development/phpTutorial/strings.htm#o9ChangingACharacterToCharacterCode%20%28ord%29)  [Adding HTML breaks automatically (nl2br)](http://www.trans4mind.com/personal_development/phpTutorial/strings.htm#o10AddingHTMLbreaksautomatically)  [Replace All Instances in a String (str\_replace)](http://www.trans4mind.com/personal_development/phpTutorial/strings.htm#o11ReplaceAllInstancesInAString)  [Reference](http://www.trans4mind.com/personal_development/phpTutorial/strings.htm#Refernce)  [Lowercase and Uppercase (strtolower, strtoupper, ucfirst, ucwords)](http://www.trans4mind.com/personal_development/phpTutorial/strings.htm#o12LowercaseUppercase)  [Replacing Part of a String at a Particular Position (substr\_replace)](http://www.trans4mind.com/personal_development/phpTutorial/strings.htm#o13Replacing%20Part%20of%20a%20String%20at%20a%20Particular%20Position%20%28substr_replace%29)  **substr**  In JavaScript there is a "substr", but the modern form is "subverting". In PHP the substr is illustrated below:  **$**string="0123456789";  echo "substr(\$string,3,4)=".substr(**$**string,3,4);  //result: substr($string,3,4)=3456  **substr**, in the example above, copies a string from $string, starting at position 3, and extracts a string with a length of 4. If the second number is omitted, then substr copies the string to the end.  If the string isn't found, then **substr** returns an empty string.  **strpos**  The function strpos finds the **first** position of a string within a string. If a number is specified, then strpos starts looking at that position. It finds the first occurrence of the string. For instance:  **$**string="cat and cat";  echo "strpos($string,\"cat\",2)=".strpos(**$**string,"cat",2);  //result: strpos(cat and cat,"cat,2")=8  Without the 2 in strpos(**$**string,"cat",2) strpos would start looking from the beginning. So  **$**string="cat and cat";  echo "strpos(\$string,\"cat\")=".strpos(**$**string,"cat");  //result: strpos($string,"cat")=0  Because a start number isn't specified, **strpos** begins at the beginning.  **When the string isn't found**  If the position isn't found, then **strpos** returns an empty string:  **$**string="cat and cat";  echo "strpos(\$string,\"dog\")=".strpos(**$**string,"dog");  //result: strpos($string,"dog")=  **Boolean False and Zero**  The following **fails** to correct this:  **$**string="cat and cat";  **if** (strpos(**$**string,"dog")>=0)  echo "if (strpos(\$string,\"dog\")>=0) strpos(\$string,\"dog\")=".strpos(**$**string,"dog");  //result: if (strpos($string,"dog")>=0) strpos($string,"dog")=  This returns the empty string, even though we have hoped it wouldn't! The function strpos returns a [Boolean](http://www.trans4mind.com/personal_development/phpTutorial/dataTypes.htm#Boolean) false when it fails to find the string, and this can sometimes be a zero! The following does work:  **if** (strpos(**$**string,"dog")>-1)  echo "if (strpos(\$string,\"dog\")>-1) strpos(\$string,\"dog\")=".strpos(**$**string,"dog");  **else**  echo "String not found with if (strpos(\$string,\"dog\")>-1).";  //result: String not found with if (strpos($string,"dog")>-1).  When using strpos, we might wish to use a check whether it has returned [Boolean false](http://www.trans4mind.com/personal_development/phpTutorial/dataTypes.htm#Boolean). We can do this as follows:  **if** (strpos(**$**string,"dog"))  echo "if (strpos(\$string,\"dog\"));strpos(\$string,\"dog\")=".strpos(**$**string,"dog");  **else**  echo "if (strpos(\$string,\"dog\"))strpos(\$string,\"dog\")= Couldn't find 'dog' in \$string";  //result: if (strpos($string,"dog"))strpos($string,"dog")= Couldn't find 'dog' in $string  echo "<br>";  In the above, we explicitly check for a Boolean false, and follow on from there. We could have used:  **if** (strpos(**$**string,"dog")!==false)  But the previous code is fine, and such a statement using "not equivalent", "**!==false**" is not necessary.  **Finding the last occurrence of a string within a string**  **strrpos**  Apparently there is no function in PHP4 which finds the last occurrence of a string within a string. **strrpos** doesn't do this! **strrpos** finds the last occurrence of a **character** in a string. If a string is entered, it will merrily function without error, but finds the last position of the first character in the string! For instance:  echo "strrpos(\"dandy\",\"dog\")=".strrpos("dandy","dog");  //result: strrpos("dandy","dog")=3  Whilst we might have expected to get a Boolean false (blank character), the program merrily gives us the wrong answer of 3. That is, it tells us that the last character "d" in "dandy" is at position 3. Because the word "dog" begins with "d", it uses that character.  **How to reverse a string (strrev)**  **$**string="PHP is Great!"; //define variable  **$**rev=strrev(**$**string); //make $rev the reversed string  echo "$string reversed is $rev"; //say some stuff to illustrate these  //Result: PHP is Great! reversed is !taerG si PHP  **How to find the length of a string (strlen)**  **$**string="PHP is Great!";  **$**length=strlen(**$**string); //put the length in the variable $length  echo " The length of $string is $length";  //Result: The length of PHP is Great! is 13  **How to find the last occurrence of a string in a string**  While there is **strripos** in PHP5, PH4 doesn't have a pre-built function. Consider the following  **$**string="01234567890123456789";  **$**item="3456";  **$**index=strpos(strrev(**$**string),strrev(**$**item));  **$**index=strlen(**$**string)-strlen(**$**item)-**$**index;  echo **$**index;  //result 13  In the string, $string, we want to find the last occurrence of $item, "3456". To do this, we reverse both strings using **strrev**(). Then we search for the string and find its position. Because the strings have been reversed, we need to compute the actual last position of the substring, $item, in the string, $string. To do this, we use the function **strlen**().  The following is an example of a function that finds the last occurrence of a string in a string:  **function** lastIndexOf(**$**string,**$**item)  **{**  **$**index=strpos(strrev(**$**string),strrev(**$**item));  **if** (**$**index)  **{**  **$**index=strlen(**$**string)-strlen(**$**item)-**$**index;  **return** **$**index;  **}**  **else**  **return** -1;  **}**  **$**string="01234567890123456789";  **$**item="3456";  echo lastIndexOf(**$**string,**$**item);  //result 13  echo "<br>";  **$**item="elephant";  echo lastIndexOf(**$**string,**$**item);  //result: -1  When the string is found, it returns a number. Otherwise, it returns -1. There are several ways to write such a function, and using strrev() is one way.  Characters in Strings  You can read a character at a given position in a string in this way:  **$**str="Hello";  **$**char=**$**str**{**0**}**; //get character at 0  echo **$**char; //show character at 0  //result: H  We assign "Hello" to $str. We then read the character at zero, using curly brackets: $str{0}. The result is, as expected, "H".  Next we write a character to the string at position 0:  echo"<br>";  **$**str**{**0**}**="C"; //write C in the first position  echo **$**str;  //result: Cello  By assigning "C" to the first position in the string, we convert "Hello" to "Cello".  What happens when we write an empty character to a string?  **$**len=strlen(**$**str);  **$**str**{**0**}**=""; //write an empty string at position zero  **$**len2=strlen(**$**str);  echo "The string is now: \"**$**str\", and it's length is now \"**$**len2 \"(was \"**$**len\")";  //result: The string is now: "ello", and it's length is now "5 "(was "5")  The result now becomes "ello". The length of the string hasn't changed, as expected.  echo"<br>";  echo **$**str**{**0**}**; //find out what is at position 0  //(empty position)  As expected, the character at position 0 is an empty character. That is, the place is still there, but is empty.  **Changing a character to character code and vice versa**  The following shows how to get the character code of a character, and, vice versa, the character from the character code.  **$**char=" ";  echo ord(**$**char);  //result 32  echo"<br>";  echo chr(68);  //result: D  **Adding HTML breaks automatically**  You can see how **nl2br** works by studying the next example.  **$**str="one  two  three"; //string is defined with line breaks  echo nl2br(**$**str); //the output, below, has HTML breaks  //added to line breaks  //one  //two  //three  nl2br adds HTML breaks to any line breaks in the string, to preserve its appearance in an HTML page.  **Replacing all instances in a string (str\_replace)**  In the next example, **str\_replace** replaces all instances of a with b:  echo str\_replace("a","b","abracadabra");  //result: bbrbcbdbbrb  **str\_replace** can also be used with arrays:  **$**a=array("day","night","morning");  **$**b=array("Day","Night","Morning");  echo str\_replace(**$**a,**$**b,"It's a wonderful day in the morning every night");  //It's a wonderful Day in the Morning every Night  **Lowercase and Uppercase (strtolower, strtoupper, ucfirst, ucwords)**  The function, strtolower(), etc, are almost immediately recognised. Below are some examples of their use:  $str = "Oh to Be in England, Now That Spring is Here!<br>";  $str = strtolower($str);  **echo** $str; // oh to be in england, now that spring is here!  **$**str = "oh to be in england, now that spring is here!<br>";  $str = strtoupper($str);  **echo** $str; // OH TO BE IN ENGLAND, NOW THAT SPRING IS HERE!  $str = 'goodbye cruel world!<br>';  $str = ucfirst($str);  **echo** $str;  //Goodbye cruel world!  $str = 'GOODBYE CRUEL WORLD!<BR>';  $str = ucfirst($bar); // GOODBYE CRUEL WORLD  $str = ucfirst(strtolower($str)); // Goodbye cruel world!  $str = 'GOODBYE CRUEL WORLD!<BR>';  $str=strtolower($str);  $str = ucwords($str);  **echo** $str; // Goodbye Cruel World!  **Replacing Part of a String at a Particular Position (substr\_replace)**  You can replace part of a string at a given position with a substring using substr\_replace.  **$**string2="abcdefgh";  **if** (strpos(**$**string2,"abc")!==false)  **{**  **$**pos=strpos("abc",**$**string2);  **$**len=strlen("abc");  **$**newstring=substr\_replace(**$**string2,"ABC",**$**pos,**$**len);  **}**  **echo** "<br>";  print\_r(**$**string2);  **echo** "<br>";  print\_r(**$**newstring);  **echo** "<br>";  //result:  //abcdefgh //ABCdefgh  The $len in the below is optional:  **$**newstring=substr\_replace(**$**string2,"ABC",**$**pos,**$**len);  Here it is set to the length of the old part of the string, and so the old part is wiped out. If it isn't specified it defaults to the length of the string, so the old string is replaced by the new. |  |