Работа со строковыми функциями в PHP

Изучите методы и функции:

strtolower, strtoupper, ucfirst, lcfirst, ucwords, strlen, substr, str_replace, strtr,substr_replace, strpos, strrpos, strstr, explode, implode, str_split, trim, ltrim, rtrim, strrev, str_shuffle,number_format, str_repeat, htmlspecialchars, strip_tags, chr, ord, str_word_count, substr_count,count_chars, strchr, strrchr.

Функция strtolower

Функция **strtolower** преобразовывает строку в нижний регистр.

Данная функция неправильно работает с кириллицей. Используйте функцию **mb_strtolower** (она работает аналогичным образом, но корректно обрабатывает кириллицу).

Все функции работы с регистром: ucfirst, lcfirst, strtoupper, strtolower, ucwords.

Синтаксис

```
strtolower(строка);
```

Примеры

Пример

В данном примере функция преобразовала строку в нижний регистр:

```
<?php
        echo strtolower('ABC');
?>
```

Результат выполнения кода:

```
abc
```

Функция strtoupper

Функция **strtoupper** преобразовывает строку в верхний регистр.

Данная функция неправильно работает с кириллицей. Используйте функцию **mb_strtoupper** (она работает аналогичным образом, но корректно обрабатывает кириллицу).

Все функции работы с регистром: ucfirst, lcfirst, strtoupper, strtolower, ucwords.

Синтаксис

```
strtoupper(строка);
```

Примеры

Пример

В данном примере функция преобразовала строку в верхний регистр:

```
<?php
        echo strtoupper('abc');
?>
```

Результат выполнения кода:

```
ABC
```

Функция ucfirst

Функция **ucfirst** преобразует первый символ строки в верхний регистр. Не работает с кириллицей.

Все функции работы с регистром: ucfirst, lcfirst, strtoupper, strtolower, ucwords.

Синтаксис

```
ucfirst(строка);
```

Примеры

Пример

Давайте переведем первый символ строки в верхний регистр:

```
<?php
        echo ucfirst('hello');?>
```

Результат выполнения кода:

Hello

Функция lcfirst

Функция **lcfirst** преобразует первый символ строки в нижний регистр. Не работает с кириллицей.

Все функции работы с регистром: ucfirst, lcfirst, strtoupper, strtolower, ucwords.

Синтаксис

```
lcfirst(строка);
```

Примеры

Пример

Давайте переведем первый символ строки в нижний регистр:

```
<?php
        echo lcfirst('Hello');?>
```

Результат выполнения кода:

hello

Функция ucwords

Функция **ucwords** преобразует первый символ каждого слова в строке в верхний регистр. Не работает с кириллицей.

Все функции работы с регистром: ucfirst, lcfirst, strtoupper, strtolower, ucwords.

Синтаксис

```
ucwords(строка);
```

Примеры

Пример

В данном примере каждое слово строки преобразуется в верхний регистр:

```
<?php
        echo ucwords('hello world');
?>
```

Результат выполнения кода:

Hello World

Функция strlen

Функция **strlen** возвращает длину строки (количество символов в строке).

Данная функция неправильно работает с кириллицей. Используйте функцию **mb_strlen** (она работает аналогичным образом, но корректно обрабатывает кириллицу).

Синтаксис

```
strlen(строка);
```

Примеры

Пример

В данном примере функция вернет длину строки 'hello':

```
<?php
        echo strlen('hello');
?>
```

Результат выполнения кода:

```
5
```

Функция substr

Функция **substr** вырезает и возвращает подстроку из строки.

Сама строка при этом не изменяется. Нумерация символов строки начинается с нуля.

Второй параметр может быть отрицательным - в этом случае отсчет начнется с конца строки, при этом последний символ будет иметь номер -1.

Последний параметр можно не указывать - в этом случае отрезание произойдет до конца строки.

Данная функция неправильно работает с кириллицей. Используйте функцию **mb_substr** (она работает аналогичным образом, но корректно обрабатывает кириллицу).

См. также функцию substr_replace, которая вырезает часть строки и заменяет ее на другую.

Синтаксис

```
substr(строка, откуда, [сколько]);
```

Примеры

Пример

Давайте вырежем 3 символа из строки позиции 1 (со второго символа, так как первый имеет номер 0):

```
<?php
        echo substr('abcdef', 1, 3);
?>
```

Результат выполнения кода:

```
bcd
```

Пример

Давайте вырежем все символы до конца строки, начиная со второго (он имеет номер 1):

```
<?php
        echo substr('abcdef', 1);
?>
```

Результат выполнения кода:

```
bcdef
```

Пример

Давайте вырежем третий и второй символы с конца, для этого укажем начало вырезания -3 (это номер третьего символа с конца), а количество символов - 2:

```
<?php
        echo substr('abcdef', -3, 2);
?>
```

Результат выполнения кода:

```
de
```

Пример

Давайте вырежем 2 последних символа, для этого укажем позицию предпоследнего символа (это -2), а третий параметр не укажем - в этом случае обрезание будет до конца строки:

```
<?php
        echo substr('abcdef', -2);
```

```
?>
```

Результат выполнения кода:

ef

Пример

Давайте вырежем последний символ:

```
<?php
        echo substr('abcdef', -1);
?>
```

Результат выполнения кода:

f

Функция str_replace

Функция **str_replace** ищет в строке заданный текст и меняет его на другой.

Первым параметром функции принимает что меняем, а вторым - на что меняем. Это могут быть две строки или два массива.

Во втором случае соответствующие элементы одного массива заменятся на соответствующие элементы второго массива (см. примеры).

Есть также функция **str_ireplace**, которая делает тоже самое, но без учета регистра.

См. функцию strtr, которая также осуществляет поиск и замену.

Синтаксис

```
str_replace(что меняем, на что меняем, где меняем);
```

Примеры

Пример

В данном примере функция заменит все найденные буквы 'a' на '!':

```
<?php
        echo = str_replace('aaabbb', '!', 'abcabc');
?>
```

Результат выполнения кода:

!!!bbb

Пример

В данном примере функция заменит 'a' на 1, 'b' на 2, 'c' на 3:

```
<?php
        echo = str_replace(['a', 'b', 'c'], [1, 2, 3], 'abcabc');
?>
```

Результат выполнения кода:

123123

Функция strtr

Функция **strtr** осуществляет поиск и замену символов в строке.

См. функцию str_replace, которая также осуществляет поиск и замену.

Синтаксис

Функция имеет два варианта работы.

В первом варианте функция принимает массив замен: ключами служит то, что мы меняем, а значениями - на что будем менять:

```
strtr(где меняем, массив замен);
```

Во втором варианте функция одним параметром принимает строку с символами, которые будут заменены, а другим параметром строку с символами, на которые будет производится замена.

Соответствующие символы первой строки будут заменены на соответствующие символы второй строки:

```
strtr(где меняем, что меняем, на что меняем);
```

Примеры

Пример

В данном примере функция заменит символы 1 и 2 на 'a' и 'b' соответственно:

```
<?php
        echo strtr('111222', ['1'=>'a', '2'=>'b']);
?>
```

Результат выполнения кода:

aaabbb

Пример

В данном примере функция также заменит символы 1 и 2 на 'a' и 'b' соответственно:

```
<?php
        echo strtr('111222', '12', 'ab');
?>
```

Результат выполнения кода:

aaabbb

Функция substr_replace

Функция **substr_replace** заменяет указанную часть строки на другую.

Эта функция вырезает указанную часть строки (параметрами указывается откуда начинать вырезание и сколько символов взять) и заменяет вырезанную часть на указанную строку.

См. также функцию str_replace, которая осуществляет поиск и замену по строке.

См. также функцию substr, которая вырезает часть подстроки.

Синтаксис

```
substr_replace(где меняем, на что меняем, с какого символа, [сколько символов]);
```

Если последний параметр не указан - замена произведется до конца строки.

Примеры

Пример

Давайте вырежем из строки символы, начиная с первого (нумерация символов начинается с нуля), 3 штуки и вместо них вставим '!!!':

```
<?php
        echo substr_replace('abcde', '!!!', 1, 3);
?>
```

Результат выполнения кода:

a!!!e

Пример

Давайте вырежем из строки символы, начиная с первого до конца строки (так как последний параметр не указан) и вместо них вставим '!!!':

```
<?php
        echo substr_replace('abcde', '!!!', 1);
?>
```

Результат выполнения кода:

a!!!

Функция strpos

Функция **strpos** возвращает позицию первого вхождения подстроки в другую строку. Первым параметром функция принимает строку, в которой осуществляется поиск, вторым параметром - подстроку, которую следует искать.

Результатом выполнения функции будет позиция первого символа найденной подстроки, а если такая подстрока не будет найдена - то false. Учтите, что нумерация идет с нуля и, если подстрока находится в начале строки, то результатом функции будет 0. Это может привести к ошибкам, если сделать, к примеру, так: if(strpos()) - в этом случае и 0, и false приведут к одинаковому результату, чего бы нам не хотелось бы.

По умолчанию функция ищет с начала строки до первого совпадения. Начало поиска можно регулировать третьим необязательным параметром - если он задан, то поиск начнется не с начала строки, а с указанного места.

Есть также функция **stripos**, которая делает тоже самое, но без учета регистра.

См. также функцию strrpos, которая возвращает позицию последнего вхождения подстроки.

Синтаксис

```
strpos(где ищем, что ищем, [откуда искать]);
```

Примеры

Пример

В данном примере функция вернет позицию первого символа 'c'. Он занимает позицию 2, так как отсчет начинается с 0:

```
<?php
        echo strpos('abc abc', 'c');
?>
```

Результат выполнения кода:

```
2
```

Пример

В данном примере задан третий параметр и поэтому поиск начнется с третьей позиции, в этом случае функция найдет уже второй символ 'c' и выведет его позицию - 6:

```
<?php
        echo strpos('abc abc', 'c', 3);
?>
```

Результат выполнения кода:

```
6
```

Пример

Проверим, что строка начинается на 'http://' (обратите внимание на сравнение по ===, а не по ==):

```
<?php
        if(strpos('http://site.ru', 'http://') === 0) {
                echo 'да';
        } else {
                echo 'нет';
        }
?>
```

Результат выполнения кода:

```
да
```

Функция strrpos

Функция **strrpos** возвращает позицию последнего вхождения подстроки.

Результатом выполнения функции будет позиция первого символа найденной подстроки, а если такая подстрока не будет найдена - то false. Учтите, что нумерация идет с нуля и, если подстрока находится в начале строки, то результатом функции будет 0. Это может привести к ошибкам, если сделать, к примеру, так: if(strpos()) - в этом случае и 0, и false приведут к одинаковому результату, чего бы нам не хотелось бы.

По умолчанию функция ищет с начала строки до первого совпадения. Начало поиска можно регулировать третьим необязательным параметром - если он задан, то поиск начнется не с начала строки, а с указанного места.

Есть также функция **strripos**, которая делает тоже самое, но без учета регистра.

См. также функцию strpos, которая возвращает позицию первого вхождения подстроки.

Синтаксис

```
strrpos(где ищем, что ищем, [откуда искать]);
```

Примеры

Пример

В данном примере функция вернет позицию последнего вхождения символа 'a':

```
<?php
        echo strrpos('abc abc', 'a');
?>
```

Результат выполнения кода:

```
4
```

Функция strstr

Функция **strstr** находит первое вхождение подстроки в строку и возвращает часть строки начиная этого места до конца строки. В отличие от strchr ищет вхождение подстроки из нескольких символов, а не вхождение одного символа.

Есть также функция **stristr**, которая делает тоже самое, но без учета регистра.

См. также функцию strchr, которая находит первое вхождение символа и возвращает оставшуюся часть строки.

См. также функцию strrchr, которая находит последнее вхождение символа и возвращает оставшуюся часть строки.

Синтаксис

```
strstr(где ищем, что ищем);
```

Примеры

Пример

В данном примере функция достанет адрес страницы без доменного имени из url (вернет подстроку, начиная с первого /, до конца строки)

```
<?php
        echo strstr('site.ru/folder1/folder2/page.html', '/');
?>
```

Результат выполнения кода:

```
/folder1/folder2/page.html
```

Функция explode

Функция **explode** разбивает строку в массив по определенному разделителю.

См. также функцию implode, которая объединяет элементы массива в строку.

Синтаксис

```
explode(разделитель, строка);
```

Примеры

Пример

Давайте разобьем части даты в массив (разделитель - дефис):

```
<?php
        $date = '2025-12-31';    $arr = explode('-', $date);       var_dump($arr);?>
```

Результат выполнения кода:

```
['2025', '12', '31']
```

Пример . Применение

Пусть дана дата в формате '2025-12-31'. Давайте сделаем из нее дату в формате '31.12.2025'. Для этого разобьем ее в массив через **explode** и сформируем новую строку в нужном нам формате:

```php
<?php
        $date = '2025-12-31';    $arr = explode('-', $date);       echo
$arr[2].'.'.$arr[1].'.'.$arr[0];?>
```

Результат выполнения кода:

31.12.2025

Функция implode

Функция **implode** сливает массив в строку с указанным разделителем.

См. также функцию explode, которая разбивает строку в массив по указанному разделителю.

Синтаксис

```
implode(разделитель, строка);
```

Примеры

Пример

Давайте сольем массив **['a', 'b', 'c']** в строку **'a-b-c'**:

```php
<?php
        $arr = ['a', 'b', 'c'];
        $str = implode('-', $arr);
        echo $str;
?>
```

Результат выполнения кода:

a-b-c

Функция str_split

Функция **str_split** разбивает строку в массив.

Первым параметром она принимает строку, а вторым - количество символов в элементе массива.

К примеру, если второй параметр задать как 3 - функция разобьет строку в массив так, чтобы в каждом элементе массива было по 3 символа.

См. также функцию explode, которая разбивает строку в массив по заданному разделителю.

Синтаксис

```
str_split(строка, количество символов в элементе массива);
```

Примеры

Пример

Давайте разобьем строку по 2 символа в элементе массива (обратите внимание на то, что последнему элементу не хватило символов и там их не 2, а один):

```php
<?php
        $str = 'abcde';
        $arr = str_split($str, 2);
        var_dump($arr);
?>
```

Результат выполнения кода:

['ab', 'cd', 'e'];

Пример

Давайте разобьем строку по 3 символа в элементе массива:

```php
<?php
        $str = 'abcdefg';
```

```
        $arr = str_split($str, 3);
        var_dump($arr);
?>
```

Результат выполнения кода:

['abc', 'def', 'g'];

Пример . Применение

Давайте найдем сумму цифр числа. Для этого разобьем число в массив с помощью **str_split** и сложим элементы этого массива с помощью array_sum:

```
<?php
        $num = 12345;    echo array_sum(str_split($num, 1));?>
```

Результат выполнения кода:

15

Функция trim

Функция **trim** удаляет пробелы с начала и конца строки. Может также удалять другие символы, если их указать вторым параметром.

Есть также функции ltrim и rtrim, которые делают то же самое, но ltrim только для левого края строки, а rtrim - для правого.

Синтаксис

```
trim(строка);
trim(строка, символы);
```

Примеры

Пример

Давайте удалим пробелы по краям строки:

```
<?php
        var_dump(trim(' hello '));
?>
```

Результат выполнения кода:

'hello'

Пример

Давайте удалим слеши по краям строки:

```
<?php
        echo trim('/hello/', '/');
?>
```

Результат выполнения кода:

hello

Пример

Давайте удалим слеши и точки по краям строки:

```
<?php
        echo trim('/hello.', '/.');
?>
```

Результат выполнения кода:

hello

Пример

Функция удаляет любое количество указанных символов, если они стоят с краю:

```
<?php
        echo trim('../../hello...', '/.');
```

```
?>
```

Результат выполнения кода:

```
hello
```

Пример

Можно указать диапазон символов с помощью двух точек '..'. К примеру, укажем, что мы хотим удалить символы от 'a' до 'z':

```
<?php
        echo trim('hello.', 'a..z');
?>
```

Результат выполнения кода:

```
ell
```

Функция strrev

Функция **strrev** переворачивает строку так, чтобы символы шли в обратном порядке.

Синтаксис

```
strrev(строка);
```

Примеры

Пример

В данном примере функция перевернет строку '12345':

```
<?php
        echo strrev('12345');
?>
```

Результат выполнения кода:

```
54321
```

Функция str_shuffle

Функция **str_shuffle** переставляет символы в строке в случайном порядке.

См. также функцию shuffle, которая переставляет элементы массива в случайном порядке.

Синтаксис

```
str_shuffle(строка);
```

Примеры

Пример

В данном примере при каждом новом обновлении страницы строка будет иметь другой порядок символов:

```
<?php
        echo str_shuffle('Hello world!');
?>
```

Результат выполнения кода:

```
lledrow!loH
```

Функция number_format

Функция **number_format** позволяет форматировать число.

В основном используется для того, чтобы отделять тройки чисел пробелами, к примеру, из **1234567**она может сделать **1 234 567**.

По умолчанию функция разделяет тройки запятыми: из **1234567** делает **1,234,567**.

Кроме того, она позволяет регулировать количество знаков после дробной части. Это количество задается вторым необязательным параметром.

Например, можно из дроби **12345.6789** сделать дробь **12 345.68** - функция расставит пробелы между тройками и округлит дробь до двух знаков в дробной части.

Третий необязательный параметр задает разделитель дробной части (по умолчанию точка, но можно сменить). Обязательно вместе с третьим параметром должен быть и четвертый - он устанавливает разделитель троек чисел (по умолчанию запятая, но можно сменить, к примеру, на пробел).

Синтаксис

Функция принимает один, два или четыре параметра (не три).

```
number_format(число);
```

```
number_format(число, количество знаков дробной части);
```

```
number_format(число, количество знаков дробной части, разделитель дробной части, разделитель тысяч);
```

Примеры

Пример

В данном примере тройки чисел отделяются запятой:

```
<?php
        echo number_format(1234567);
?>
```

Результат выполнения кода:

```
1,234,567
```

Пример

В данном примере тройки чисел отделяются запятой, а дробная часть округляется до двух знаков:

```
<?php
        echo number_format(1234.567, 2);
?>
```

Результат выполнения кода:

```
1,234.57
```

Пример

В данном примере тройки чисел отделяются пробелом, дробная часть округляется до двух знаков, разделителем дробной части служит слеш /:

```
<?php
        echo number_format(1234.567, 2, '/', ' ');
?>
```

Результат выполнения кода:

```
1 234/57
```

Пример

В данном примере тройки чисел отделяются пробелом, дробная часть округляется до двух знаков, разделителем дробной части служит точка:

```
<?php
        echo number_format(1234.567, 2, '.', ' ');
?>
```

Результат выполнения кода:

```
1 234.57
```

Функция str_repeat

Функция **str_repeat** повторяет строку заданное количество раз.

См. также функцию str_pad, которая дополняет строку до заданного размера другой строкой.

Синтаксис

```
str_repeat(строка, сколько раз повторить);
```

Примеры

Пример

В данном примере функция повторила строку 'x' 5 раз:

```
<?php
        echo str_repeat('x', 5);
?>
```

Результат выполнения кода:

```
xxxxx
```

Функция htmlspecialchars

Функция **htmlspecialchars** позволяет вывести теги в браузер так, чтобы он не считал их командами, а выводил как строки.

Функция преобразует амперсанд **&** в **&**, **<** в **<**, **>** в **>**.

См. также функцию strip_tags, которая удаляет теги из строки.

См. также функцию htmlspecialchars_decode, которая выполняет преобразование, обратное функции htmlspecialchars.

Синтаксис

```
htmlspecialchars(строка);
```

Примеры

Пример

В данном примере теги выведутся на экран, а не преобразуются в команды браузера:

```
<?php
        echo htmlspecialchars('<b>жирный текст</b>');
?>
```

Результат выполнения кода:

```
<b>жирный текст</b>
```

Функция strip_tags

Функция **strip_tags** удаляет HTML теги из строки (не трогая их содержимого). Вторым необязательным параметром можно указать разрешенные теги - они не будут удалены. Их указываем в таком формате: '<b>' или '<b><p>', если хотим оставить несколько тегов.

См. также функцию htmlspecialchars, которая позволяет вывести теги в браузер.

Синтаксис

```
strip_tags(строка, [разрешенные теги]);
```

Примеры

Пример

В данном примере функция удалит HTML теги из строки:

```
<?php
        echo strip_tags('hello <b>world</b>');
?>
```

Результат выполнения кода:

```
hello world
```

Функция chr

Функция **chr** находит символ по его ASCII коду.

См. также функцию ord, которая возвращает ASCII-код символа.

Синтаксис

```
chr(код символа);
```

Примеры

Пример

В данном примере выводится символ, который имеет код 97:

```
<?php
        echo chr(97);
?>
```

Результат выполнения кода:

```
a
```

Пример . Применение

Давайте выведем случайную маленькую букву латинского алфавита. Для этого посмотрим на таблицу ASCII и увидим, что маленькие латинские буквы имеют коды от 97 до 122. Поэтому сгенерируем случайное число в этом диапазоне с помощью mt_rand и результат возьмем в chr:

```
<?php
echo chr(mt_rand(97, 122));
?>
```

Пример . Применение

Давайте теперь сформируем случайную строку из 6 маленьких латинских букв. Для этого описанную в предыдущем примере операцию повторим 6 раз в цикле:

```
<?php
        $str = '';
        for ($i = 1; $i <= 6; $i++) {
        $str .= chr(mt_rand(97, 122));
        }
        echo $str;
?>
```

Пример . Применение

Большие латинские буквы имеют диапазон 65-90, а маленькие - 97-122. То есть между ними есть дыра. Давайте получим случайный символ маленькую или большую латинскую букву. Для этого с помощью range сформируем 2 массива: первый с числами от 65 до 90, а второй с числами от 97 до 122. Объединим их вместе с помощью array_merge и затем выведем случайный элемент этого массива с помощью array_rand:

```
<?php
$codes = array_merge(range(65, 90), range(97, 122));
        echo chr($codes[array_rand($codes)]);
?>
```

Функция ord

Функция **ord** возвращает ASCII код символа. Параметром принимает один символ или целую строку (в этом случае возвращает код ее первого символа).

См. также функцию chr, которая возвращает символ по его коду.

Синтаксис

```
ord(символ или строка);
```

Примеры

Пример

Давайте узнаем код символа 'W':

```
<?php
        echo ord('W');
?>
```

Результат выполнения кода:

```
87
```

Пример

Выведем код первого символа строки 'World':

```
<?php
        echo ord('World');
?>
```

Результат выполнения кода:

```
87
```

Функция str_word_count

Функция **str_word_count** подсчитывает количество слов в строке.

Функция может принимать второй необязательный параметр, число 1 или 2.

Если он не задан, то возвращается целое число, равное количеству слов.

Если передано 1, то возвращается массив, содержащий все слова, входящие в строку.

Если передано 2, то возвращается массив, ключами которого являются позиции в строке, а значениями - соответствующие слова.

См. также функцию substr_count, которая подсчитывает сколько раз встречается подстрока в строке.

Синтаксис

```
str_word_count(строка, [число]);
```

Примеры

Пример

Давайте подсчитаем количество в строке:

```
<?php
        echo str_word_count('Hello my world');
?>
```

Результат выполнения кода:

```
3
```

Функция substr_count

Функция **substr_count** подсчитывает сколько раз встречается подстрока в строке.

См. также функцию str_word_count, которая подсчитывает количество слов в строке.

Синтаксис

```
substr_count(строка, подстрока);
```

Примеры

Пример

Давайте подсчитаем, сколько раз в нашей строке встречается подстрока 'test':

```
<?php
        echo substr_count('test www test', 'test');
?>
```

Результат выполнения кода:

```
2
```

Функция count_chars

Функция **count_chars** подсчитывает сколько раз встречаются различные символы в строке.

Первым параметром функция принимает строку, а вторым необязательным параметром - модификатор, который изменяет работу функции:

- 0 - массив, индексами которого являются ASCII коды, а значениями - число вхождений соответствующего символа.
- 1 - то же, что и для 0, но информация о символах с нулевым числом вхождений не включается в массив.
- 2 - то же, что и для 0, но в массив включается информация только о символах с нулевым числом вхождений.
- 3 - строка, состоящая из символов, которые входят в исходную строку хотя бы раз.
- 4 - строка, состоящая из символов, которые не входят в исходную строку.

По умолчанию функция ведет себя так, будто второй параметр поставлен в 0.

Синтаксис

```
count_chars(строка, [модификатор]);
```

Функция strchr

Функция **strchr** находит первое вхождение подстроки в строку и возвращает часть строки начиная этого места до конца строки.

См. также функцию strstr, которая находит первое вхождение подстроки и возвращает оставшуюся часть строки.

См. также функцию strrchr, которая находит последнее вхождение символа и возвращает оставшуюся часть строки.

Синтаксис

```
strchr(где ищем, что ищем);
```

Если второй параметр состоит более чем из одного символа, используется только первый символ.

Примеры

Пример . Применение

В данном примере функция достанет адрес страницы без доменного имени из url (вернет подстроку, начиная с первого /, до конца строки)

```
<?php
        echo strchr('site.ru/folder1/folder2/page.html', '/');
?>
```

Результат выполнения кода:

```
/folder1/folder2/page.html
```

Функция strrchr

Функция **strrchr** находит последнее вхождение символа в строку и возвращает часть строки начиная этого места до конца строки.

См. также функцию strchr, которая находит первое вхождение символа и возвращает оставшуюся часть строки.

Синтаксис

```
strrchr(где ищем, что ищем);
```

Если второй параметр состоит более чем из одного символа, используется только первый символ.

Примеры

Пример

В данном примере функция достанет адрес страницы из url (вернет подстроку, начиная с последнего /, до конца строки)

```
<?php
        echo strrchr('site.ru/folder1/folder2/page.html', '/');
?>
```

Результат выполнения кода:

Задачи на функции работы со строками в PHP

Примеры решения задач

Задача

Задача. Дана строка **'minsk'**. Сделайте из нее строку **'MINSK'**.

Решение: воспользуемся функцией strtoupper и сразу получим результат:

```
<?php
        echo strtoupper('minsk');
?>
```

Задача

Задача. Дана строка **'минск'**. Сделайте из нее строку **'МИНСК'**.

Решение: функцией **strtoupper** мы не можем воспользоваться, так как она некорректно работает с русскими буквами. Воспользуемся функцией mb_strtoupper и сразу получим результат:

```
<?php
        echo mb_strtoupper('минск');
?>
```

Задача

Задача. Дана строка **'MINSK'**. Сделайте из нее строку **'Minsk'**.

Решение: готовой функции для решения задачи в PHP не существует. Поэтому сначала воспользуемся функцией strtolower (в результате получится **'minsk'**), а затем функцией ucfirst:

```
<?php
        echo ucfirst(strtolower('minsk'));
?>
```

Задача

Задача. В переменной **$date** лежит дата в формате **'31-12-2030'**. Преобразуйте эту дату в формат **'2030.12.31'**.

Решение: для начала разобьем строку **'31-12-2030'** в массив с помощью функции explode:

```
<?php
        //Разбиваем строку в массив по разделителю '-':
        $arr = explode('-', '31-12-2030');
?>
```

В полученном массиве в **$arr[0]** будет лежать **31** (то есть день), в **$arr[1]** - месяц, а в **$arr[2]** - год. Сольем элементы этого массива в новую строку:

```
<?php
        //Разбиваем строку в массив по разделителю '-':
        $arr = explode('-', '31-12-2030');

        //Получим дату в нужном формате:
        echo $arr[2].'.'.$arr[1].'.'.$arr[0];
?>
```

Задачи для решения

Работа с регистром символов

Для решения задач данного блока вам понадобятся следующие функции: strtolower, strtoupper, ucfirst, lcfirst,ucwords.

Дана строка **'php'**. Сделайте из нее строку **'PHP'**.

Дана строка **'PHP'**. Сделайте из нее строку **'php'**.

Дана строка **'london'**. Сделайте из нее строку **'London'**.

Дана строка **'London'**. Сделайте из нее строку **'london'**.

Дана строка **'london is the capital of great britain'**. Сделайте из нее строку **'London Is The Capital Of Great Britain'**.

Дана строка **'LONDON'**. Сделайте из нее строку **'London'**.

Работа с strlen

Для решения задач данного блока вам понадобятся следующие функции: strlen.

Дана строка **'html css php'**. Найдите количество символов в этой строке.

Дана переменная **$password**, в которой хранится пароль пользователя. Если количество символов пароля больше 5-ти и меньше 10-ти, то выведите пользователю сообщение о том, что пароль подходит, иначе сообщение о том, что нужно придумать другой пароль.

Работа с substr

Для решения задач данного блока вам понадобятся следующие функции: substr.

Дана строка **'html css php'**. Вырежьте из нее и выведите на экран слово 'html', слово 'css' и слово 'php'.

Дана строка. Вырежите и выведите на экран последние **3** символа этой строки.

Дана строка. Проверьте, что она начинается на **'http://'**. Если это так, выведите 'да', если не так - 'нет'.

Дана строка. Проверьте, что она начинается на **'http://'** или на **'https://'**. Если это так, выведите 'да', если не так - 'нет'.

Дана строка. Проверьте, что она заканчивается на **'.png'**. Если это так, выведите 'да', если не так - 'нет'.

Дана строка. Проверьте, что она заканчивается на **'.png'** или на **'.jpg'**. Если это так, выведите 'да', если не так - 'нет'.

Дана строка. Если в этой строке более 5-ти символов - вырежите из нее первые **5** символов, добавьте троеточие в конец и выведите на экран. Если же в этой строке **5** и менее символов - просто выведите эту строку на экран.

Работа с str_replace

Для решения задач данного блока вам понадобятся следующие функции: str_replace.

Дана строка **'31.12.2013'**. Замените все точки на дефисы.

Дана строка **$str**. Замените в ней все буквы 'a' на цифру 1, буквы 'b' - на 2, а буквы 'c' - на 3.

Дана строка с буквами и цифрами, например, **'1a2b3c4b5d6e7f8g9h0'**. Удалите из нее все цифры. То есть в нашем случае должна получится строка **'abcbdefgh'**.

Работа с strtr

Для решения задач данного блока вам понадобятся следующие функции: strtr.

Дана строка **$str**. Замените в ней все буквы 'a' на цифру 1, буквы 'b' - на 2, а буквы 'c' - на 3. Решите задачу двумя способами работы с функцией **strtr** (массив замен и две строки замен).

Работа с substr_replace

Для решения задач данного блока вам понадобятся следующие функции: substr_replace.

Дана строка **$str**. Вырежите из нее подстроку с 3-го символа (отсчет с нуля), 5 штук и вместо нее вставьте '!!!'.

Работа с strpos, strrpos

Для решения задач данного блока вам понадобятся следующие функции: strpos, strrpos.

Дана строка 'abc abc abc'. Определите позицию первой буквы 'b'.

Дана строка 'abc abc abc'. Определите позицию последней буквы 'b'.

Дана строка 'abc abc abc'. Определите позицию первой найденной буквы 'b', если начать поиск не с начала строки, а с позиции 3.

Дана строка 'aaa aaa aaa aaa aaa'. Определите позицию второго пробела.

Проверьте, что в строке есть две точки подряд. Если это так - выведите 'есть', если не так - 'нет'.

Проверьте, что строка начинается на **'http://'**. Если это так - выведите 'да', если не так - 'нет'.

Работа с explode, implode

Для решения задач данного блока вам понадобятся следующие функции: explode, implode.

Дана строка **'html css php'**. С помощью функции explode запишите каждое слово этой строки в отдельный элемент массива.

Дан массив с элементами **'html', 'css', 'php'**. С помощью функции implode создайте строку из этих элементов, разделенных запятыми.

В переменной **$date** лежит дата в формате **'2013-12-31'**. Преобразуйте эту дату в формат **'31.12.2013'**.

Работа с str_split

Для решения задач данного блока вам понадобятся следующие функции: str_split.

Дана строка **'1234567890'**. Разбейте ее на массив с элементами **'12', '34', '56', '78', '90'**.

Дана строка **'1234567890'**. Разбейте ее на массив с элементами **'1', '2', '3', '4', '5', '6', '7', '8', '9', '0'**.

Дана строка **'1234567890'**. Сделайте из нее строку **'12-34-56-78-90'** не используя цикл.

Работа с trim, ltrim, rtrim

Для решения задач данного блока вам понадобятся следующие функции: trim, ltrim, rtrim.

Дана строка. Очистите ее от концевых пробелов.

Дана строка **'/php/'**. Сделайте из нее строку **'php'**, удалив концевые слеши.

Дана строка **'слова слова слова.'**. В конце этой строки может быть точка, а может и не быть. Сделайте так, чтобы в конце этой строки гарантировано стояла точка. То есть: если этой точки нет - ее надо добавить, а если есть - ничего не делать. Задачу решите через **rtrim** без всяких ифов.

Работа с strrev

Для решения задач данного блока вам понадобятся следующие функции: strrev.

Дана строка **'12345'**. Сделайте из нее строку **'54321'**.

Проверьте, является ли слово **палиндромом** (одинаково читается во всех направлениях, примеры таких слов: **madam, otto, kayak, nun, level**).

Работа с str_shuffle

Для решения задач данного блока вам понадобятся следующие функции: str_shuffle.

Дана строка. Перемешайте символы этой строки в случайном порядке.

Создайте строку из 6-ти случайных маленьких латинских букв так, чтобы буквы не повторялись. Нужно сделать так, чтобы в нашей строке могла быть любая латинская буква, а не ограниченный набор.

Работа с number_format

Для решения задач данного блока вам понадобятся следующие функции: number_format.

Дана строка **'12345678'**. Сделайте из нее строку **'12 345 678'**.

Работа с str_repeat

Для решения задач данного блока вам понадобятся следующие функции: str_repeat.

Нарисуйте пирамиду, как показано на рисунке, только у вашей пирамиды должно быть 9 рядов, а не 5. Решите задачу с помощью **одного цикла** и функции str_repeat.

```
x
xx
xxx
xxxx
xxxxx
```

Нарисуйте пирамиду, как показано на рисунке. Решите задачу с помощью **одного цикла** и функции str_repeat.

```
1
22
333
4444
55555
666666
7777777
88888888
999999999
```

Работа с strip_tags и htmlspecialchars

Для решения задач данного блока вам понадобятся следующие функции: htmlspecialchars, strip_tags.

Дана строка **'html, php, js'**. Удалите теги из этой строки.

Дана строка **$str**. Удалите все теги из этой строки, кроме тегов и <i>.

Дана строка **'html, php, js'**. Выведите ее на экран **'как есть'**: то есть браузер не должен преобразовать в жирный.

Работа с chr и ord

Для решения задач данного блока вам понадобятся следующие функции: chr, ord.

Узнайте код символов **'a'**, **'b'**, **'c'**, **пробела**.

Изучите таблицу ASCII. Определите границы, в которых располагаются буквы английского алфавита.

Выведите на экран символ с кодом **33**.

Запишите в переменную **$str** случайный символ - большую букву латинского алфавита.*Подсказка: с помощью таблицы ASCII определите какие целые числа соответствуют большим буквам латинского алфавита.*

Запишите в переменную **$str** случайную строку **$len** длиной, состоящую из маленьких букв латинского алфавита. *Подсказка: воспользуйтесь циклом for или while.*

Дана буква английского алфавита. Узнайте, она маленькая или большая.

Работа с strchr, strrchr

Для решения задач данного блока вам понадобятся следующие функции: strchr, strrchr.

Дана строка **'ab-cd-ef'**. С помощью функции **strchr** выведите на экран строку **'-cd-ef'**.

Дана строка **'ab-cd-ef'**. С помощью функции **strrchr** выведите на экран строку **'-ef'**. Показать решение.

Работа с strstr

Для решения задач данного блока вам понадобятся следующие функции: strstr.

Дана строка **'ab--cd--ef'**. С помощью функции **strstr** выведите на экран строку **'--cd--ef'**.

Задачи

Преобразуйте строку **'var_test_text'** в **'varTestText'**. Скрипт, конечно же, должен работать с любыми аналогичными строками.

Дан массив с числами. Выведите на экран все числа, в которых есть цифра 3.