

# Лекция 4

## Операции с символьными переменными

# Выработка случайных чисел

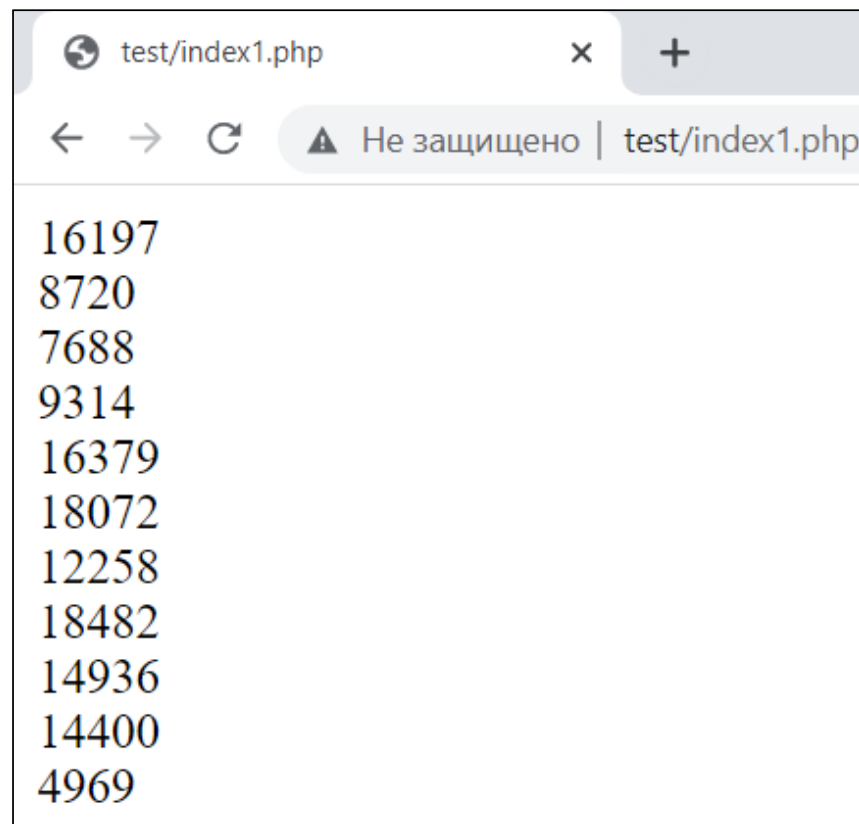
При вызове без параметров `min` и `max`, `rand()` и `mt_rand()` возвращают псевдослучайное целое в диапазоне от 0 до `getrandmax()` / `mt_getrandmax()`.

Начиная с PHP версии 7.0 можно генерировать криптографически безопасные псевдослучайные целые числа (т.е., числа, для которых случайность результата очень критична). Это делается с помощью функции `random_int()`, которая также принимает параметры `min` и `max`.

# Выработка случайных чисел

```
<?php
for ($i=0; $i<=10; $i++)
{
    echo random_int(1000, 20000);
    echo "<br>";
}
echo "<br><br>";

for ($i=0; $i<=10; $i++)
{
    echo mt_rand(10, 200);
    echo "<br>";
}
?>
```



```
test/index1.php x +
< > ↻ ⚠ Не защищено | test/index1.php

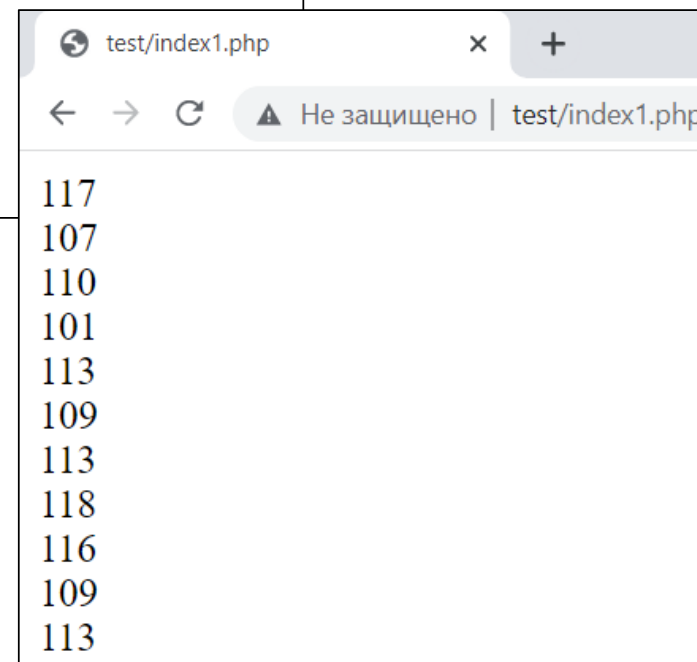
16197
8720
7688
9314
16379
18072
12258
18482
14936
14400
4969
```

# Выработка случайных чисел

Очевидно, что указанные функции выработки псевдослучайных чисел возвращают только целые числа, но случайное целое число из заданного диапазона можно легко преобразовать в соответствующее число с плавающей точкой (скажем, в число из диапазона от 0.0 до 1.0 включительно) с помощью выражения наподобие `rand() / getrandmax()`. После этого указанный диапазон можно масштабировать и сдвигать по мере необходимости.

# Выработка случайных чисел

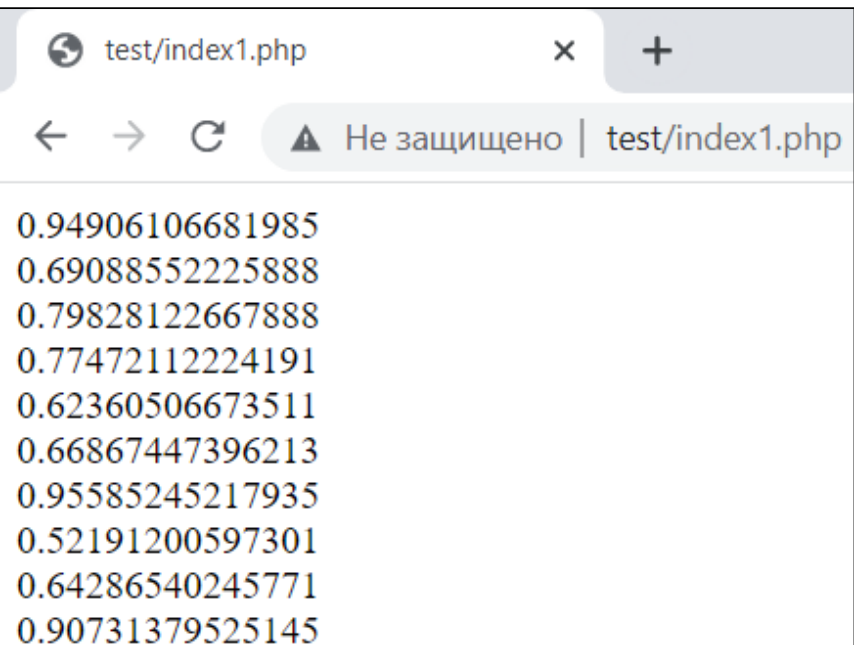
```
<?php
for ($i=0; $i<=10; $i++){
    $random = 100.0 + 20.0 * mt_rand() / mt_getrandmax();
    echo (int)$random;
    echo "<br>";
}
?>
```



# Выработка случайных чисел

$A + (B - A) * \text{слчис}(\text{от } 0 \text{ до } 1)$

```
<?php
for($i=0; $i<10; $i++){
echo mt_rand() / mt_getrandmax()."<br>";
}
?>
```



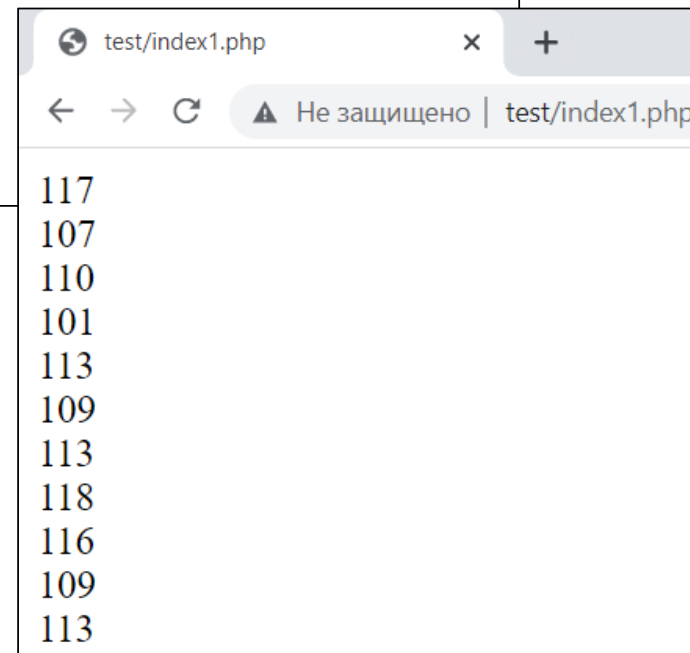
test/index1.php

← → ↻ ⚠ Не защищено | test/index1.php

0.94906106681985  
0.69088552225888  
0.79828122667888  
0.77472112224191  
0.62360506673511  
0.66867447396213  
0.95585245217935  
0.52191200597301  
0.64286540245771  
0.90731379525145

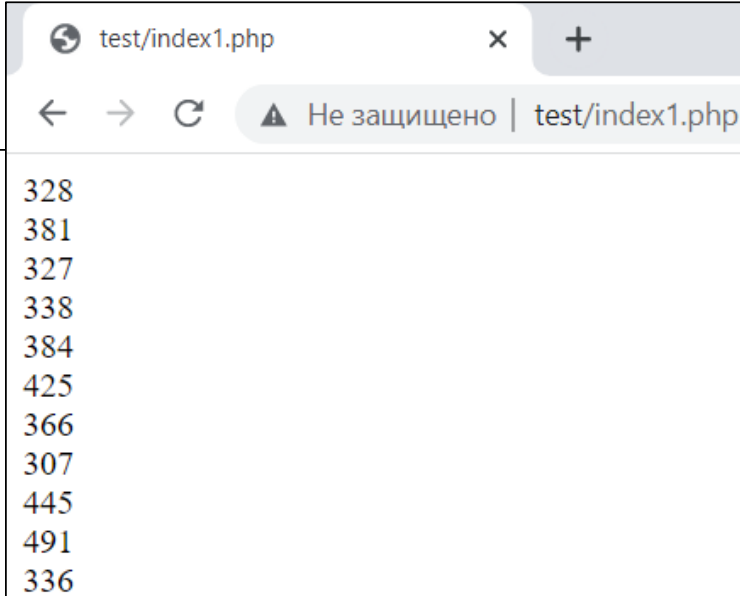
# Выработка случайных чисел

```
<?php
for ($i=0; $i<=10; $i++){
    $random = 100.0 + (120.0-100.0) * mt_rand() / mt_getrandmax();
    echo (int)$random;
    echo "<br>";
}
?>
```



# Выработка случайных чисел

```
<?php
for ($i=0; $i<=10; $i++){
    $random = 300.0 + (500.0-300.0) * mt_rand() / mt_getrandmax();
    echo (int)$random;
    echo "<br>";
}
?>
```



test/index1.php

← → ↻ ⚠ Не защищено | test/index1.php

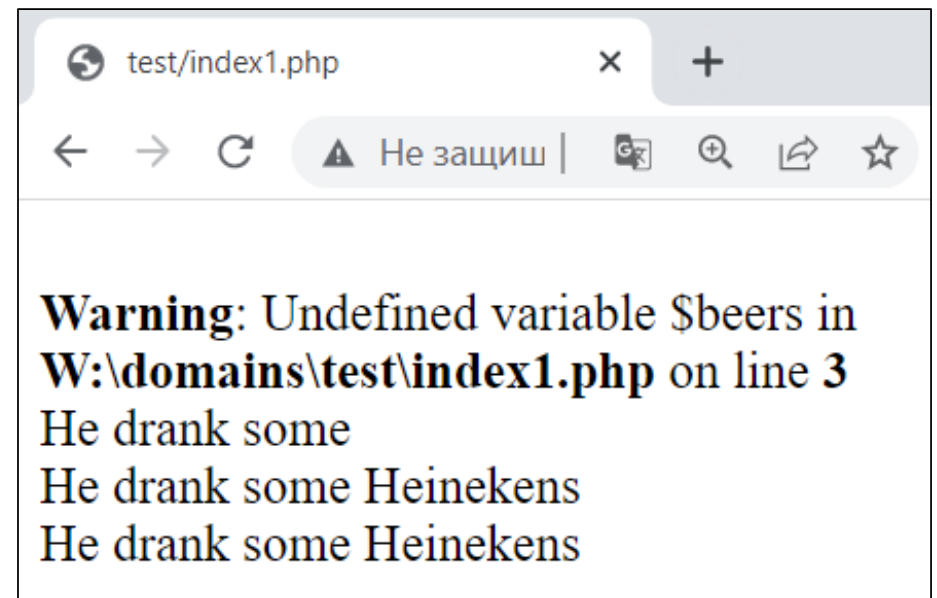
328  
381  
327  
338  
384  
425  
366  
307  
445  
491  
336



# Обработка строковых переменных (строк)

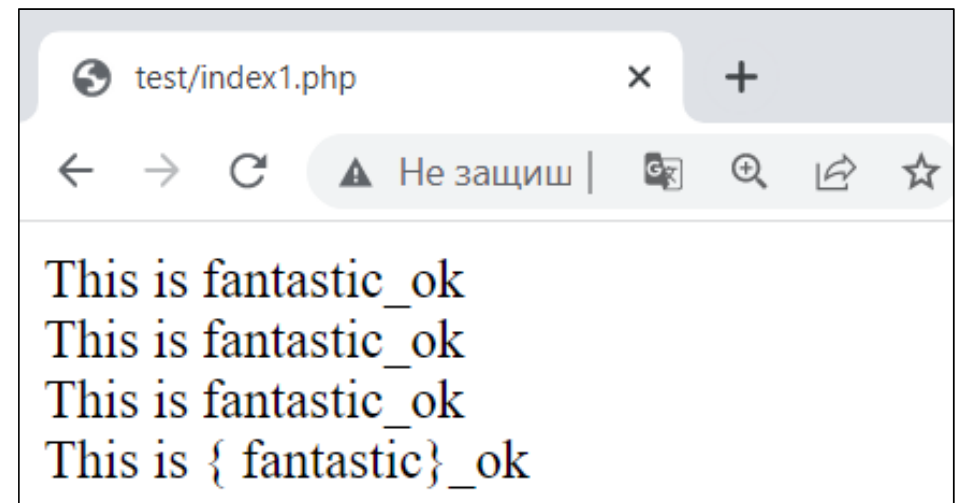
Существует синтаксис, дающий возможность обработки переменной, значения массива (array) или свойства объекта (object). Для этого используются {}.

```
<?php
$beer = 'Heineken';
echo "He drank some $beers";
// не работает, 's' это верный символ для
имени переменной
echo "<br>";
echo "He drank some ${beer}s";
// работает
echo "<br>";
echo "He drank some {$beer}s";
// работает
?>
```



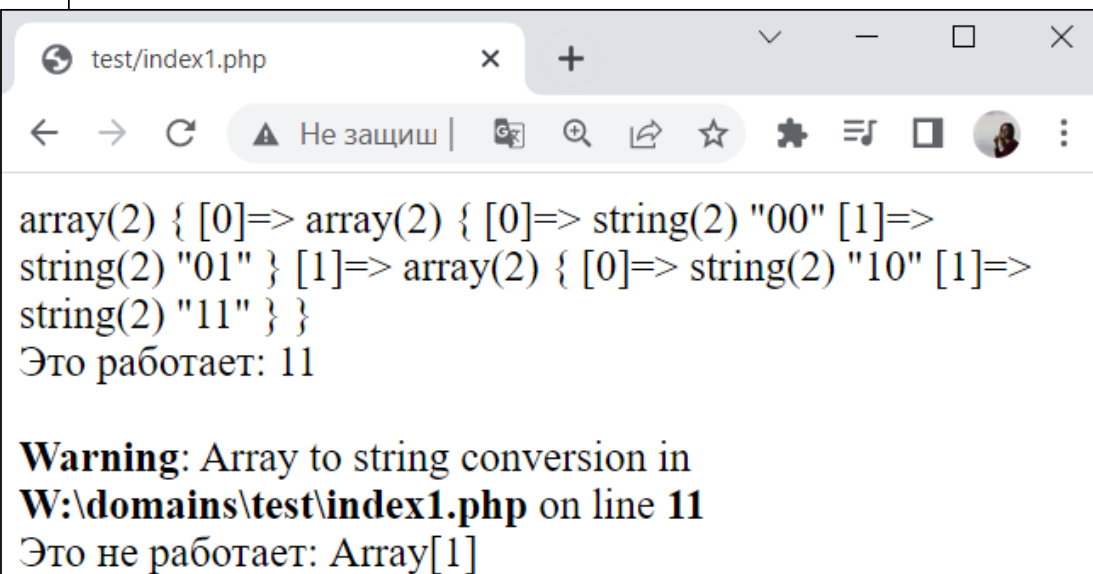
# Обработка строковых переменных (строк)

```
<?php
$great = 'fantastic';
// Работает, выведет: b is fantastic
echo "This is {$great}_ok";
echo "<br>";
echo "This is ${great}_ok";
echo "<br>";
echo "This is $great"."_ok";
echo "<br>";
echo "This is { $great}_ok";
echo "<br>";
?>
```



Точно также могут быть обработаны элемент массива (array) или свойство объекта (object). В индексах массива закрывающая квадратная скобка (]) обозначает конец определения индекса.

```
<?php
// Пример работы с массивом
$arr[0][0]="00";
$arr[1][0]="10";
$arr[0][1]="01";
$arr[1][1]="11";
var_dump ($arr);
echo "<br>";
echo "Это работает: {$arr[1][1]}";
echo "<br>";
echo "Это не работает: $arr[1][1]";
?>
```



test/index1.php

Не защищ |

array(2) { [0]=> array(2) { [0]=> string(2) "00" [1]=> string(2) "01" } [1]=> array(2) { [0]=> string(2) "10" [1]=> string(2) "11" } }

Это работает: 11

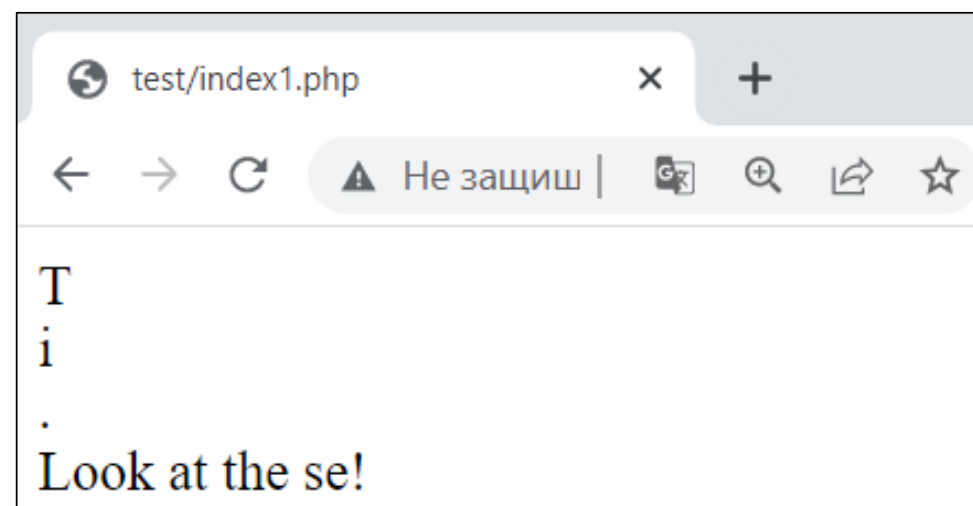
**Warning:** Array to string conversion in  
**W:\domains\test\index1.php** on line 11

Это не работает: Array[1]

# Доступ к символу в строке и его изменение

```
<?php
// Получение первого символа строки
$str = 'This is a test.';
$first = $str[0];
echo $first, "<br>";
// Получение третьего символа строки
$first = $str[0];
$third = $str[2];
echo $third, "<br>";
// Получение последнего символа строки
$str = 'This is still a test.';
$last = $str[strlen($str)-1];
echo $last, "<br>";
// Изменение последнего символа строки
$str = 'Look at the sea';
$str[strlen($str)-1] = '!';
echo $str, "<br>";
?>
```

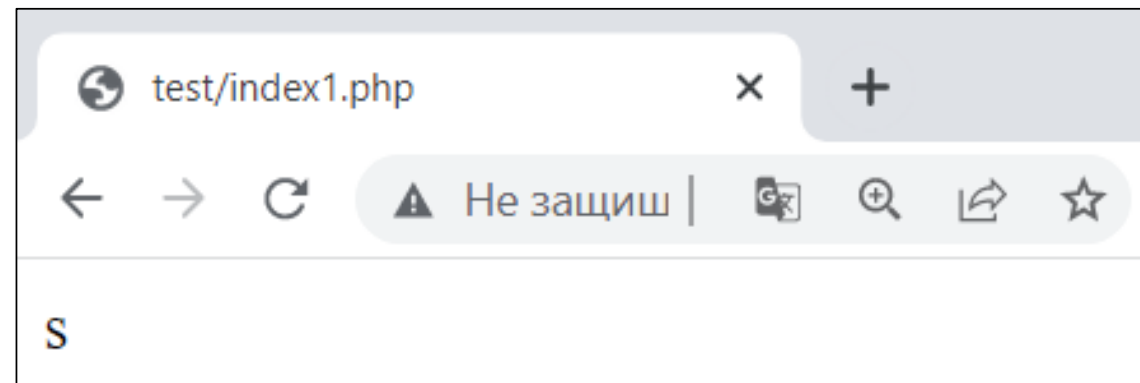
Символы в строках можно использовать или модифицировать: до PHP 8.0 - `str{i}`, после PHP 8.0 – `str[i]`.



# Доступ к символу в строке и его изменение

```
<?php
// Получение первого символа строки
$str = 'This is a test.';
$first = $str{0};
echo $first, "<br>";
?>
```

```
<?php
// Получение 3-его символа с конца
$str = 'This is a test.';
$first = $str[-3];
echo $first, "<br>";
?>
```



# Строковые функции и операторы

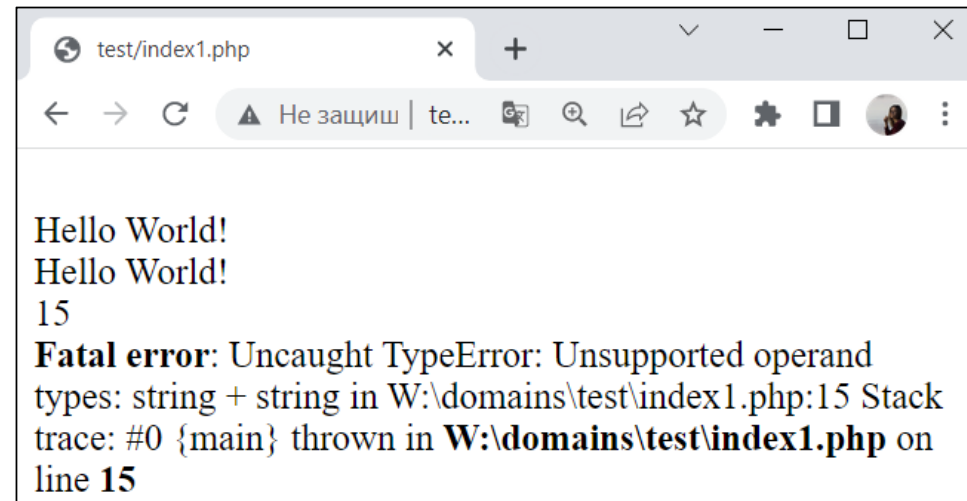
```
<?php
$a = "Hello ";
$b = $a . "World!";
// $b содержит строку "Hello World!" - Это
конкатенация
echo "<br>".$b;

$a = "Hello ";
$a .= "World!";
// $a содержит строку "Hello World!" - Это
присвоение с конкатенацией
echo "<br>".$a;

$a = "5" + "10";
echo "<br>".$a;

$a = "Hello " + "World!";
echo $a;
?>
```

Конкатенация строк. Использование в PHP оператора "+" для конкатенации строк некорректно: если строки содержат числа, то вместо объединения строк будет выполнена операция сложения двух чисел.

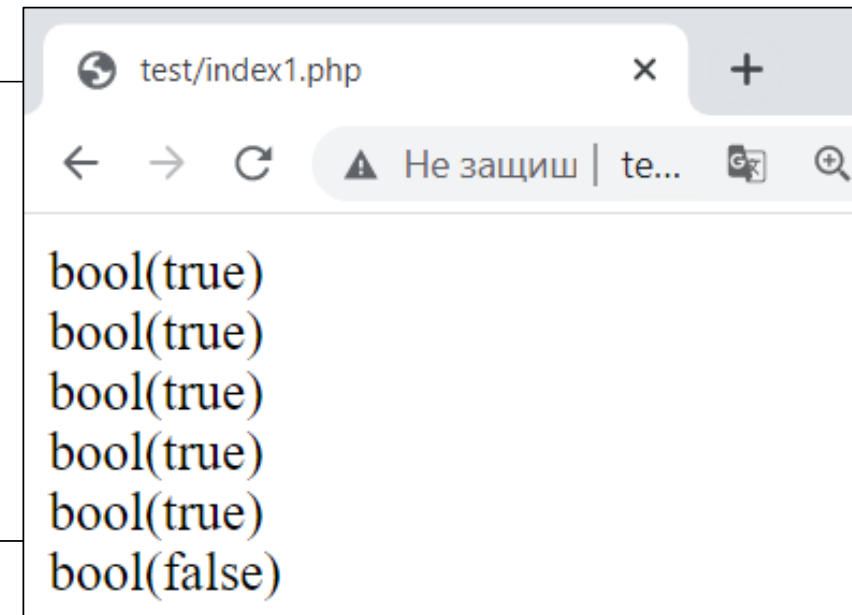


# Строковые функции и операторы

## Операторы сравнения строк

PHP8 сравнивает строку и число как числа только если строка представляет собой число. Иначе, число будет конвертировано в строку, и будет производиться строковое сравнение. До PHP8 любая строка, которую PHP не удастся перевести в число (в том числе и пустая строка), будет восприниматься как 0.

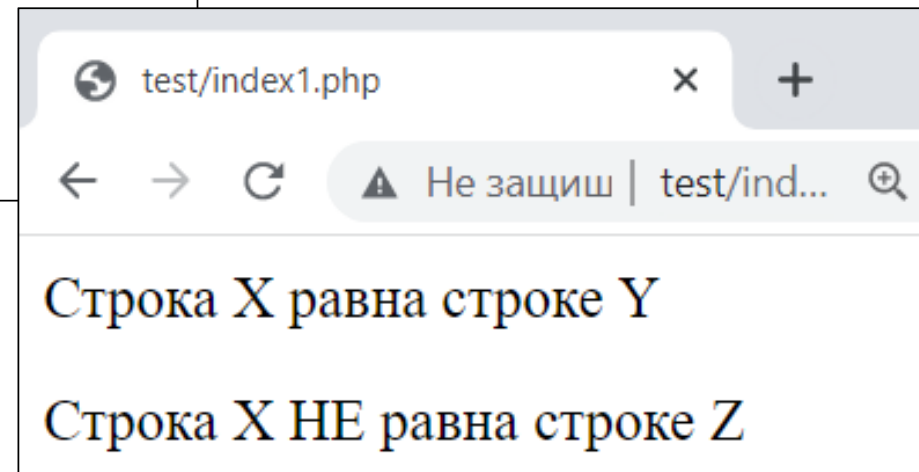
```
<?php
var_dump("1" == "01"); // 1 == 1 -> true
var_dump("10" == "1e1"); // 10 == 10 -> true
var_dump(100 == "1e2"); // 100 == 100 -> true
var_dump(0 == "0"); // 0 == 0 -> true
var_dump(0 == "0.0"); // 0 == 0 -> true
var_dump(0 == ""); // false (до PHP8 - true)
?>
```



# Строковые функции и операторы

## Операторы сравнения строк

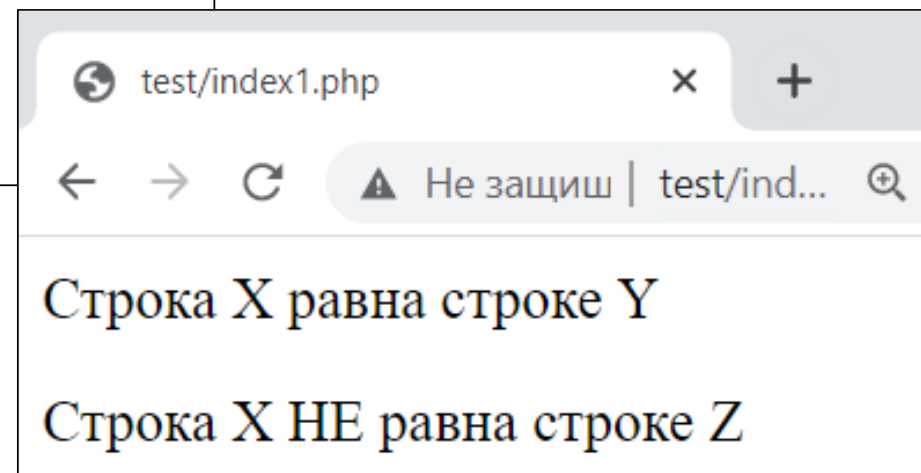
```
<?php
$x="Строка";
$y="Строка";
$z="Строчка";
if ($x == $z) echo "<p>Строка X равна строке Z</p>";
if ($x == $y) echo "<p>Строка X равна строке Y</p>";
if ($x != $z) echo "<p>Строка X НЕ равна строке Z</p>";
// Выводит:
// Строка X равна строке Y
// Строка X НЕ равна строке Z
?>
```





Чтобы избежать путаницы и преобразования типов, рекомендуется пользоваться оператором эквивалентности при сравнении строк. Оператор эквивалентности позволяет всегда корректно сравнивать строки, поскольку сравнивает величины и по значению, и по типу.

```
<?php
$x="Строка";
$y="Строка";
$z="Строчка";
if ($x === $z) echo "<p>Строка X равна строке Z</p>";
if ($x === $y) echo "<p>Строка X равна строке Y</p>";
if ($x !== $z) echo "<p>Строка X НЕ равна строке Z</p>";
// Выводит:
// Строка X равна строке Y
// Строка X НЕ равна строке Z
?>
```

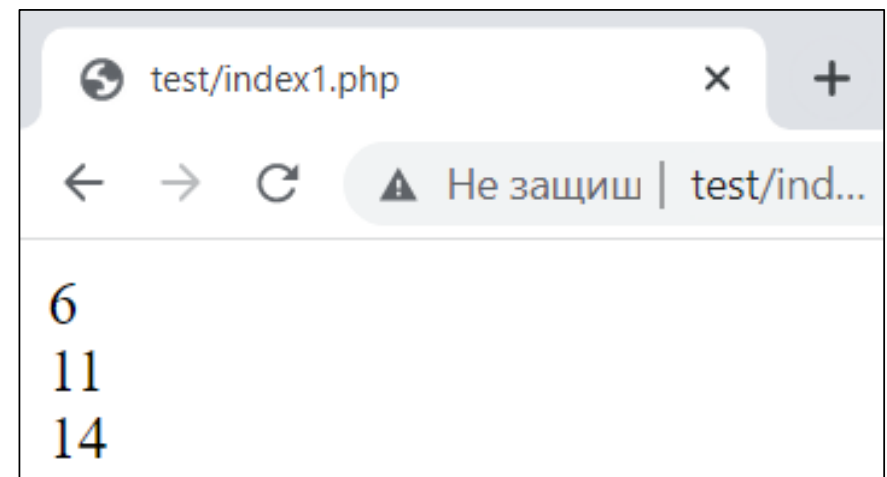


# Функции для работы со строками

## **strlen(string \$st)**

Одна из наиболее полезных функций. Возвращает просто длину строки, т. е., сколько символов содержится в \$st. Строка может содержать любые символы.

```
<?php
$x = "Hello!";
echo strlen($x) . "<br>"; // Выводит 6
$x = "Hello Anna!";
echo strlen($x) . "<br>"; // Выводит 11
$x = "Hello World!!!";
echo strlen($x) . "<br>"; // Выводит 14
?>
```



# Функции для работы со строками

**strpos(string \$haystack, string \$needle, int \$offset=0)**

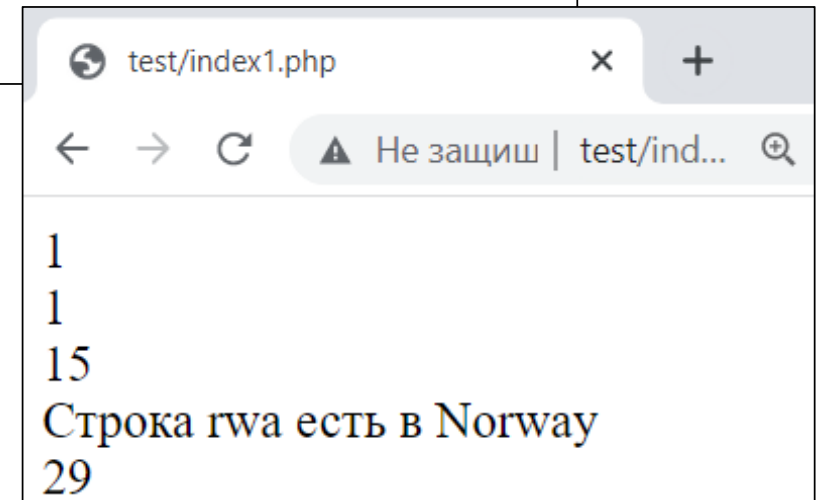
Пытается найти в строке `$haystack` подстроку `$needle` и в случае успеха возвращает позицию (индекс) этой подстроки в строке. Необязательный параметр `$offset` можно задавать, если поиск нужно вести не с начала строки, а с какой-то другой позиции.

## Список изменений

Версия	Описание
8.0.0	Передача целого числа (int) в <b>needle</b> больше не поддерживается.
7.3.0	Передача целого числа (int) в <b>needle</b> объявлена устаревшей.
7.1.0	Добавлена поддержка отрицательных значений <b>offset</b> .

**strpos(string \$haystack, string \$needle, int \$offset=0)**

```
<?php
echo strpos("Hello","el");           // Выводит 1
echo "<br>";
echo strpos("Hello! Hello! Hello! Hello! Hello!","el"); // Выводит 1
echo "<br>";
echo strpos("Hello! Hello! Hello! Hello! Hello!","el", 10); // Выводит 15
echo "<br>";
if (strpos("Norway","rwa") !== false) echo "Строка rwa есть в Norway";
echo "<br>";
echo strpos("Hello! Hello! Hello! Hello! Hello!","el", -6); // Выводит 29
echo "<br>";
?>
```



# Функции для работы со строками

## `substr(string $str, int $start [,int $length])`

Данная функция тоже востребуется очень часто. Ее назначение — возвращать участок строки `$str`, начиная с позиции `$start` и длиной `$length`. Если `$length` не задана, то подразумевается подстрока от `$start` до конца строки `$str`.

Если `$start` больше, чем длина строки, или же значение `$length` равно нулю, то возвращается пустая подстрока.

### Список изменений

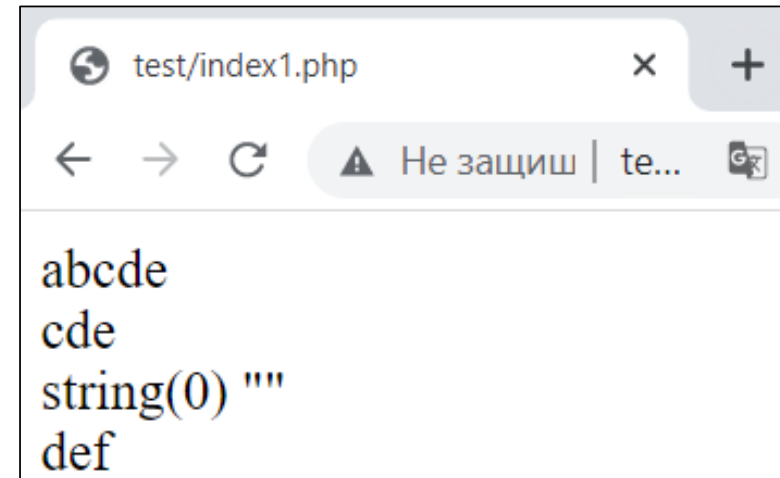
Версия	Описание
8.0.0	Параметр <b>length</b> теперь допускает значение <code>null</code> . Если значение параметра <b>length</b> явно задано как <b>null</b> , функция возвращает подстроку, заканчивающуюся в конце строки; ранее возвращалась пустая строка.
8.0.0	Функция возвращает пустую строку там, где раньше возвращала <b>false</b> .

# Функции для работы со строками

**substr(string \$str, int \$start [,int \$length])**

Если **\$length** отрицательный, то будет отброшено указанное этим аргументом число символов с конца строки **\$str**

```
<?php
$rest = substr("abcdef", 0, -1);
// возвращает "abcde"
$rest = substr("abcdef", 2, -1);
// возвращает "cde"
$rest = substr("abcdef", 4, -4);
// возвращает ""; до PHP 8.0.0 возвращалось false
var_dump($rest);
$rest = substr("abcdef", -3);
// возвращает "def"
?>
```



# Функции для работы со строками

## **strcmp(string \$str1, string \$str2)**

Сравнивает две строки посимвольно (точнее, побайтово) и возвращает: 0, если строки полностью совпадают; <0 (число несовпадения символов), если строка **\$str1** лексикографически меньше **\$str2**; и >0 (число несовпадения символов), если, наоборот, **\$str1** больше **\$str2**. Так как сравнение идет побайтово, то регистр символов влияет на результаты сравнений.

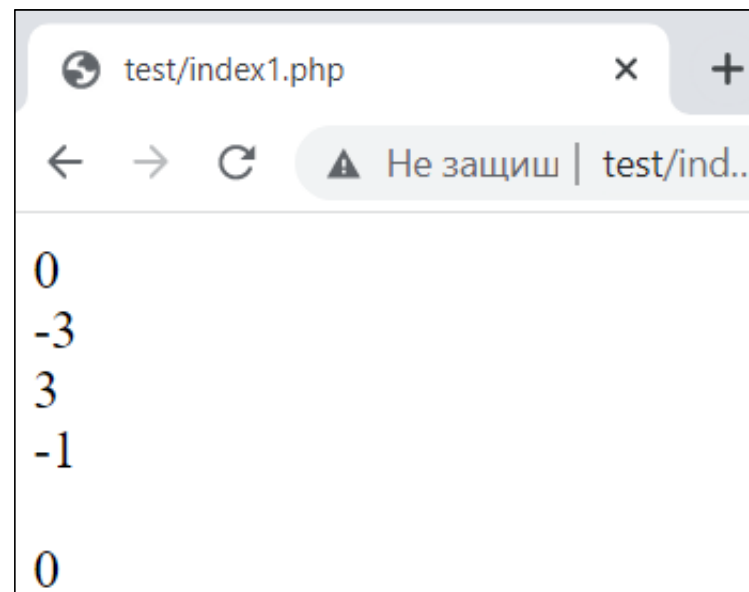
## **strcasecmp(string \$str1, string \$str2)**

То же самое, что и strcmp(), только при работе не учитывается регистр букв. Например, с точки зрения этой функции "ab" и "AB" равны.

# Функции для работы со строками

```
<?php
$str1 = "Programmer";
$str2 = "Programmer";
$str3 = "Program";
$str4 = "programmeR";
echo strcmp($str1,$str2);
echo "<br>";
echo strcmp($str3,$str2);
echo "<br>";
echo strcmp($str2,$str3);
echo "<br>";
echo strcmp($str1,$str4);
echo "<br>";
echo "<br>";

echo strcasecmp($str1,$str4);
echo "<br>";
?>
```





# Функции для работы с блоками текста

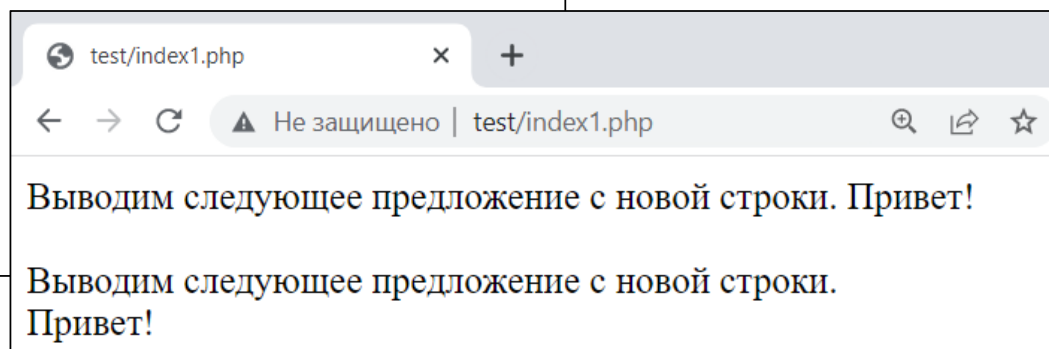
## `str_replace(string $from, string $to, string $str)`

Заменяет в строке `$str` все вхождения подстроки `$from` (с учетом регистра) на `$to` и возвращает результат. Исходная строка, переданная третьим параметром, при этом не меняется.

```
<?php
$str="Выводим следующее предложение с новой строки. \n";
$str1="Привет!";
echo $str, $str1;

echo "<br>", "<br>";

$str2=str_replace("\n", "<br>", $str);
echo $str2, $str1;
?>
```



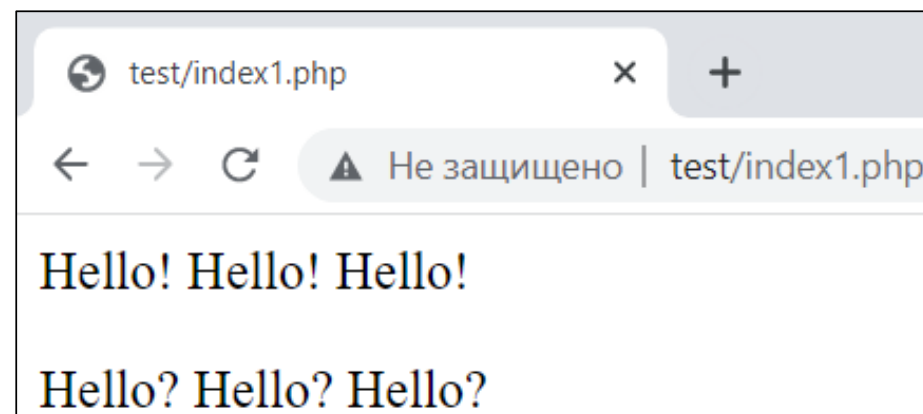
# Функции для работы с блоками текста

`str_replace(string $from, string $to, string $str)`

```
<?php
$str="Hello! Hello! Hello!";
echo $str;

echo"<br>","<br>";

$str1=str_replace("!","?",$str);
echo $str1;
?>
```



# Функции для работы с блоками текста

```
str_pad(  
    string $string,  
    int $length,  
    string $pad_string = " ",  
    int $pad_type = STR_PAD_RIGHT  
)
```

str\_pad — дополняет строку другой строкой до заданной длины.

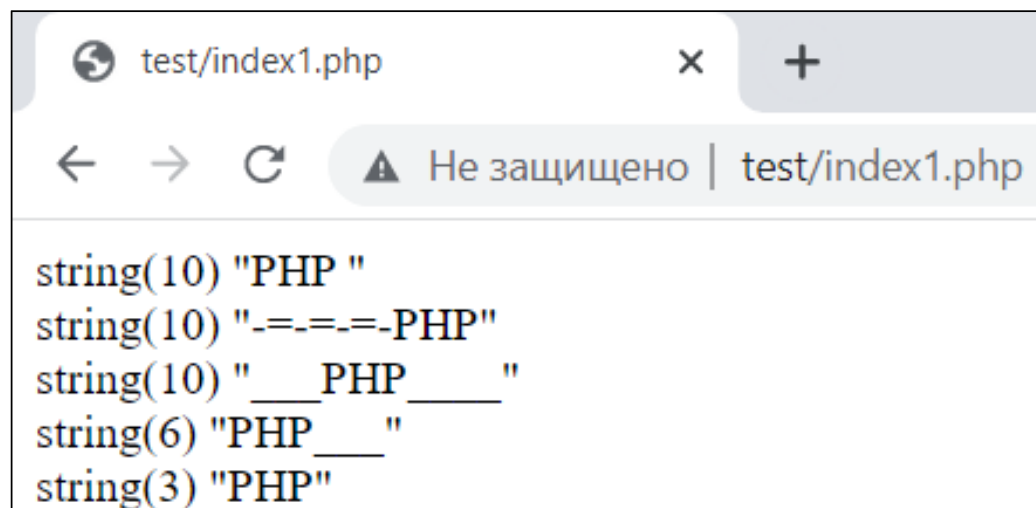
Эта функция возвращает строку **string**, дополненную слева, справа или с обеих сторон до заданной длины. Если необязательный аргумент **pad\_string** не передан, то **string** будет дополнен пробелами, иначе он будет дополнен символами из **pad\_string** до нужной длины.

Если значение **length** отрицательно, меньше или равно длине входной строки, то дополнения не происходит и возвращается исходная строка **string**.

Необязательный аргумент **pad\_type** может иметь значение **STR\_PAD\_RIGHT**, **STR\_PAD\_LEFT** или **STR\_PAD\_BOTH**. Если не указан, то по умолчанию используется **STR\_PAD\_RIGHT**.

# Функции для работы с блоками текста

```
<?php
$input = "PHP";
var_dump(str_pad($input, 10));           // Выводит "PHP      "
var_dump(str_pad($input, 10, "-=", STR_PAD_LEFT)); // Выводит "-=-=-=-PHP"
var_dump(str_pad($input, 10, "_", STR_PAD_BOTH)); // Выводит "__PHP__"
var_dump(str_pad($input, 6, "____"));      // Выводит "PHP____"
var_dump(str_pad($input, 2, "*"));          // Выводит "PHP"
?>
```



The screenshot shows a web browser window with a single tab titled 'test/index1.php'. The address bar displays 'test/index1.php' with a warning icon and the text 'Не защищено' (Not secure). The main content area of the browser shows the output of the PHP script, which is a series of string representations of the padded 'PHP' string: 'string(10) "PHP "', 'string(10) "-=-=-=-PHP"', 'string(10) "\_\_PHP\_\_"', 'string(6) "PHP\_\_\_\_"', and 'string(3) "PHP"'.

```
string(10) "PHP "
string(10) "-=-=-=-PHP"
string(10) "__PHP__"
string(6) "PHP____"
string(3) "PHP"
```

# Функции для работы с блоками текста

```
WordWrap(  
    string $str,  
    int $width=75,  
    string $break="\n«,  
    bool $cut_long_words = false  
)
```

Эта функция оказывается невероятно полезной, например, при форматировании текста письма перед автоматической отправкой его адресату при помощи mail().

Она разбивает блок текста **\$str** на несколько строк, завершаемых символами **\$break**, так, чтобы на одной строке было не более **\$width** букв.

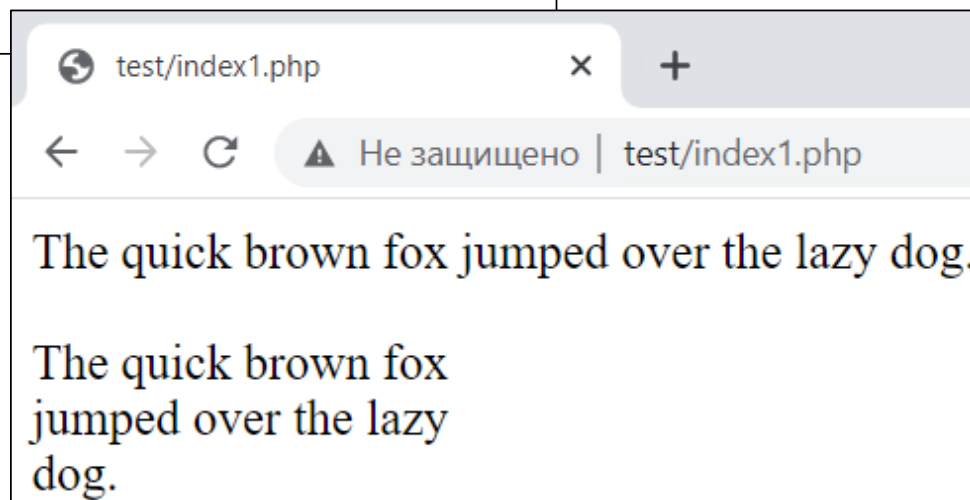
Разбиение происходит по границе слова, так что текст остается читаемым. Возвращается получившаяся строка с символами перевода строки, заданными в **\$break**.

Если параметр **\$cut\_long\_words** установлен в true, строка всегда будет переноситься на указанной ширине width или раньше. Поэтому, если исходная строка содержит слово длиннее заданной ширины строки, то оно будет разорвано. Если установлен в false, функция не разделяет слово, даже если width меньше длины слова.

# Функции для работы с блоками текста

**WordWrap(string \$str, int \$width=75, string \$break="\n")**

```
<?php
$text = "The quick brown fox jumped over the lazy dog.";
echo $text;
$newtext = wordwrap($text, 20, "<br />\n");
echo $newtext;
?>
```



# Функции для работы с блоками текста

## **strip\_tags (string \$str [, string \$allowable\_tags])**

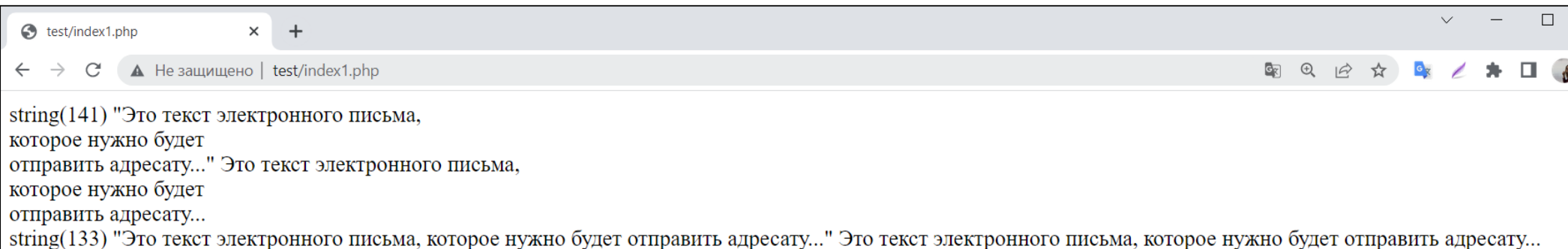
Еще одна полезная функция для работы со строками. Эта функция удаляет из строки все тэги и возвращает результат. В параметре **\$allowable\_tags** можно передать тэги, которые не следует удалять из строки. Они должны перечисляться вплотную друг к другу.

```
$stripped = strip_tags ($str);  
// Удаляет все html - теги из строки (текста)  
$stripped = strip_tags($str, "<head><title>");  
// Удалит все html - теги, кроме html - тегов <head> и <title>
```

# Функции для работы с блоками текста

**strip\_tags (string \$str [, string \$allowable\_tags])**

```
<?php
$str = "Это текст электронного письма,<br> которое нужно будет <br> отправить
адресату...";
var_dump ($str);
echo $str, "<br>";
$stripped = strip_tags ($str); // Удаляет все html - теги из строки (текста)
var_dump ($stripped);
echo $stripped, "<br>";
?>
```



string(141) "Это текст электронного письма,  
которое нужно будет  
отправить адресату..." Это текст электронного письма,  
которое нужно будет  
отправить адресату...

string(133) "Это текст электронного письма, которое нужно будет отправить адресату..." Это текст электронного письма, которое нужно будет отправить адресату...



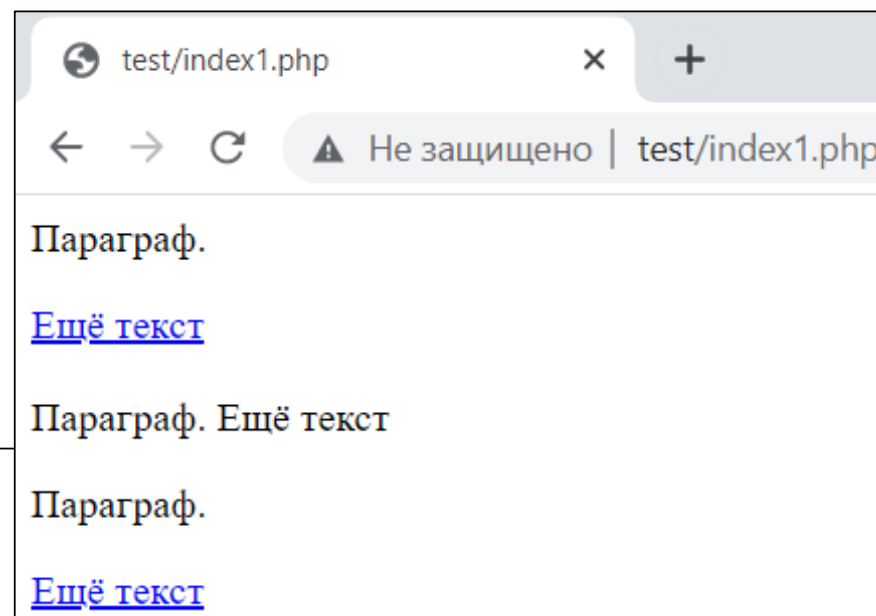
# Функции для работы с блоками текста

**strip\_tags (string \$str [, string \$allowable\_tags])**

```
<?php
$text = '<p>Параграф.</p><!-- Комментарий --> <a href="#fragment">Ещё текст</a>';
echo $text."<br><br>";
echo strip_tags($text);
echo "<br>";

// Разрешаем <p> и <a>
echo strip_tags($text, '<p><a>');

// Начиная с PHP 7.4.0, строка выше
может быть записана как:
// echo strip_tags($text, ['p', 'a']);
?>
```



# Функции для преобразования СИМВОЛЬНЫХ переменных в массив

**array explode ( string \$delimiter , string \$string [, int \$limit ] )**

Возвращает массив строк, полученных разбиением строки `$string` с использованием `$delimiter` в качестве разделителя.

*Список параметров:* **delimiter** – разделитель, **string** – входная строка.

*limit:*

- ✓ Если аргумент `limit` является положительным, возвращаемый массив будет содержать максимум `limit` элементов, при этом последний элемент будет содержать остаток строки `string`.
- ✓ Если параметр `limit` отрицателен, то будут возвращены все компоненты кроме последних `-limit`.
- ✓ Если `limit` равен нулю, то он расценивается как 1.

В результате возвращается массив (array) строк (string), созданный делением параметра `string` по границам, указанным параметром `delimiter`.

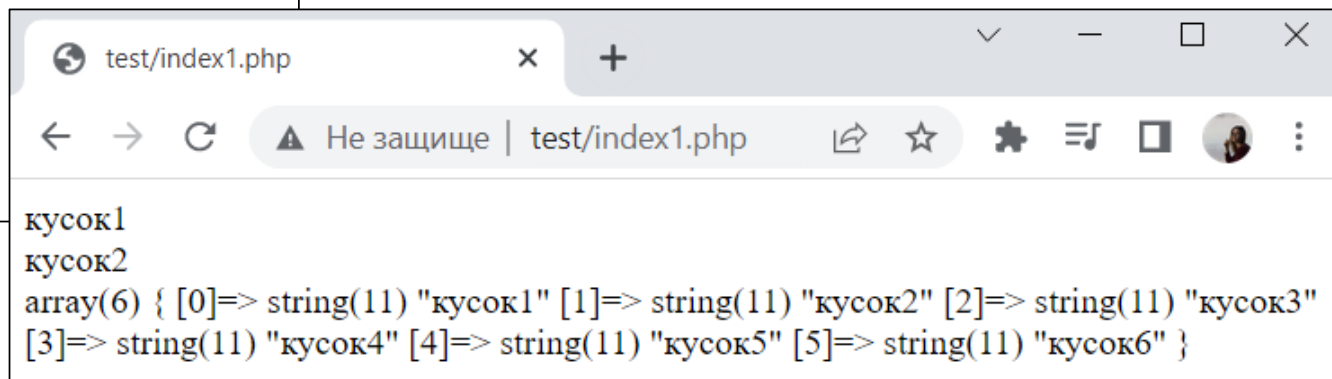
## Список изменений

Версия	Описание
8.0.0	<b>explode()</b> теперь выбрасывает <a href="#">TypeError</a> , если параметр <b>separator</b> является пустой строкой (""). Ранее вместо исключения <b>explode()</b> возвращала <b>false</b> .

# Функции для преобразования символьных переменных в массив

`array explode ( string $delimiter , string $string [, int $limit ] )`

```
<?php
// Пример 1
$pizza  = "кусок1 кусок2 кусок3 кусок4
кусок5 кусок6";
$pieces = explode(" ", $pizza);
echo $pieces[0]; // кусок1
echo "<br>";
echo $pieces[1]; // кусок2
echo "<br>";
var_dump ($pieces);
?>
```



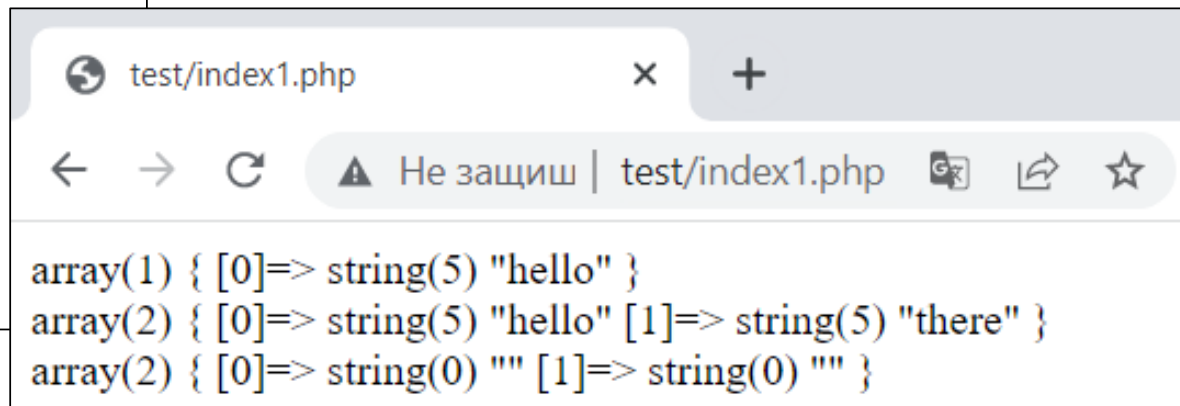
# Функции для преобразования символьных переменных в массив

```
array explode ( string $delimiter , string $string [, int $limit ] )
```

```
<?php
```

```
/*  
    Строка, которая не содержит разделителя,  
    будет просто возвращать массив с одним  
    значением оригинальной строки.  
*/
```

```
$input1 = "hello";  
$input2 = "hello,there";  
$input3 = ',';  
var_dump( explode( ',', $input1 ) );  
var_dump( explode( ',', $input2 ) );  
var_dump( explode( ',', $input3 ) );  
?>
```

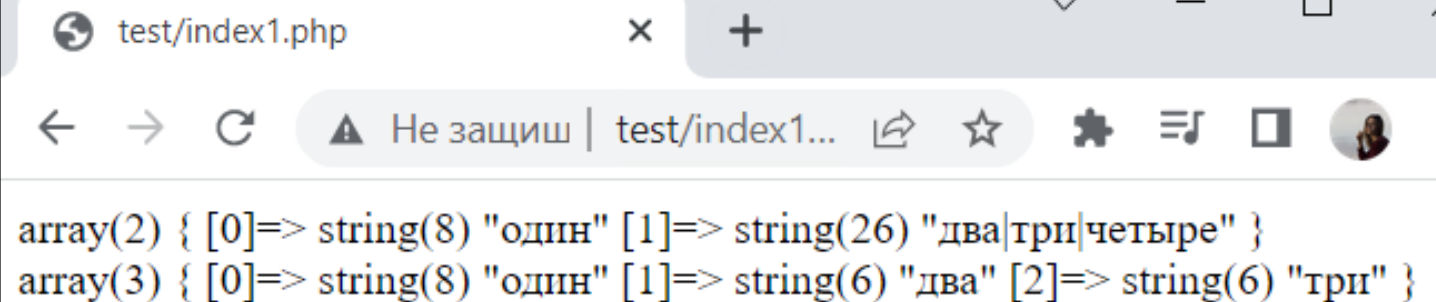


```
test/index1.php  
array(1) { [0]=> string(5) "hello" }  
array(2) { [0]=> string(5) "hello" [1]=> string(5) "there" }  
array(2) { [0]=> string(0) "" [1]=> string(0) "" }
```

# Функции для преобразования символьных переменных в массив

```
array explode ( string $delimiter , string $string [, int $limit ] )
```

```
<?php
$str = 'один|два|три|четыре';
var_dump( explode('|', $str, 2) );
echo "<br>";
var_dump( explode('|', $str, -1) );
echo "<br>";
?>
```



```
array(2) { [0]=> string(8) "один" [1]=> string(26) "два|три|четыре" }
array(3) { [0]=> string(8) "один" [1]=> string(6) "два" [2]=> string(6) "три" }
```

# Функции для преобразования символьных переменных в массив

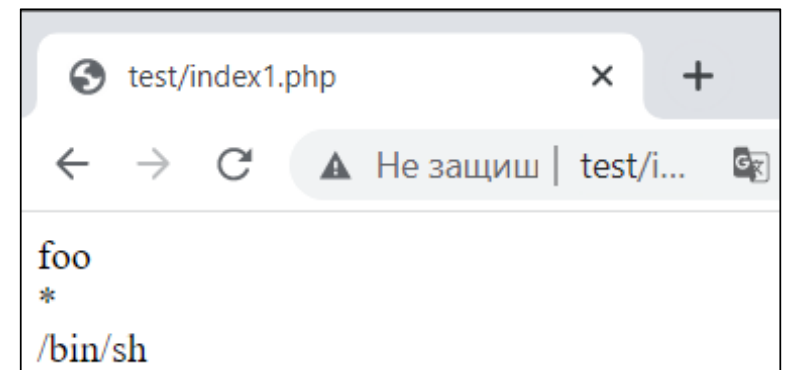
```
array list ( mixed $var1 [, mixed $... ] )
```

list — присваивает переменным из списка значения подобно массиву

Подобно array(), это не функция, а языковая конструкция.

list() используется для того, чтобы присвоить списку переменных значения за одну операцию.

```
<?php  
$data = "foo*:1023:1000::/home/foo:/bin/sh";  
list($user, $pass, $uid, $gid, $gecos, $home,  
$shell) = explode(":", $data);  
echo $user, "<br>"; // foo  
echo $pass, "<br>"; // *  
echo $shell, "<br>"; // /bin/sh  
?>
```



```
array list ( mixed $var1 [, mixed $... ] )
```

```
<?php
```

```
$info = array('кофе', 'коричневый', 'кофеин');
```

```
// Составить список всех переменных
```

```
list($drink, $color, $power) = $info;
```

```
echo "$drink - $color, а $power делает его особенным.\n";
```

```
// Составить список только некоторых из них
```

```
list($drink, , $power) = $info;
```

```
echo "В $drink есть $power.\n";
```

```
// Или только третья
```

```
list( , , $power) = $info;
```

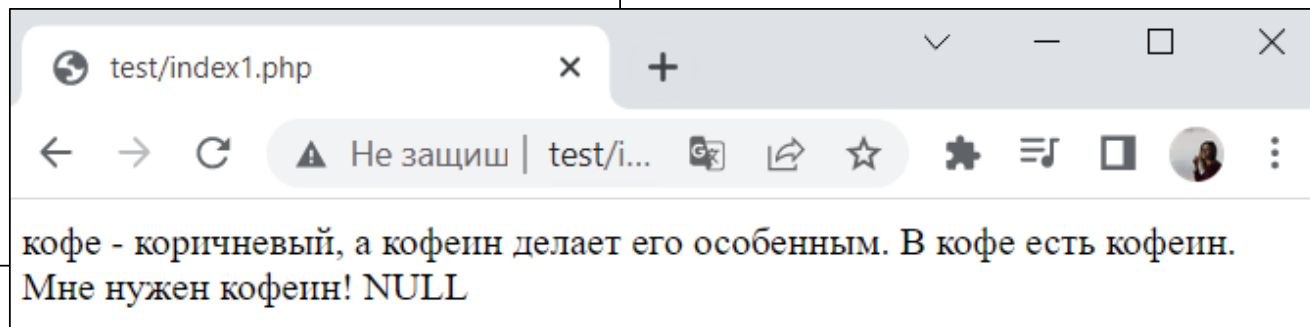
```
echo "Мне нужен $power!\n";
```

```
// list() не работает со строками
```

```
list($bar) = "abcde";
```

```
var_dump($bar); // NULL
```

```
?>
```



# Функции для преобразования массива строк в символьную переменную (строку)

**`implode(string $separator, array $array)`**

Объединяет элементы массива в строку. Список параметров:

**`$separator`**

Необязательный. По умолчанию равен пустой строке.

**`$array`**

Массив объединяемых строк.

Возвращает строку, содержащую строковое представление всех элементов массива в указанном порядке, с разделителем между каждым элементом.

## Список изменений

Версия	Описание
8.0.0	Передача <b><code>separator</code></b> после <b><code>array</code></b> больше не поддерживается.
7.4.0	Передача <b><code>separator</code></b> после <b><code>array</code></b> (т.е. использование недокументированного порядка параметров) устарела.

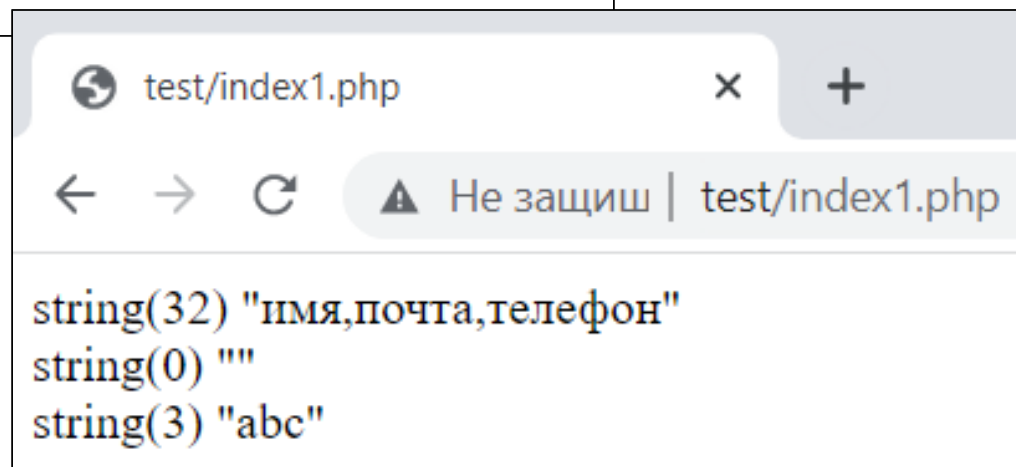


**implode(string \$separator, array \$array)**

```
<?php
$array = ['имя', 'почта', 'телефон'];
var_dump(implode(",", $array)); // string(32) "имя,почта,телефон"

// Пустая строка при использовании пустого массива:
var_dump(implode('привет', [])); // string(0) ""

// Параметр separator не обязателен:
var_dump(implode(['a', 'b', 'c'])); // string(3) "abc"
?>
```



# Функции для экранирования специальных СИМВОЛОВ

**quotemeta(string \$string)**

Возвращает модифицированную строку, в которой перед каждым символом из следующего списка:

`.\+*?[^]($)`

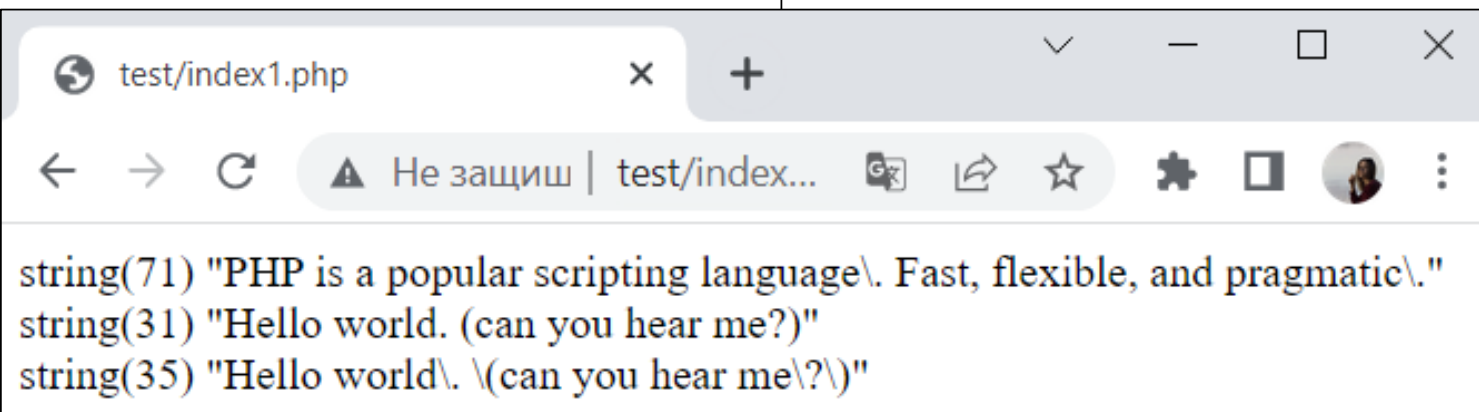
вставлен обратный слеш (\).

Возвращает экранированную строку или **false**, если в качестве параметра **string** была указана пустая строка.

# Функции для экранирования специальных СИМВОЛОВ

**quotemeta(string \$string)**

```
<?php
var_dump(quotemeta('PHP is a popular scripting language.
Fast, flexible, and pragmatic.'));
$str = "Hello world. (can you hear me?)";
var_dump($str);
echo "<br>";
$str1= quotemeta($str);
var_dump($str1);
?>
```



# Функции для работы с отдельными символами

## **`chr(int $code)`**

Данная функция возвращает строку, состоящую из символа с кодом `$code`.

## **`ord(int $char)`**

Данная функция возвращает код символа `$char`.

# Таблица ASCII

## ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	\$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

# Функции для работы с отдельными символами

```
<?php
for ($i=1; $i<=255; $i++)
{
    $symbol=chr($i);
    echo "код  $i  -  СИМВОЛ  $symbol", "<br>";
}
?>
```

```
test/index1.php x +
← → ↻ ⚠ Не защищ | test/index1.php
код 33 - СИМВОЛ !
код 34 - СИМВОЛ "
код 35 - СИМВОЛ #
код 36 - СИМВОЛ $
код 37 - СИМВОЛ %
код 38 - СИМВОЛ &
код 39 - СИМВОЛ '
код 40 - СИМВОЛ (
код 41 - СИМВОЛ )
код 42 - СИМВОЛ *
код 43 - СИМВОЛ +
код 44 - СИМВОЛ ,
код 45 - СИМВОЛ -
код 46 - СИМВОЛ .
код 47 - СИМВОЛ /
код 48 - СИМВОЛ 0
код 49 - СИМВОЛ 1
код 50 - СИМВОЛ 2
код 51 - СИМВОЛ 3
код 52 - СИМВОЛ 4
код 53 - СИМВОЛ 5
код 54 - СИМВОЛ 6
код 55 - СИМВОЛ 7
```

# Функции удаления пробелов

```
trim(string $string, string $characters = " \n\r\t\v\x00")
```

Удаляет пробелы (или другие символы) из начала и конца строки.

Эта функция возвращает строку **\$string** с удалёнными из начала и конца строки пробелами. Если второй параметр не передан, trim() удаляет следующие символы:

" " (ASCII 32 (0x20)), обычный пробел.

"\t" (ASCII 9 (0x09)), символ табуляции.

"\n" (ASCII 10 (0x0A)), символ перевода строки.

"\r" (ASCII 13 (0x0D)), символ возврата каретки.

"\0" (ASCII 0 (0x00)), NUL-байт.

"\v" (ASCII 11 (0x0B)), вертикальная табуляция.

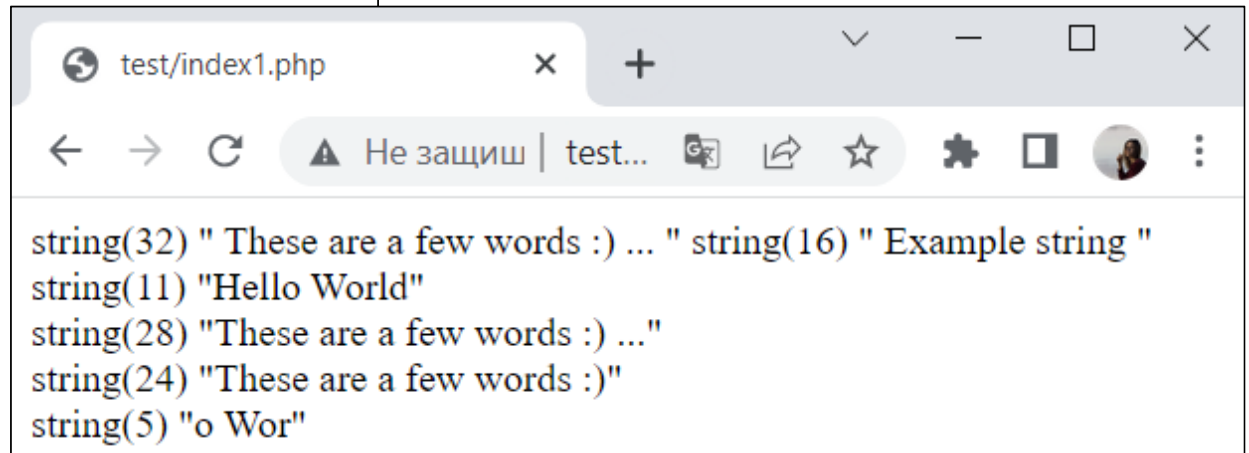
```
trim(string $string, string $characters = " \n\r\t\v\x00")
```

```
<?php
$text    = "\t\tThese are a few words :) ... ";
$binary  = "\x09Example string\x0A";
$hello   = "Hello World";
var_dump($text, $binary, $hello);
echo "<br>";

$trimmed = trim($text);
var_dump($trimmed);
echo "<br>";

$trimmed = trim($text, " \t.");
var_dump($trimmed);
echo "<br>";

$trimmed = trim($hello, "Hdle");
var_dump($trimmed);
echo "<br>";
?>
```





# Функции удаления пробелов

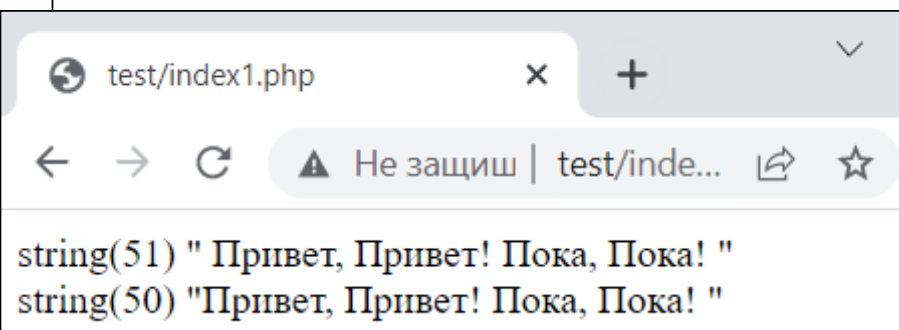
## **ltrim(string \$st)**

То же, что и trim(), только удаляет исключительно ведущие пробелы, а концевые не трогает.

## **chop(string \$st)**

Удаляет только концевые пробелы, ведущие не трогает.

```
<?php
$a="\tПривет, Привет!\t\t Пока, Пока!\t";
var_dump ($a);
echo "<br>";
$b=ltrim($a);
var_dump ($b);
?>
```

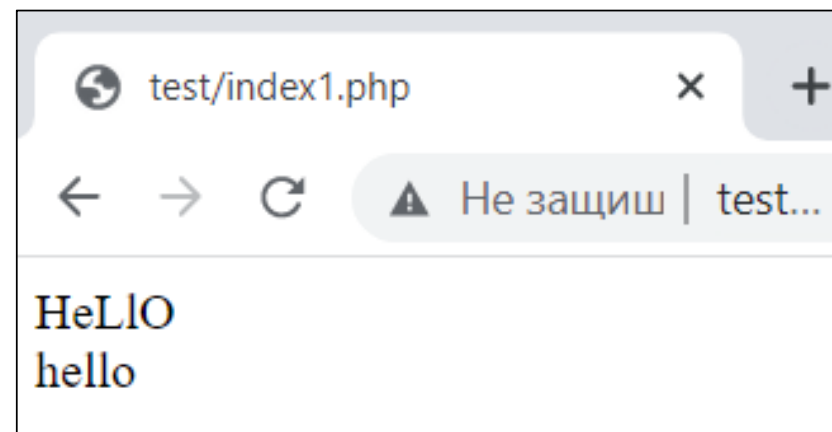


# Функции преобразования символов

## `strtr(string $str, string $from, string $to)`

Эта функция применяется не столь широко, но все-таки иногда она бывает довольно полезной. Она заменяет в строке `$str` все символы, встречающиеся в `$from`, на их "парные" (то есть расположенные в тех же позициях, что и во `$from`) из `$to`.

```
<?php
$addr="HeLlO";
echo $addr;
echo "<br>";
$addr=strtr($addr,"ABCDEFGHIJKLMNOPQRSTUVWXYZ",
"abcdefghijklmnopqrstuvwxyz");
echo $addr;
?>
```



# Функции изменения регистра

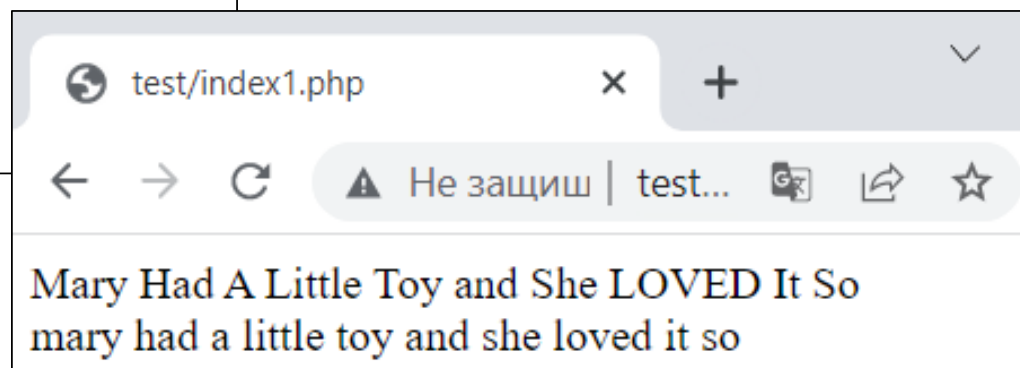
## **strtolower(string \$str)**

Преобразует строку в нижний регистр.

## **strtoupper(string \$str)**

Переводит строку в верхний регистр.

```
<?php
$str = "Mary Had A Little Toy and She LOVED It So";
echo $str, "<br>";
$str = strtolower($str);
echo $str;
?>
```



# Установка локали (локальных настроек)

Для установки локали используется функция `SetLocale()`:

**`SetLocale(string $category, string $locale)`**

Функция устанавливает текущую локаль, с которой будут работать функции преобразования регистра символов, вывода даты-времени и т.д.

То, какую именно категорию функций затронет вызов `SetLocale()`, задается в параметре **`$category`**. Он может принимать следующие строковые значения:

- `LC_CTYPE` — активизирует указанную локаль для функций перевода в верхний/нижний регистры;
- `LC_NUMERIC` — активизирует локаль для функций форматирования дробных чисел — а именно, задает разделитель целой и дробной части в числах;
- `LC_TIME` — задает формат вывода даты и времени по умолчанию;
- `LC_ALL` — устанавливает все вышеперечисленные режимы.

# Функции форматных преобразований

```
print ( string $arg )
```

Аналог `echo`. Единственное отличие от *echo* в том, что *print* принимает только один аргумент.

```
printf ( string $format [, mixed $args [, mixed $... ]] )
```

Выводит строку, отформатированную в соответствии с аргументом `format`.

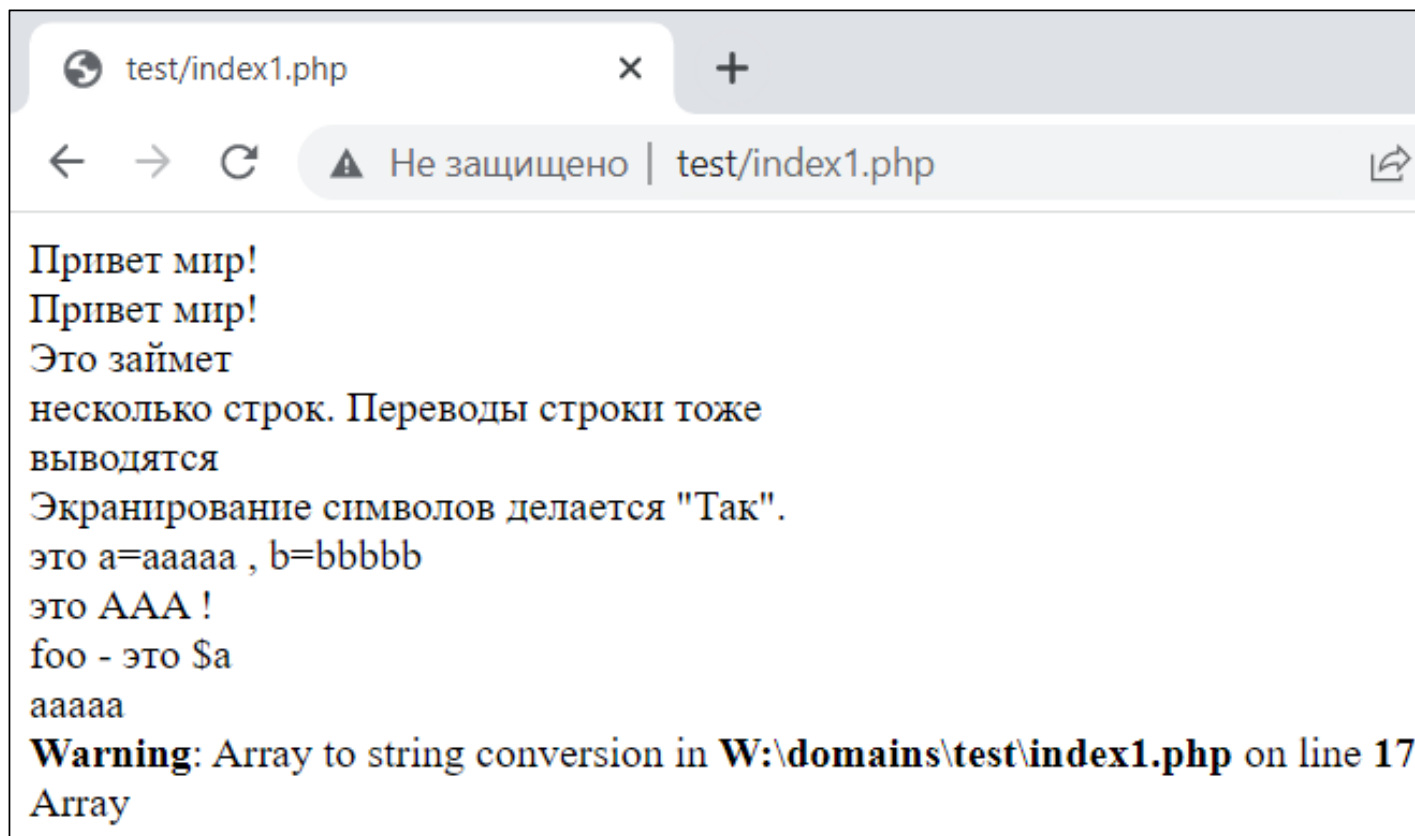
```
sprintf(string $format [, mixed args, ...])
```

Она возвращает строку, составленную на основе строки форматирования, содержащей некоторые специальные символы, которые будут впоследствии заменены на значения соответствующих переменных из списка аргументов.

# Функции форматных преобразований

```
<?php
print("Привет мир!");
print "<br> Привет мир!";
print "<br>Это займет<br>несколько строк. Переводы
строки тоже<br>выводятся";
print "<br>Экранирование символов делается \"Так\".";
// с print можно использовать переменные ...
$a = "aaaaa";
$b = "bbbbbb";
print "<br>это a=$a , b=$b";
// ... и массивы
$bar = array("value" => "AAA", "type" => "BBB");
print "<br>это {$bar['value']} !";
// При использовании одиночных кавычек выводится имя
переменной, а не значение
print '<br>foo - это $a<br>';
// Если вы не используете другие символы, можно вывести
просто значения переменных
print $a;
print $bar; // ошибка
?>
```

# Функции форматных преобразований



The screenshot shows a web browser window with a single tab titled 'test/index1.php'. The address bar shows 'test/index1.php' with a warning icon and the text 'Не защищено' (Not secure). The main content area displays the output of a PHP script, which includes several lines of text and a warning message. The text is as follows:

```
Привет мир!  
Привет мир!  
Это займет  
несколько строк. Переводы строки тоже  
выводятся  
Экранирование символов делается "Так".  
это a=aaaaa , b=bbbbbb  
это AAA!  
foo - это $a  
aaaaa  
Warning: Array to string conversion in W:\domains\test\index1.php on line 17  
Array
```

# Функции форматных преобразований

Каждый спецификатор формата включает максимум пять элементов (в порядке их следования после символа %):

>>> Необязательный спецификатор размера поля, который указывает, сколько символов будет отведено под выводимую величину. В качестве символов-заполнителей (если значение имеет меньший размер, чем размер поля для его вывода) может использоваться пробел или 0, по умолчанию подставляется пробел. Можно задать любой другой символ-наполнитель, если указать его в строке форматирования, предварив апострофом '.

>>> Опциональный спецификатор выравнивания, определяющий, будет результат выровнен по правому или по левому краю поля. По умолчанию производится выравнивание по правому краю, однако можно указать и левое выравнивание, задав символ - (минус).

>>> Необязательное число, определяющее размер поля для вывода величины. Если результат не будет в поле помещаться, то он "вылезет" за края этого поля, но не будет усечен.

>>> Необязательное число, предваренное точкой ".", предписывающее, сколько знаков после запятой будет в результирующей строке. Этот спецификатор учитывается только в том случае, если происходит вывод числа с плавающей точкой, в противном случае он игнорируется.

>>> Наконец, обязательный (заметьте — единственный обязательный!) спецификатор типа величины, которая будет помещена в выходную строку.

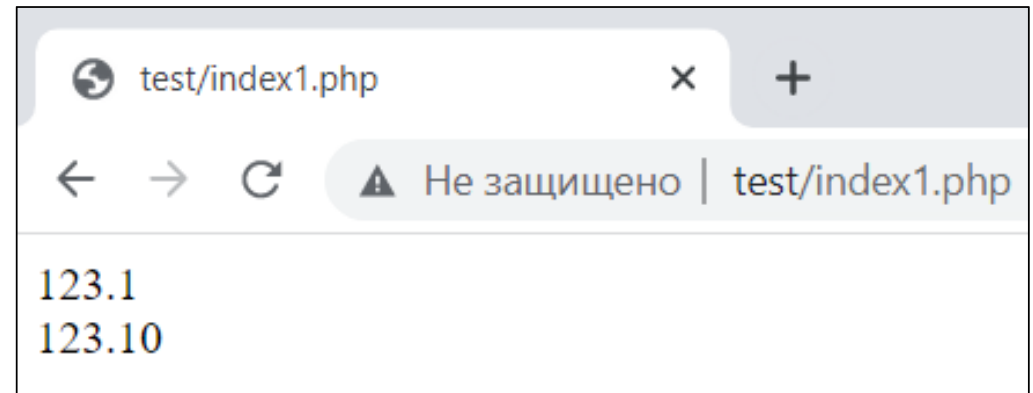


# Функции форматных преобразований

- b** — очередной аргумент из списка выводится как двоичное целое число;
- c** — выводится символ с указанным в аргументе кодом;
- d** — целое число;
- f** — число с плавающей точкой;
- o** — восьмеричное целое число;
- s** — строка символов;
- x** — шестнадцатеричное целое число с маленькими буквами *a-z*;
- X** — шестнадцатеричное число с большими буквами *A—Z*.

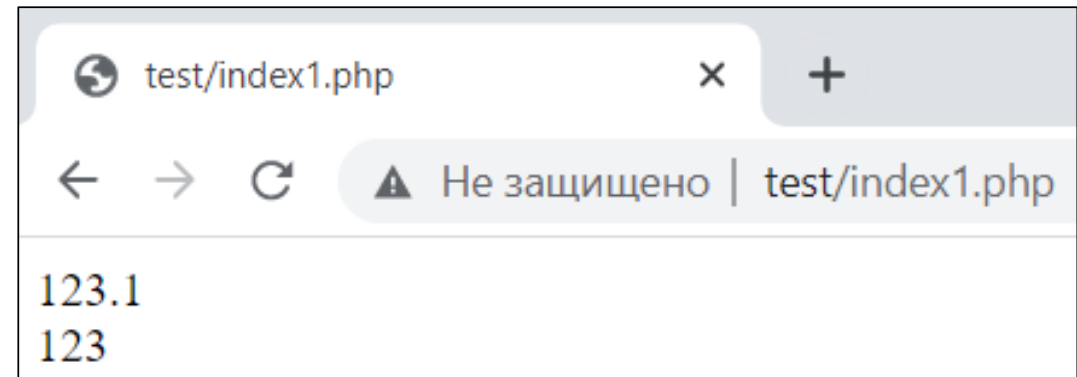
# Форматирование чисел с плавающей точкой

```
<?php
$money1 = 68.75;
$money2 = 54.35;
$money = $money1 + $money2;
echo $money, "<br>";
// echo $money выведет "123.1"...
$formatted = sprintf("%.2f", $money);
// echo $formatted выведет "123.10"!
echo $formatted;
?>
```



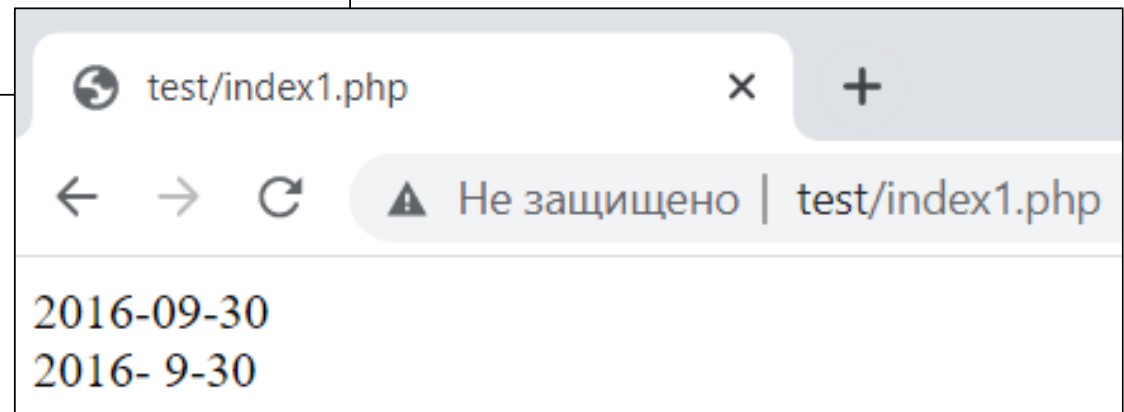
# Форматирование чисел с плавающей точкой

```
<?php
$money1 = 68.75;
$money2 = 54.35;
$money = $money1 + $money2;
echo $money, "<br>";
// echo $money выведет "123.1"...
$formatted = sprintf ("%d", $money);
echo $formatted;
?>
```



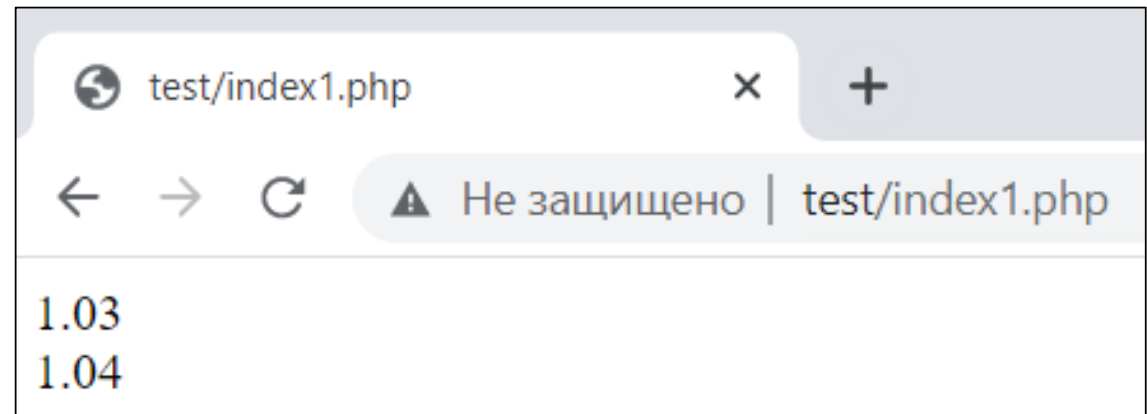
# Форматирование чисел с плавающей точкой

```
<?php
$year = 2016;
$month = 9;
$day = 30;
$isodate=sprintf("%04d-%02d-%02d",$year,$month,$day);
echo $isodate;
echo "<br>";
$isodate=sprintf("%4d-%2d-%2d",$year,$month,$day);
echo $isodate;
?>
```



# Форматирование чисел с плавающей точкой

```
<?php
printf("%.2f", 1.035);
echo "<br>";
printf("%.2f", round(1.035, 2));
?>
```



# Форматирование чисел с плавающей точкой

```
number_format(float $num, int $decimals = 0,  
string $decimal_separator = ".", string $thousands_separator = ",")
```

Список параметров:

**num** - формируемое число.

**decimals** - устанавливает число знаков после запятой. Если 0, **decimal\_separator** опускается в возвращаемом значении.

**decimal\_separator** - устанавливает разделитель дробной части.

**thousands\_separator** - устанавливает разделитель тысяч.

## Список изменений

Версия	Описание
8.0.0	До этой версии функция <b>number_format()</b> принимала один, два или четыре параметра (но не три).
7.2.0	<b>number_format()</b> была изменена, чтобы не возвращать -0, ранее -0 могло быть возвращено в случаях, когда <b>num</b> был -0.01.

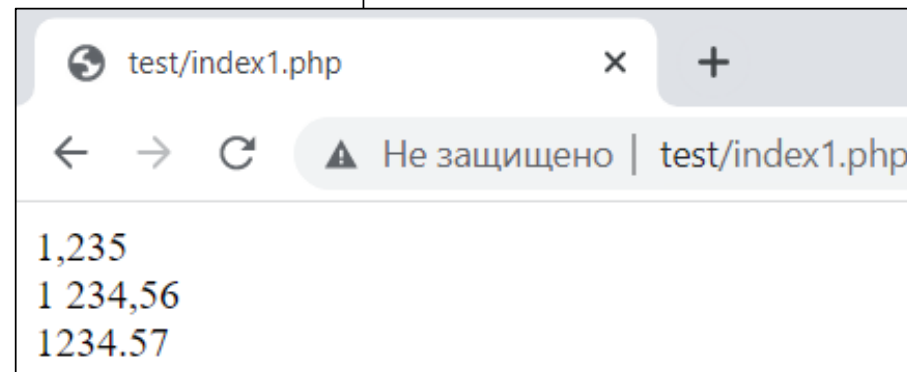
# Форматирование чисел с плавающей точкой

```
<?php
$number = 1234.56;

// английский формат (по умолчанию)
$english_format_number = number_format($number);
// 1,235
print $english_format_number . "<br>";

// французский формат
$nombre_format_francais = number_format($number, 2, ',', ' ');
// 1 234,56
print $nombre_format_francais . "<br>";

$number = 1234.5678;
// английский формат без разделителей групп
$english_format_number = number_format($number, 2, '.', '');
// 1234.57
print $english_format_number . "<br>";
?>
```



# Хэш-функции

## **md5(string \$str)**

Возвращает хэш-код строки \$str, основанный на алгоритме корпорации RSA Data Security под названием "MD5 Message-Digest Algorithm".

## **crc32(string \$str)**

Функция crc32() вычисляет 32-битную контрольную сумму строки \$str. Эта функция работает гораздо быстрее md5(), но в то же время выдает гораздо менее надежные "хэш-коды" для строки.

## **sha1 ( string \$str [, bool \$raw\_output = false ] )**

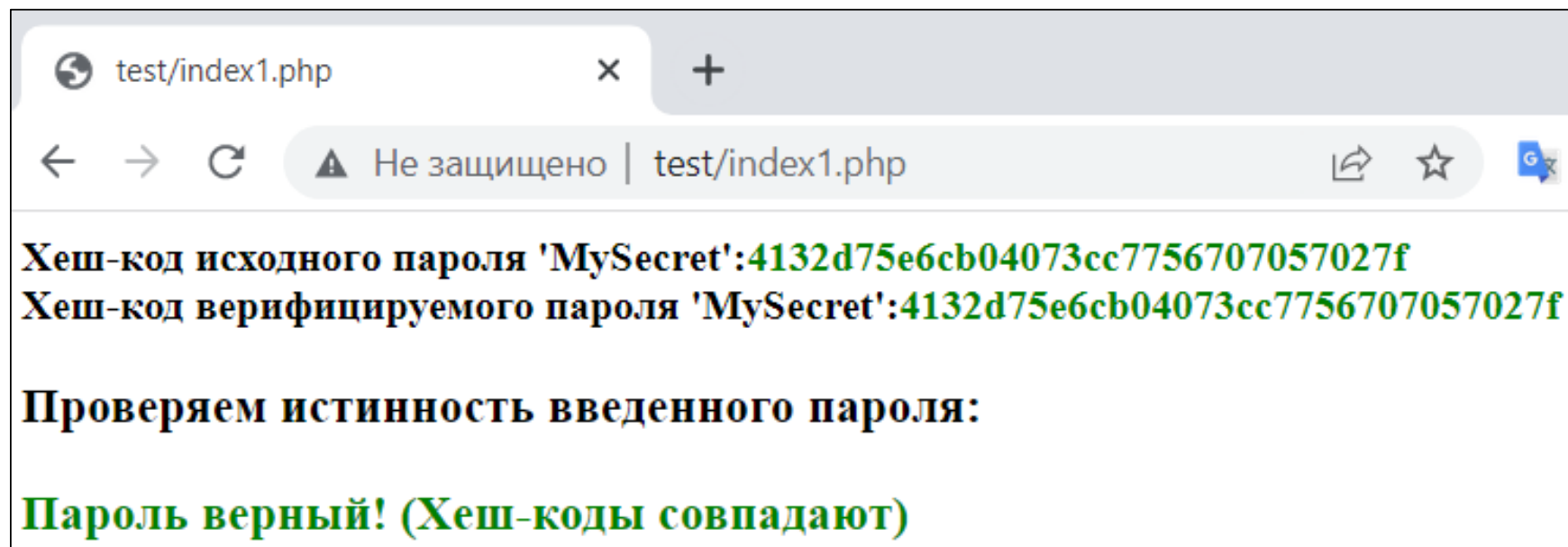
Возвращает SHA1-хэш строки str, вычисленный по алгоритму US Secure Hash Algorithm 1. Если необязательный аргумент raw\_output имеет значение TRUE, хэш возвращается в виде бинарной строки из 20 символов, иначе он будет возвращен в виде 40-символьного шестнадцатеричного числа.



# Хэш-функции

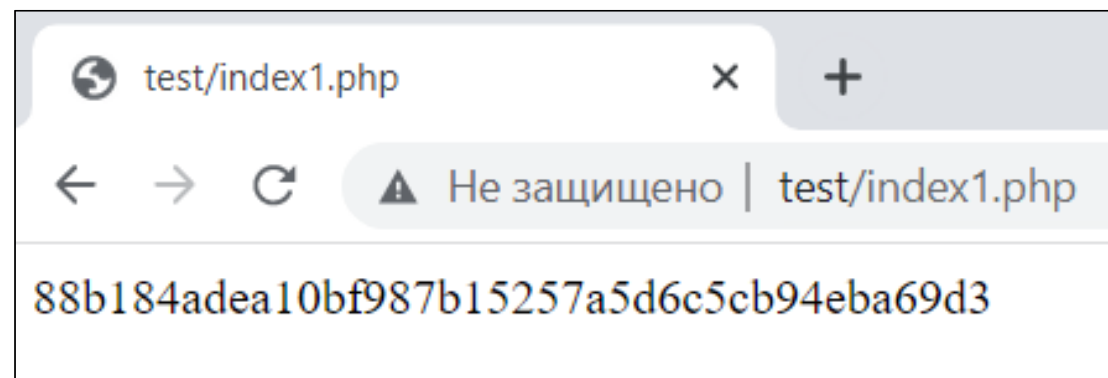
```
<?php
$pass_a = "MySecret";
$pass_b = "MySecret";
// Выводим хеш-код строки MySecret ($pass_a) - исходный пароль
echo "<b>Хеш-код исходного пароля '$pass_a':</b><b
style=\"color:green\">".md5($pass_a)."</b><br>";
// Выводим хеш-код строки MySecret ($pass_b) - верифицируемый пароль
echo "<b>Хеш-код верифицируемого пароля '$pass_b':</b><b
style=\"color:green\">".md5($pass_b)."</b><br>";
// Сравниваем хеш-коды MD5 исходного и верифицируемого пароля
echo "<h3>Проверяем истинность введенного пароля:</h3>";
if (md5($pass_a)===md5($pass_b)) echo "<h3 style=\"color:green\">Пароль
верный! (Хеш-коды совпадают)</h3>";
else echo "<h3 style=\"color:red\">Пароль неверный! (Хеш-коды не
совпадают)</h3>";
// В данной ситуации выводит: Пароль верный! (Хеш-коды совпадают)
// Попробуйте изменить значение строки $pass_b :)
?>
```

# Хэш-функции



# Хэш-функции

```
<?php  
$str = 'яблоко';  
echo sha1($str);  
?>
```



## Хэш-функции (для файла)

**md5\_file ( string \$filename [, bool \$raw\_output = false ] )**

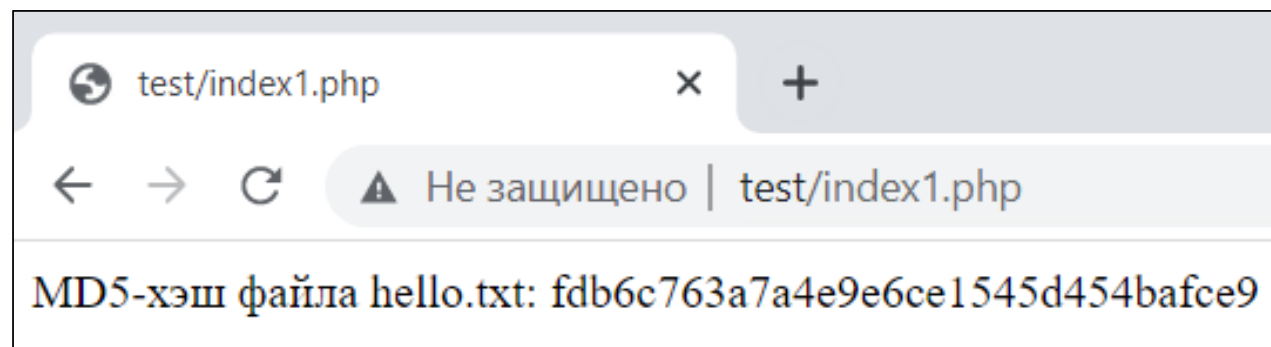
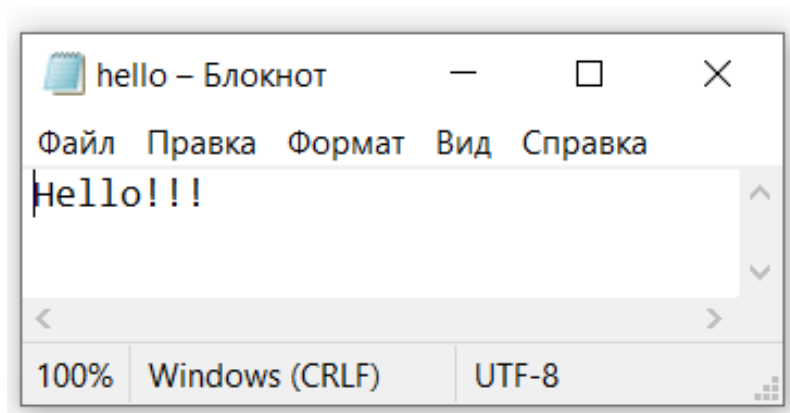
Вычисляет MD5-хэш файла, имя которого задано аргументом filename, используя алгоритм MD5 RSA Data Security, Inc. и возвращает этот хэш. Хэш представляет собой 32-значное шестнадцатеричное число. Если raw\_output имеет значение TRUE, то возвращается бинарная строка из 16 символов.

**sha1\_file ( string \$filename [, bool \$raw\_output = false ] )**

Вычисляет SHA1-хэш файла, имя которого задано аргументом filename, используя алгоритм US Secure Hash Algorithm 1 и возвращает этот хэш. Если raw\_output установлен в TRUE, возвращает 20-символьный хэш в бинарном формате.

# Хэш-функции (для файла)

```
<?php
$file = "hello.txt";
echo 'MD5-хэш файла ' . $file . ': ' . md5_file($file);
?>
```



# Хэш-функции (для файла)

```
<?php
$file = "hello.txt";
echo 'SHA-хэш файла ' . $file . ': ' . Sha1_file($file);
?>
```

