#### Тип string (строки)

***Строка*** – это набор символов. В PHP символ – это то же самое, что байт, это значит, что существует ровно 256 различных символов. Это также означает, что PHP не имеет встроенной поддержки Unicode. В PHP практически не существует ограничений на размер *строк*, поэтому нет абсолютно никаких причин беспокоиться об их длине.

*Строка* в PHP может быть определена тремя различными *способами*:

* с помощью *одинарных кавычек*;
* с помощью *двойных кавычек*;
* *heredoc-синтаксисом*.

##### Одинарные кавычки

Простейший *способ* определить *строку* – это заключить ее в *одинарные кавычки* " ' ". Чтобы использовать *одинарную кавычку* внутри *строки*, как и во многих других языках, перед ней необходимо поставить символ обратной косой черты " \ ", т. е. экранировать ее. Если обратная косая черта должна идти перед *одинарной кавычкой* либо быть в конце *строки*, необходимо продублировать ее " \\' ".

Если внутри *строки*, заключенной в *одинарные кавычки*, обратный слэш " \ " встречается перед любым другим символом (отличным от " \ " и " ' " ), то он рассматривается как обычный символ и выводится, как и все остальные. Поэтому обратную косую черту необходимо экранировать, только если она находится в конце *строки*, перед закрывающей кавычкой.

В PHP существует ряд комбинаций символов, начинающихся с символа обратной косой черты. Их называют ***управляющими последовательностями***, и они имеют специальные значения, о которых мы расскажем немного позднее. Так вот, в отличие от двух других синтаксисов, *переменные* и *управляющие последовательности* для специальных символов, встречающиеся в *строках*, заключенных в *одинарные кавычки*, не *обрабатываются* .

<?php

// Выведет: Чтобы вывести ' надо

// перед ней поставить \

echo 'Чтобы вывести \' надо перед ней поставить \\';

// Выведет: Вы хотите удалить C:\\*.\*?

echo 'Вы хотите удалить C:\\\*.\*?';

// Выведет: Это не вставит: \n новую строку

echo 'Это не вставит: \n новую строку';

// Выведет: Переменные $expand также $either не подставляются

echo 'Переменные $expand также $either не подставляются';

?>

##### Двойные кавычки

Если *строка* заключена в *двойные кавычки* " " ", PHP распознает большее количество *управляющих последовательностей* для специальных символов. Некоторые из них приведены в таблице:

|  |  |
| --- | --- |
| **Последовательность** | **Значение** |
| \n | Новая строка |
| \r | Возврат каретки |
| \t | Горизонтальная табуляция |
| \\ | Обратная косая черта |
| \$ | Знак доллара |
| \" | Двойная кавычка |

Самым важным свойством *строк* в *двойных кавычках*является *обработка переменных*.

##### Heredoc

Другой *способ определения строк* – это использование ***heredoc-синтаксиса***. В этом случае *строка* должна начинаться с символа <<<, после которого идет идентификатор. Заканчивается *строка* этим же идентификатором. Закрывающий идентификатор должен начинаться в первом столбце *строки*. Кроме того, идентификатор должен соответствовать тем же правилам именования, что и все остальные метки в PHP: содержать только буквенно-цифровые символы и знак подчеркивания и начинаться не с цифры или знака подчеркивания.

*Heredoc* -текст ведет себя так же, как и *строка* в *двойных кавычках*, при этом их не имея. Это означает, что вам нет необходимости экранировать кавычки в *heredoc*, но вы по-прежнему можете использовать перечисленные выше *управляющие последовательности*. *Переменные* внутри *heredoc* тоже *обрабатываются*.

<?php

$str = <<<EOD

Пример строки, охватывающей несколько

строчек, с использованием

heredoc-синтаксиса

EOD;

// Здесь идентификатор – EOD. Ниже

// идентификатор EOD

$name = 'Вася';

echo <<<EOD

Меня зовут "$name".

EOD;

// это выведет: Меня зовут "Вася".

?>

**Функции для обработки строк в PHP**

Функции для обработки строк перечислены ниже:

|  |  |
| --- | --- |
| chr | Возвращает символ по его коду ASCII |
| chunk\_split | Разбивает строку на подстроки заданной длины |
| crypt | Зашифровывает строку с использованием одного из алгоритмов |
| [echo](http://php720.com/lesson/view-lesson.php) | Выводит одну или несколько строк |
| explode | Разбивает строку на подстроки, ограниченные заданным разделителем, и форматирует из них массив |
| html\_entity\_decode | Декодирует все HTML-представления в соответствующие символы. Функция обратно по отношению к htmlentites |
| htmlentites | Кодирует все специальные символы в их HTML-представление |
| htmlspecialchars | Кодирует все символы в их HTML-представление |
| implode | Формирует строку из элементов массива |
| ltrim | Удаляет начальные пробелы из строки |
| rtrim | Удаляет конечные пробелы из строки |
| number\_format | Представляет число в виде строки в различных форматах |
| ord | Возвращает ASCII-код символа |
| parse\_str | Разбивает строку URL и присваивает значение переменным |
| print | Выводит строку |
| printf | Выводит строку с форматированием |
| sprintf | Возвращает строку с форматированием |
| setlocale | Устанавливает информацию о кодовой странице |
| similar\_text | Вычисляет степень похожести двух строк |
| sscanf | Разбивает строку по шаблону и присваивает полученные значения переменным |
| str\_ireplace | То же самое, что и str\_replace, но без учета различий в регистре символов |
| str\_pad | Дополняет строку до заданной длины другой строкой |
| str\_repeat | Повторяет строку заданное количество раз |
| str\_replace | Ищет в строке все вхождения подстроки и меняет на заданную строку |
| str\_shuffle | Случайным образом перемешивает все символы в строке |
| str\_split | Формирует массив из символов строки |
| str\_word\_count | Подсчитывает количество слов в строке |
| strcasecmp | Выполняет побайтовое сравнение строк без учета регистра символов |
| strcht | То же самое что strstr |
| strcmp | Выполняет побайтовое сравнение строк с учетом регистра символов |
| strip\_tags | Удаляет из строки все HTML-и PHP-теги |
| stripos | Ищет первое вхождение подстроки в строке без учета регистра символов |
| stristr | То же самое что strstr, но без учета регистра символов |
| strlen | Возвращает длину строки |
| strnatcasecmp | То же самое что strnatcmp, но без учета регистра символов |
| strncmp | Выполняет побайтовое сравнение первых n символов строк |
| strpos | Ищет первое вхождение подстроки в строке |
| strrchr | Ищет последнее вхождение символа в строке |
| strrev | Инвертирует строку - прочитывает ее справа налево |
| strripos | Ищет последнее вхождение подстроки в строке без учета регистра символов |
| strrpos | Ищет последнее вхождение подстроки в строке |
| strspn | Возвращает длину участка строки, состоящего из заданных символов |
| strstr | Возвращает часть строки от первого вхождения подстроки до конца |
| strtolower | Преобразует прописные буквы в строчные |
| strtoupper | Преобразует строчные буквы в прописные |
| strtr | Преобразует заданные символы в строке |
| substr\_compare | Сравнивает две строки, начиная с заданного смещения |
| substr\_count | Подсчитывает, сколько раз заданная подстрока встречается в строке |
| substr\_replace | Ищет в заданном участке строки все вхождения подстроки и меняет на другую строку |
| substr | Возвращает заданную часть исходной строки |
| trim | Удаляет начальные и конечные пробелы из строки |
| ucfirst | Преобразует первую букву строки в прописную |

# Использование строковых функций в PHP

### Базовые строковые функции

**strlen(string $st)**

Одна из наиболее полезных функций. Возвращает просто длину строки, т. е., сколько символов содержится в $st. Пример:

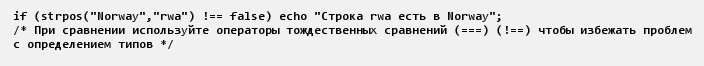
### Безимени.jpg

**strpos(string $where, string $what, int $fromwhere=0)**

Пытается найти в строке $where подстроку (то есть последовательность символов) $what и в случае успеха возвращает позицию (индекс) этой подстроки в строке. Необязательный параметр $fromwhere можно задавать, если поиск нужно вести не с начала строки $from, а с какой-то другой позиции. В этом случае следует эту позицию передать в $fromwhere. Если подстроку найти не удалось, функция возвращает false. Однако будьте внимательны, проверяя результат вызова strpos() на false — используйте для этого только оператор ===. Пример:

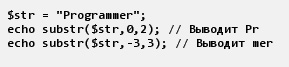
### Безимени-2.jpg

И еще пример:



**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. Однако эта функция может делать и еще довольно полезные вещи. К примеру, если мы передадим в $start отрицательное число, то будет считаться, что это число является индексом подстроки, но только отсчитываемым от конца $str (например, -1 означает "начиная с последнего символа строки"). Параметр $length, если он задан, тоже может быть отрицательным. В этом случае последним символом возвращенной подстроки будет символ из $str с индексом $length, определяемым от конца строки. Примеры:



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

Сравнивает две строки посимвольно (точнее, побайтово) и возвращает: 0, если строки полностью совпадают; -1, если строка $str1 лексикографически меньше $str2; и 1, 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 с блоками текста

Перечисленные ниже функции чаще всего оказываются полезны, если нужно проводить однотипные операции с многострочными блоками текста, заданными в строковых переменных.

**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. Например, вот так мы можем заместить все символы перевода строки на их HTML эквивалент — тэг <br>:

Безимени.jpg

Как видим, то, что в строке <br>\n тоже есть символ перевода строки, никак не влияет на работу функции, т. е. функция производит лишь однократный проход по строке. Для решения описанной задачи также применима функция nl2br(), которая работает чуть быстрее.

**string nl2br(string $string)**

Заменяет в строке все символы новой строки \n на <br>\n и возвращает результат. Исходная строка не изменяется. Обратите внимание на то, что символы \r, которые присутствуют в конце строки текстовых файлов Windows, этой функцией никак не учитываются, а потому остаются на старом месте.

**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. Примеры:

Безимени.jpg

### Функции для работы с отдельными символами

Как и в других языках программирования, в PHP можно работать с символами строк отдельно.

Обратиться к любому символу строки можно по его индексу:

Безимени.jpg

**chr(int $code)**

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

Безимени-2.jpg

**ord($char)**

Данная функция возвращает код символа $char. Вот пример:

Безимени-3.jpg

### Функции удаления пробелов

Иногда трудно даже представить, какими могут быть странными пользователи, если дать им в руки клавиатуру и попросить напечатать на ней какое-нибудь слово. Так как клавиша пробела — самая большая, то пользователи имеют обыкновение нажимать ее в самые невероятные моменты. Этому способствует также и тот факт, что символ с кодом 32, обозначающий пробел, как вы знаете, на экране не виден. Если программа не способна обработать описанную ситуацию, то она, в лучшем случае после тягостного молчания отобразит в браузере что-нибудь типа "неверные входные данные", а в худшем — сделает при этом что-нибудь необратимое.

Между тем, обезопасить себя от паразитных пробелов чрезвычайно просто, и разработчики PHP предоставляют нам для этого ряд специализированных функций. Не волнуйтесь о том, что их применение замедляет программу. Эти функции работают с молниеносной скоростью, а главное, одинаково быстро, независимо от объема переданных им строк.

**trim(string $str)**

Возвращает копию $str, только с удаленными ведущими и концевыми пробельными символами. Под пробельными символами я здесь и далее подразумеваю: пробел " ", символ перевода строки \n, символ возврата каретки \r и символ табуляции \t. Например, вызов trim(" test\n ") вернет строку "test". Эта функция используется очень широко. Старайтесь применять ее везде, где есть хоть малейшее подозрение на наличие ошибочных пробелов. Поскольку работает она очень быстро.

**ltrim(string $st)**

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

**chop(string $st)**

Удаляет только концевые пробелы, ведущие не трогает.

### Функции изменения регистра

Довольно часто нам приходится переводить какие-то строки, скажем, в верхний регистр, т. е. делать все прописные буквы в строке заглавными. В принципе, для этой цели можно было бы воспользоваться функцией strtr(), рассмотренной выше, но она все же будет работать не так быстро, как нам иногда хотелось бы. В PHP есть функции, которые предназначены специально для таких нужд. Вот они:

**strtolower(string $str)**

Преобразует строку в нижний регистр. Возвращает результат перевода.

Надо заметить, что при неправильной настройке локали (это набор правил по переводу символов из одного регистра в другой, переводу даты и времени, денежных единиц и т. д.) функция будет выдавать неправильные результаты при работе с буквами кириллицы.  
Возможно, в несложных программах, а также если нет уверенности в поддержке соответствующей локали операционной системой, проще будет воспользоваться "ручным" преобразованием символов, задействуя функцию strtr():

Безимени.jpg

Главное достоинство данного способа — то, что в случае проблем с кодировкой для восстановления работоспособности сценария вам придется всего лишь преобразовать его в ту же кодировку, в которой у вас хранятся документы на сервере.

**strtoupper(string $str)**

Переводит строку в верхний регистр. Возвращает результат преобразования. Эта функции также прекрасно работает со строками, составленными из латиницы, но с кирилицей может возникнуть все та же проблема.