Основы PHP

Функции для работы со строками

Базовые строковые функции

strlen(string $st)

Одна из наиболее полезных функций. Возвращает просто длину строки, т. е., сколько символов содержится в $st. Строка может содержать любые символы, в том числе и с нулевым кодом (что запрещено в Си).
Пример:

```
<?php
	$x = “Hello”;
	echo strlen($x); // Выводит 6
?>
```

Базовые строковые функции

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 ===.

```php
<?php
    echo strpos("Hello","el"); // Выводит 1
?>
```

Базовые строковые функции

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.
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").

Базовые строковые функции

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

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:

```
<?php
	$str = "Programmer";
	echo substr($str,0,2); // Выводит Pr
	echo substr($str,-3,3); // Выводит mer
?>
```

Базовые строковые функции

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

Использование отрицательного length.

```
<?php
	$rest = substr("abcdef", 0, -1); // возвращает "abcde"
	$rest = substr("abcdef", 2, -1);  // возвращает "cde"
	$rest = substr("abcdef", 4, -4);  // возвращает ""
	$rest = substr("abcdef", -3, -1); // возвращает "de"
?>
```

Базовые строковые функции

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 равнение идет побайтово, то регистр символов влияет на результаты сравнений.

strcmp(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

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.
Эта функция работает значительно быстрее, чем ereg_replace(), которая используется при работе с регулярными выражениями PHP, и ее часто используют, если нет необходимости в каких-то экзотических правилах поиска подстроки.
*Например, вот так мы можем заместить все символы перевода строки на их HTML эквивалент — тэг
:*

```
<?php $st = str_replace("\n","<br>\n",$str); ?>
```

Как видим, то, что в строке
\n тоже есть символ перевода строки, это никак не влияет на работу функции, т. е. функция производит лишь однократный проход по строке.

Функции для работы с блоками текста

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

Эта функция, появившаяся в PHP4, 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.

Функции для работы с блоками текста

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

Пример:

```
<?php
    $str = "Это текст эл-го письма, которое нужно будет отправить адресату...";
    // Разбиваем текст по 20 символов
    $str = WordWrap ($str, 20, "<br>");
    echo $str;
    // Выводит:
    /* Это текст
    электронного письма,
    которое нужно будет
    отправить
    адресату... */
?>
```

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.
Примеры:

```
<?php
	$stripped = strip_tags ($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 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:

```
$str = "PHP";
echo $str[0]; // Выводит 'P'
```

chr(int $code)

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

```
echo chr(75); //Выводит K
```

ord($char)

Данная функция возвращает код символа $char. Вот пример:

```
echo ord('A'); // Выводит 65 - код буквы 'A'
```

Функции удаления пробелов

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 преобразования символов

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

Эта функция 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.

UrlEncode(string $str)

Функция URL-кодирует строку $str и возвращает результат.
Эту функцию удобно применять, если вы, например, хотите динамически сформировать ссылку <a href=...> на какой-то сценарий, но не уверены, что его параметры содержат только алфавитно-цифровые символы.
В этом случае воспользуйтесь функцией так:

```
echo "<a href=/script.php?param=".UrlEncode($UserData);
```

Теперь, даже если переменная $UserData включает символы =, & или даже пробелы, все равно сценарию будут переданы корректные данные.

Функции изменения регистра

strtolower(string $str)

Эта функция преобразует строку в нижний регистр. Возвращает результат перевода.

strtoupper(string $str)

Данная функция переводит строку в верхний регистр. Возвращает результат преобразования.

Эти функции также прекрасно работает со строками, составленными из латиницы, но с кириллицей может возникнуть все та же проблема.

Хэш-функции

md5(string $str)

Возвращает хэш-код строки $str, основанный на алгоритме корпорации RSA DataSecurity под названием "MD5 Message-Digest Algorithm".
Хэш-код — это просто строка, практически уникальная для каждой из строк $str.
Если длина строки $str может достигать нескольких тысяч символов, то ее MD5-код занимает максимум 32 символа.

crc32(string $str)

Функция crc32() вычисляет 32-битную контрольную сумму строки $str. То есть, результат ее работы — 32 битное (4-байтовое) целое число. Эта функция работает гораздо быстрее md5(), но в то же время выдает гораздо менее надежные "хэш-коды" для строки.

Установка локали (локальных настроек)

Локалью будем называть совокупность локальных настроек системы, таких как формат даты и времени, язык, кодировка.
Настройки локали сильно зависят от операционной системы.

SetLocale(string $category, string $locale)

Функция устанавливает текущую локаль, с которой будут работать функции преобразования регистра символов, вывода даты-времени и т.д.
Вообще говоря, для каждой категории функций локаль определяется отдельно и выглядит по-разному.
То, какую именно категорию функций затронет вызов SetLocale(), задается в параметре $category.

Установка локали (локальных настроек)

Параметр $category 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.*

Установка локали (локальных настроек)

Теперь о параметре $locale. Как известно, каждая локаль, установленная в системе, имеет свое уникальное имя, по которому к ней можно обратиться. Именно оно и фиксируется в этом параметре.
Однако, есть два важных исключения из этого правила.

- *Во-первых, если величина $locale равна пустой строке "", то устанавливается та локаль, которая указана в глобальной переменной окружения с именем, совпадающим с именем категории $category (или LANG — она практически всегда присутствует в Unix).*
- *Во-вторых, если в этом параметре передается 0, то новая локаль не устанавливается, а просто возвращается имя текущей локали для указанного режима.*

Функции преобразования кодировок

convert_cyr_string(string $str, char $from, char $to)

Функция переводит строку $str из кодировки $from в кодировку $to.