http://kharchuk.ru/popular-preg

|  |  |  |  |
| --- | --- | --- | --- |
| Примеры часто используемых регулярных выражений | [PDF](http://kharchuk.ru/popular-preg?format=pdf) | [Печать](http://kharchuk.ru/popular-preg?tmpl=component&print=1&page=) | [E-mail](http://kharchuk.ru/component/mailto/?tmpl=component&link=aHR0cDovL2toYXJjaHVrLnJ1L3BvcHVsYXItcHJlZw%3D%3D) |

|  |
| --- |
| [Статьи](http://kharchuk.ru/home)- [PHP программирование](http://kharchuk.ru/home/7-PHP%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) |
| **Получить расширение файла**  preg\_replace("/.\*?\./", '', 'photo.jpg');    **Взять то, что находится между тегами <title> и </title>**  if (preg\_match('|<title.\*?>(.\*)</title>|sei', $str, $arr)) $title = $arr[1]; else $title='';  Обратите внимание: берется не нулевой элемент массива, а первый! Если title будет встречаться несколько раз, то будет вырезано от первого и до последнего!  **Найти текст, заключенный в какой-то тег и заменить его на другой тег**  Например: <TITLE> ... </TITLE> заменить аналогично на <МОЙ\_ТЕГ> ... </МОЙ\_ТЕГ> в HTML-файле:  preg\_replace("!<title>(.\*?)</title>!si","<МОЙ\_ТЕГ>\\1</МОЙ\_ТЕГ>",$string);    **Проверяем, является ли переменная числом**  if (!preg\_match("|^[\d]+$|", $var)) ...    **Запретим пользователю использовать в своем имени любые символы, кроме букв русского и латинского алфавита, знака "\_" (подчерк), пробела и цифр:**  if (preg\_match("/[^(\w)|(\x7F-\xFF)|(\s)]/",$username)) { echo "invalid username"; exit; }    **Проверка адреса e-mail**  Для поля ввода адреса e-mail добавим в список разрешенных символов знаки "@" и "." и "-", иначе пользователь не сможет корректно ввести адрес. Зато уберем русские буквы и пробел:  if (preg\_match("/[^(\w)|(\@)|(\.)|(\-)]/",$usermail)) { echo "invalid mail"; exit; }    **Проверка на число**  if(preg\_match('/^\d+$/', $var)) echo $var;    **Проверка имени файла**  if (preg\_match("/(^[a-zA-Z0-9]+([a-zA-Z\\_0-9\.-]\*))$/" , $filename)==NULL) { echo "invalid filename"; exit; }    **Проверка расширения файла**  Архивы (zip, rar, ...)  /\.(?:z(?:ip|[0-9]{2})|r(?:ar|[0-9]{2})|jar|bz2|gz|tar|rpm)$/i  Аудио (mp3, wav, ...)  /\.(?:mp3|wav|og(?:g|a)|flac|midi?|rm|aac|wma|mka|ape)$/i  Программы (exe, xpi, ...)  /\.(?:exe|msi|dmg|bin|xpi|iso)$/i  Изображения (jpg, png, ...)  /\.(?:jp(?:e?g|e|2)|gif|png|tiff?|bmp|ico)$/i  Видео (mpeg, avi, ...)  /\.(?:mpeg|ra?m|avi|mp(?:g|e|4)|mov|divx|asf|qt|wmv|m\dv|rv|vob|asx|ogm)$/i    **Выборка цен**  Часто возникает проблема по парсингу интересующих программиста данных из HTML, который не всегда хорошего качества, все было бы терпимо, если бы еще не вставки на javascript'е, вот пример такого текста:  <TD>20.02<BR>05:30 <TD class=l>Товар 1<BR>Товар 2 <TD><B>35</B> <TD><A href="http://ссылка/" id=sfsd32dfs onclick="return m(this)">26.92</A><BR><A href="http://ссылка/" id=r3\_3143svsfd onclick="return m(this)">27.05</A> <TD><B>270.5</B> </TR>  Те цифры, которые написаны через точку, являются ценами. Задача состоит в том, чтобы собрать все цены, которые находятся между тегами <a>... </a> Видим, что помимо цен между заданными тегами, есть такие, которые идут сразу после тега <TD>, а также стоят между тегами <B>...</B>. Ясно, что описать достаточно точно содержимое атрибутов тега <A> представляется задачей не самой легкой, поэтому надо ее упростить! Любой тег имеет закрывающий знак '>', наша задача описать, что этот знак идет перед ценой, но так как перед ценой может стоять тег <B> и тег <TD>, но эти цены нам не нужны. Каким образом мы узнаем, что цена стоит между тегами <A>...</A>? По тегу, который идет после цены, если это не тег </B>, то это будет либо тег </A> либо <BR>, а так же по тегу перед ценой если этот тег <TD>. Путем таких размышлений мы пришли к выводу, что должно стоять справа, а что должно стоять слева искомой строки, которая описывается как цифры, разделенные точкой: \d\*\.\d\* То, что должно совпасть слева, мы описали как символ '>', записываем: (?<=>) - выглядит немного странно, но совпадение справа записывается вот так (?<=), а внтури него после ?<= идет символ '>' То, что должно совпасть справа описывается (?=) внутри мы пишем </A>. Теперь опишем, что не должно стоять перед ценой: (?<!<TD>) перед ценой не должен стоять тег <TD>, это и есть негативная ретроспективная проверка. При помощи негативной опережающей проверки опишем, что не должно стоять справа цены: (?!<\/B>) справа от цены не должен стоять тег </B>. Результирующее регулярное выражение, которое описывает все приведенные условия выглядит вот так:  preg\_match\_all("/(?<!<TD>)(?<=>)\d\*\.\d\*(?!<\/B>)(?=<\/A>)/", $string, $matches); print\_r($matches);  После рассмотрения первого примера стоит сделать замечания и пояснения по поводу использования позиционных проверок.  1. Написанные друг за другом проверки применяются независимо друг от друга в одной точке, не меняя ее. Естественно, что совпадение будет найдено, если все проверки совпадут. В нашем примере это были точки перед и после цены. С точки зрения логики применения проверок нет никакой разницы, будет ли стоять проверка на тег <TD> перед проверкой на знак '>'. Правда, с точки зрения оптимизации первой позиционной проверкой должна идти та, которая имеет наибольшую вероятность несовпадения.  2. Совпавшие значения ретроспективных проверок не сохраняются. Т.е. если в нашем примере совпадает опережающая проверка, которая указывает, что после цены идет тег </A>, то сам тег </A>, который заключен в конструкцию (?=) не будет запоминаться в специальных перменных /1,/2 и т.д. Сделано это из-за того, что позиционная проверка совпадает не со строкой, а с местом в строке (она описывает место, где произошло совпадение, а не символы, которые совпали).  3. Нужно указать что PCRE не позволяет делать проверки на совпадение текста произвольной длинны. То есть нельзя делать, например, такую проверку: /(?<=\d+)  Механизм поиска совпадения в ретроспективной проверке реализован так, что при поиске механизму должна подаваться строка фиксированной длины, для того, чтобы в случае несовпадения, механизм мог вернуться назад на фиксированое количество символов и продолжить поиск совпадений в других позиционных проверках. Думаю, что сразу это понять сложно, но представьте себе как происходит поиск совпадения в части (?)(?<=>) вышеописанного регулярного выражения. Берется строка, в которой происходит поиск, отсчитывается от начала столько символов, сколько символов будет в совпадении позиционной проверки, в нашем варианте это 4: <, T, D, > с этого места происходит "заглядывание назад" (ретроспективные проверки на английском языке звучит как lookbhind assertions), т.е. все предыдущие 4 символа проверяются на совпадение со строкой <TD>, если механизм не нашел совпадения, то ему надо вернуться на 4 символа назад, выполнить тоже самое с проверкой (?<=>), т.е. отсчитать один символ, "заглянуть" назад, попробовать найти проверку предыдущего символа с символом '>'. Представьте себе, что условие совпадения состоит из строки нефиксированной длинны: (??) подобная запись должна означать, что перед ценой, не должен стоять тег <TD> в количестве максимимум один экземпляр (либо вообще не стоять). Вот и получается, что после того, как механизм отсчитает 4 символа от начала, он проверит на совпадение с <TD>, но в условии указано, что тега может и не быть вообще, тогда возникает вопрос, на сколько знаков верунться назад, чтобы проверить на совпадение другие проверки. На 4 или вообще не возвращаться? Сразу возникает вопрос, а зачем идти вперед, чтобы потом "заглянуть" назад? Делается это для того, чтобы в случае совпадения всех проверок сразу же начать проверку тех символов, которые идут после позиционных проверок.    **Выбрать все изображения со страницы**  Как-то мне нужно было получить все изображения, которые использовались на сайте. Что для этого надо сделать? Правильно, надо в браузере нажать на "Сохранить как", указать куда сохранить страницу. Появится файл с исходным кодом страницы и папка с изображениями. Но вы никогда не сохраните в эту папку изображения, которые прописаны в стилях объектов по крайней мере в эксплорере:  style="background-image:url(/editor/em/Unlink.gif);"  Для проведения вышеописанной операции надо: 1. попросить хозяина хоста использовать контент, размещенный на его сайте. 2. найти в тексте все строки, подобные приведенной выше, и выделить в них относительный путь к файлу 3. сформировать файл в котором будут выводиться изображения при помощи: <img src=полный\_путь\_к\_изображению> **Делаем:** В переменную $content получаем исходный код страницы. А дальше используя регулярные выражения ищем относительные пути, которые прописаны в стилях. Каждый раз, когда я описываю, как я реализовал пример, я сначала тщательно описываю, что ищем, и тщательно описываю, в каком контексте происходит поиск. Проанализировав исходный код страницы стало понятно, что кроме как в описании стилей относительные пути к изображениям нигде не используются. Слева от относительного пути идет последовательность символов: url( Справа от относительного пути стоит закрывающаяся круглая скобка. Между этими последовательностями символов могут быть буквы латинского алфавита, цифры и слеши, а также точка перед расширением файла.  Начнем с простого. Символы латинского алфавита, цифры, точка и слеш описываются символьным классом: [a-z.\/] их может быть сколько угодно, на самом деле больше 3 (имя файла, минимум один символ, точка, расширение, минимум один символ), но в данном случае, зная контекст, это некритично, поэтому указываем квантификатор \* [a-z.\/]\* Слева должны идти 'url(' и мы это описываем при помощи позитивной ретроспективной проверки: (?<=url() Но обратите внимание на то, что скобка в регулярных выражениях является спецсимволом группировки, поэтому чтобы она стала символом, надо перед ней поставить другой спецсимвол - слеш. (?<=url\() Справа от относительного пути должна стоять закрывающаяся круглая скобка. Это условие описывается при помощи позитивной опережающей проверки: (?=\)) Как видите, перед одной из скобок стоит слеш, что означает, что она интепретируется не как спецсимвол, а как литерал. Ниже приведен полный код на PHP, который выполняет все действия, кроме вопроса о разрешении использовать контент:  preg\_match\_all("/(?<=url\()[a-z.\/]\*(?=\))/i", $content, $matches); foreach($matches[0] as $item) { echo "<img src = http://htmlweb.ru".$item.">"; }    **Парсер всех внешних и внутренних ссылок со страницы**  В массиве $vnut только ссылки внутренние, в массиве $vnech только внешние ссылки.  $html=file\_get\_contents ('http://www.popsu.net'); $url='popsu.net'; $vnut=array(); $vnech=array(); preg\_match\_all('~<a [^<>]\*href=[\'"]([^\'"]+)[\'"][^<>]\*>(((?!~si',$html, $matches); foreach ($matches[1] as $val) { if (!preg\_match("~^[^=]+://~", $val) || preg\_match("~^[^://]+://(www\.)?".$url."~i", $val)) { $vnut[]=$val; } else $vnech[]=$val; } $vnut=array\_unique ($vnut); $vnech=array\_unique ($vnech); print\_r ($vnut); print\_r ($vnech);    **Является ли строка числом, длиной до 77 цифр:**  if (preg\_match("/^[0-9]{1,77}$/",$string)) echo "ДА";    **Состоит ли строка только из букв, цифр и "\_", длиной от 8 до 20 символов:**  if (preg\_match("/^[a-zа-я0-9\_]{8,20}$/",$string)) echo "yes"; else echo "no";    **Проверка строки на допустимость**  Есть ли в строке любые символы, кроме допустимых. Допустимыми считаются буквы, цифры и "\_". Длину тут проверять нельзя, разве что просто дополнительным условием strlen($string). Не путайте с предыдущим примером - хоть результат и одинаковый, но метод другой, "от противного"  if ( ! preg\_match("/[^a-zа-я0-9\_]/",$string)) echo "нет посторонних букв (OK)"; else echo "есть посторонние буквы (FALSE)";  Для регистро независимого сравнения используйте preg\_match с модификатором **i**().  **Проверка повторяющихся символов**  Есть ли в строке идущие подряд символы, не менее 3-х символов подряд (типа "абвгДДДеё", но не "ааббаабб"):  if (preg\_match("/(.)\\1\\1/",$string)) echo "yes"; else echo "no";    **Заменить везде в тексте СТРОКУ1 на СТРОКУ2**  (задача решается без регулярных выражений):  $string=str\_replace("СТРОКА1","СТРОКА2",$string);    **Заменить кривые коды перехода строки на нормальные:**  для этого нужно только удалить "\r". Переходы бывают нормальными (но разными!): "\n" или "\r\n". Еще бывают глюки, типа "\r\r\n".  $string=str\_replace("\r","",$string);    **Заменить все повторяющиеся пробелы на один**  Не пытайтесь здесь применить str\_replace, это хорошая функция, но не для данного примера.  $string=preg\_replace("/ХХ+/","Х",$string); // вместо Х поставьте пробел    **Удаление многократно повторяющихся знаков препинания**  Удаление знаков препинания, которые повторяются больше 3 раз, т.е. !!!!! -> !!!, ????? -> ??? и т.д. Заменяются следующие символы: . ! ? ( )  $text = preg\_replace('#(\.|\?|!|\(|\)){3,}#', '\1\1\1', $text);    **Сложная замена**  В тексте есть некоторые слова, допустим "СЛОВО" и "ЛЯЛЯЛЯ" (и т.д.), которые нужно одинаковым образом заменить на тоже самое, но с добавками. Возможно, слова отсутствуют или встречаются много раз в любом регистре.  Т.е. если было "слово" или "СлОвО" (или еще как), нужно заменить это на "<b>слово</b>" или "<b>СлОвО</b>" (смотря, как было). Другими словами нужно найти перечень слов в любом регистре и вставить по краям найленных слов фиксированные строки (на "<b>" и "</b>").  $string=preg\_replace("/(слово1|слово2|ляляля|слово99)/si","<b>\\1</b>",$string);    **Проверка URL на корректность**  Поддерживает все, что только может быть в УРЛ... Помните о том, что вы должны не только проверять, но и принимать новое значение от функции, т.к. та дописывает "http://" в случае его отсутствия.  // функция для удаления опасных сиволов function pregtrim($str) { return preg\_replace("/[^\x20-\xFF]/","",@strval($str)); }  // // проверяет URL и возвращает: // \* +1, если URL пуст // if (checkurl($url)==1) echo "пусто" // \* -1, если URL не пуст, но с ошибками // if (checkurl($url)==-1) echo "ошибка" // \* строку (новый URL), если URL найден и отпарсен // if (checkurl($url)==0) echo "все ок" // либо if (strlen(checkurl($url))>1) echo "все ок" // // Если протокола не было в URL, он будет добавлен ("http://") // function checkurl($url) { // режем левые символы и крайние пробелы $url=trim(pregtrim($url)); // если пусто - выход if (strlen($url)==0) return 1; //проверяем УРЛ на правильность if (!preg\_match("~^(?:(?:https?|ftp|telnet)://(?:[a-z0-9\_-]{1,32}". "(?::[a-z0-9\_-]{1,32})?@)?)?(?:(?:[a-z0-9-]{1,128}\.)+(?:com|net|". "org|mil|edu|arpa|gov|biz|info|aero|inc|name|[a-z]{2})|(?!0)(?:(?". "!0[^.]|255)[0-9]{1,3}\.){3}(?!0|255)[0-9]{1,3})(?:/[a-z0-9.,\_@%&". "?+=\~/-]\*)?(?:#[^ '\"&<>]\*)?$~i",$url,$ok)) return -1; // если не правильно - выход // если нет протокала - добавить if (!strstr($url,"://")) $url="http://".$url; // заменить протокол на нижний регистр: hTtP -> http $url=preg\_replace("~^[a-z]+~ie","strtolower('\\0')",$url); return $url; }  Таким образом для проверки нужно использовать нечто такое:  $url=checkurl($url); // перезаписали УРЛ в самого себя if ($url) exit("Ошибочный URL");  // Returns true if "abc" is found anywhere in $string.  preg\_match("/abc/", $string);  // Returns true if "abc" is found at the beginning of $string.  preg\_match("/^abc/", $string);  // Returns true if "abc" is found at the end of $string.  preg\_match("/abc$/", $string);  **Возвращает true если browser = Netscape 2, 3 or MSIE 3.**  preg\_match("/(ozilla.[23]|MSIE.3)/i", $\_SERVER["HTTP\_USER\_AGENT"]);  // Places three space separated words into $regs[1], $regs[2] and $regs[3].  preg\_match("/([[:alnum:]]+) ([[:alnum:]]+) ([[:alnum:]]+)/i", $string, $regs);  **Добавить <br /> в начало всех строк**  $string = preg\_replace("/^/", " ", $string);  **Добавить <br /> в конец всех строк**  $string = preg\_replace("/$/", " ", $string);  // Get rid of any newline characters in $string.  $string = preg\_replace("/\n/", "", $string);  **Удалить все аттрибуты у всех тегов, кроме a, p, img**  preg\_replace("/<([^ap(img)].\*?)\s.\*?>/is", "<\\1>", $string);  **Выбрать локальные URL**  Как можно выбрать не просто все урлы в HTML странице а те которые не начинаются на http://, другими словами локальные.  preg\_match\_all("#\s(?:href|src|url)=(?:[\"\'])?(.\*?)(?:[\"\'])?(?:[\s\>])#i", $buffer, $matches);  **Выбрать все параметры:**  $string = '<table border=\'0\' cellpadding = "0" cellspacing=0 style= "border-collapse: collapse">'; if (preg\_match\_all('#\s+([^=\s]+)\s\*=\s\*((?(?="|\') (?:"|\')([^"\']+)(?:"|\') | ([^\s]+)))#isx', $string, $matches)) { print\_r($matches); }  Array ( [0] => Array ( [0] => border='0' [1] => cellpadding = "0" [2] => cellspacing=0 [3] => style= "border-collapse: collapse" ) [1] => Array ( [0] => border [1] => cellpadding [2] => cellspacing [3] => style ) [2] => Array ( [0] => '0' [1] => "0" [2] => 0 [3] => "border-collapse: collapse" ) [3] => Array ( [0] => 0 [1] => 0 [2] => [3] => border-collapse: collapse ) [4] => Array ( [0] => [1] => [2] => 0 [3] => ) )  **Конвертор HTML в текст**  // $document на выходе должен содержать HTML-документ. // Необходимо удалить все HTML-теги, секции javascript, // пробельные символы. Также необходимо заменить некоторые // HTML-сущности на их эквивалент.  $search = array ("'<script[^>]\*?>.\*?</script>'si", // Вырезает javaScript "'<[\/\!]\*?[^<>]\*?>'si", // Вырезает HTML-теги "'([\r\n])[\s]+'", // Вырезает пробельные символы "'&(quot|#34);'i", // Заменяет HTML-сущности "'&(amp|#38);'i", "'&(lt|#60);'i", "'&(gt|#62);'i", "'&(nbsp|#160);'i", "'&(iexcl|#161);'i", "'&(cent|#162);'i", "'&(pound|#163);'i", "'&(copy|#169);'i", "'&#(\d+);'e"); // интерпретировать как php-код  $replace = array ("", "", "\\1", "\"", "&", "<", ">", " ", chr(161), chr(162), chr(163), chr(169), "chr(\\1)");  $text = preg\_replace($search, $replace, $document);    **Найти и заменить все "http://" на ссылки**    Вариант 1:  $text = preg\_replace('#(?<!\])\bhttp://[^\s\[<]+#i', "<a href=\"$0\" target=\_blank><u>Посмотреть на сайте</u></a>", nl2br(stripslashes($text)));    Вариант 2, с выделением домена:  // Cuts off long URLs at $url\_length, and appends "..." function reduceurl($url, $url\_length) { $reduced\_url = substr($url, 0, $url\_length); if (strlen($url) > $url\_length) $reduced\_url .= '...'; return $reduced\_url; }  $linktext = preg\_replace("#\[(([a-zA-Z]+://)([a-zA-Z0-9?&%.;:/=+\_-]\*))\]#e", "'<a href=\"$1\" target=\"\_blank\">' . reduceurl(\"$3\", 30) . '</a>'", $linktext);    Еще один вариант, учитывающий "WWW."  // match protocol://address/path/file.extension?some=variable&another=asf% $text = preg\_replace("/\s(([a-zA-Z]+:\/\/)([a-z][a-z0-9\_\..-]\*[a-z]{2,6})([a-zA-Z0-9\/\*-?&%]\*))\s/i", " <a href=\"$1\">$3</a> ", $text);  // match www.something.domain/path/file.extension?some=variable&another=asf% $text = preg\_replace("/\s(www\.([a-z][a-z0-9\_\..-]\*[a-z]{2,6})([a-zA-Z0-9\/\*-?&%]\*))\s/i", " <a href=\"http://$1\">$2</a> ", $text);    **Разбор адесов E-mail.**  $text = "Адреса: [user-first@mail.ru](mailto:user-first@mail.ru), [second.user@mail.ru](mailto:second.user@mail.ru)."; $html = preg\_replace( '{ [\w-.]+ # имя ящика @ [\w-]+(\.[\w-]+)\* # имя хоста }xs', '<a href="mailto:$0">$0</a>', $text ); echo $html;  То же самое, но немножко по-другому:  $html = preg\_replace( '/(\S+)@([a-z0-9.-]+)/is', '<a href="mailto:$0">$0</a>', $text);    **Проверить, что в строке есть число (одна или более цифра)**  preg\_match('/(\d+)/s', "article\_123.html", $pockets); // Совпадение (подвыражение в скобках) окажется в $pockets[1]. echo $pockets[1]; // выводит 123  **Найти в тексте адрес E-mail**  // \S означает "не пробел", а [a-z0-9.]+ - // "любое число букв, цифр или точек". Модификатор 'i' после '/' // заставляет PHP не учитывать регистр букв при поиске совпадений. // Модификатор 's', стоящий рядом с 'i', говорит, что мы работаем // в "однострочном режиме" (см. ниже в этой главе). preg\_match('/(\S+)@([a-z0-9.]+)/is', "Привет от [somebody@mail.ru](mailto:somebody@mail.ru)!", $p); // Имя хоста будет в $p[2], а имя ящика (до @) - в $p[1]. echo "В тексте найдено: ящик - $p[1], хост - $p[2]";    **Превращение E-mail в HTML-ссылку.**  $text = "Привет от [somebody@mail.ru](mailto:somebody@mail.ru), а также от [other@mail.ru](mailto:other@mail.ru)!"; $html = preg\_replace( '/(\S+)@([a-z0-9.]+)/is', // найти все E-mail '<a href="mailto:$0">$0</a>', // заменить их по шаблону $text // искать в $text ); echo $html;    **Простейший разбор даты.**  $str = " 15-16/2000 "; // к примеру $re = '{ ^\s\*( # начало строки (\d+) # день \s\* [[:punct:]] \s\* # разделитель (\d+) # месяц \s\* [[:punct:]] \s\* # разделитель (\d+) # год )\s\*$ # конец строки }xs'; // Разбиваем строку на куски при помощи preg\_match(). preg\_match($re, $str, $pockets) or die("Not a date: $str"); // Теперь разбираемся с карманами. echo "Дата без пробелов: '$pockets[1]' <br>"; echo "День: $pockets[2] <br>"; echo "Месяц: $pockets[3] <br>"; echo "Год: $pockets[4] <br>";    **Замена по шаблону**  $text = htmlspecialchars(file\_get\_contents(\_\_FILE\_\_)); $html = preg\_replace('/(\$[a-z]\w\*)/is', '<b>$1</b>', $text); echo "<pre>$html</pre>";    **Обратные ссылки**  $str = "Hello, this <b>word</b> is bold!"; $re = '|<(\w+) [^>]\* > (.\*?) </\1>|xs'; preg\_match($re, $str, $pockets) or die("Нет тэгов."); echo htmlspecialchars("'$pockets[2]' обрамлено тэгом '$pockets[1]'");    **"Жадные" квантификаторы**  $str = "Hello, this <b>word</b> is <b>bold</b>!"; $re = '|<(\w+) [^>]\* > (.\*) </\1>|xs'; preg\_match($re, $str, $pockets) or die("Нет тэгов."); echo htmlspecialchars("'$pockets[2]' обрамлено тэгом '$pockets[1]'");    **Сравнение "жадных" и "ленивых" квантификаторов**  $str = '[b]жирный текст [b]а тут - еще жирнее[/b] вернулись[/b]'; $to = '<b>$1</b>'; $re1 = '|\[b\] (.\*) \[/b\]|ixs'; $re2 = '|\[b\] (.\*?) \[/b\]|ixs'; $result = preg\_replace($re1, $to, $str); echo "Жадная версия: ".htmlspecialchars($result)."<br />"; $result = preg\_replace($re2, $to, $str); echo "Ленивая версия: ".htmlspecialchars($result)."<br />";    **Многострочность.**  $str = file\_get\_contents(\_\_FILE\_\_); $str = preg\_replace('/^/m', "\t", $str); echo "<pre>".htmlspecialchars($str)."</pre>";    **Использование PREG\_OFFSET\_CAPTURE**  $st = '<b>жирный текст</b>'; $re = '|<(\w+).\*?>(.\*?)</\1>|s'; preg\_match($re, $st, $p, PREG\_OFFSET\_CAPTURE); echo "<pre>"; print\_r($p); echo "</pre>";    **Применение preg\_grep()**  foreach (preg\_grep('/^ex\d/s', glob("\*")) as $fn) echo "Файл примера: $fn<br />";    **Различные флаги preg\_match\_all()**  Header("Content-type: text/plain"); $flags = array( "PREG\_PATTERN\_ORDER", "PREG\_SET\_ORDER", "PREG\_SET\_ORDER|PREG\_OFFSET\_CAPTURE", ); $re = '|<(\w+).\*?>(.\*?)</\1>|s'; $text = "<b>текст</b> и еще <i>другой текст</i>"; echo "Строка: $text\n"; echo "Выражение: $re\n\n"; foreach ($flags as $name) { preg\_match\_all($re, $text, $pockets, eval("return $name;")); echo "Флаг $name:\n"; print\_r($pockets); echo "\n"; }    **preg\_replace\_callback()**  // Пользовательская функция. Будет вызываться для каждого // совпадения с регулярным выражением. function toUpper($pockets) { return $pockets[1].strtoupper($pockets[2]).$pockets[3]; } $str = '<hTmL><bOdY bgcolor="white">Three captains, one ship.</bOdY></html>'; $str = preg\_replace\_callback('{(</?)(\w+)(.\*?>)}s', "toUpper", $str); echo htmlspecialchars($str);  **Получение строки GET-запроса.**  Для начала поставим самую простую задачу - получить часть URL, содержащую GET-параметры.  function ggp($url) { // get GET-parameters string preg\_match('/^(.+?)(\?.\*?)?(#.\*)?$/', $url, $matches); $gp = (isset($matches[2])) ? $matches[2] : ''; return $gp; }  Не стоит забывать, что адрес может вовсе не содержать никакого GET-запроса, и массив вхождений может не иметь второго элемента 3.  **Исключение GET-запроса из URL.**  Иногда нужно получить URL без GET-параметров (например, при перенаправлении запросов с помощью mod\_rewrite зачастую требуется проводить анализ URL, чтобы сформировать ответ клиенту; нередко для анализа нужна только статическая часть URL, а часть, где передается GET-запрос, не нужна и даже мешает).  // удаление GET-параметров из URL $str = preg\_replace('/^(.+?)(\?.\*?)?(#.\*)?$/', '$1$3', $url);  **заменить все символы кроме чисел и запятой на ''**  $value = preg\_replace('/[^\d,]+/', '', $value); // заменить все символы кроме чисел и запятой на ''    **Есть ли в строке параметров сессия (PHPSESSID):**  print $\_SERVER['REQUEST\_URI'].'<br>'; if (preg\_match("/=([a-f0-9]{32})&/i", $\_SERVER['REQUEST\_URI'].'&')){ print 'Да'; // удалить сессию из строки параметров //print str\_replace('&&','&',str\_replace('?&','?',preg\_replace("/&\*sid=([a-f0-9]{32})&\*/i", '&', $\_SERVER['REQUEST\_URI']))); } else print 'Нет';  **Удалить из строки параметр page и добавить другой page.**  Этот пример позволяет заменить один параметр на другой не испортив все остальные параметры строки. Если Вы найдете более оптимальное решение, присылайте.  $href1=str\_replace('&&','&',str\_replace('?&','?',preg\_replace("/&\*page=([0-9]{1,3})&\*/i", '&', $\_SERVER['REQUEST\_URI']))); $href1=str\_replace('?&','?',$href1.(strpos($href1, '?')===false?'?':'&').'page=');  **Проверка формата времени. date: mm:hh.**  $time = "10:11"; if (!preg\_match('/^([0-1][0-9]|[2][0-3]):([0-5][0-9])$/', $time)) echo "Время введено неправильно";  **Как вытащить слова из текста?**  Это регулярное выражение PHP разбирает текст на отдельные слова, основываясь на определении: слово - это непрерывная последовательность букв английского или русского алфавитов.  $x="Типа, %^& читайте\_\_\_люди~~~~\_\_маны\_\_ На... РУССКОМ!! Будете+здоровы. abc, qwe, zxc"; preg\_match\_all('/([a-zA-Zа-яА-Я]+)/',$x,$ok); for ($i=0; $i<count($ok[1]); $i++) echo $ok[1][$i]."<br>";  Результат будет таким:  Типа читайте люди маны На РУССКОМ Будете здоровы abc qwe zxc  **Как заставить работать с русскими буквами в UTF-8?**  В PHP это решается вот так:  preg\_replace("/[^\p{L}0-9\+\-\\_:\.@ ]/u", "", $\_string));  \p{L} = все буквы /u = работать с UTF-8 |