- PHP Manual
- Function Reference
- Text Processing
- PCRE
- PCRE Functions

Change language: English

Submit a Pull Request Report a Bug

preg_match

(PHP 4, PHP 5, PHP 7, PHP 8)

preg_match — Perform a regular expression match

Description ¶

```
preg_match(
    string $pattern,
    string $subject,
    array &$matches = null,
    int $flags = 0,
    int $offset = 0
): int|false
```

Searches `subject` for a match to the regular expression given in `pattern`.

Parameters ¶

`pattern`

The pattern to search for, as a string.

`subject`

The input string.

`matches`

If `matches` is provided, then it is filled with the results of search. *$matches[0]* will contain the text that matched the full pattern, *$matches[1]* will have the text that matched the first captured parenthesized subpattern, and so on.

`flags`

`flags` can be a combination of the following flags:

`PREG_OFFSET_CAPTURE`

If this flag is passed, for every occurring match the appendant string offset (in bytes) will also be returned. Note that this changes the value of `matches` into an array where every element is an array consisting of the matched string at offset `0` and its string offset into `subject` at offset `1`.

```
<?php
preg_match('/(foo)(bar)(baz)/', 'foobarbaz', $matches, PREG_OFFSET_CAPTURE);
print_r($matches);
?>
```

The above example will output:

```
Array
(
    [0] => Array
        (
            [0] => foobarbaz
            [1] => 0
        )

    [1] => Array
        (
            [0] => foo
            [1] => 0
        )

    [2] => Array
        (
            [0] => bar
            [1] => 3
        )

    [3] => Array
        (
            [0] => baz
            [1] => 6
        )
)
```

PREG_UNMATCHED_AS_NULL

If this flag is passed, unmatched subpatterns are reported as `null`; otherwise they are reported as an empty string.

```php
<?php
preg_match('/(a)(b)*(c)/', 'ac', $matches);
var_dump($matches);
preg_match('/(a)(b)*(c)/', 'ac', $matches, PREG_UNMATCHED_AS_NULL);
var_dump($matches);
?>
```

The above example will output:

```
array(4) {
  [0]=>
  string(2) "ac"
  [1]=>
  string(1) "a"
  [2]=>
  string(0) ""
  [3]=>
  string(1) "c"
}
array(4) {
  [0]=>
  string(2) "ac"
  [1]=>
  string(1) "a"
  [2]=>
  NULL
  [3]=>
  string(1) "c"
}
```

`offset`

Normally, the search starts from the beginning of the subject string. The optional parameter `offset` can be used to specify the alternate place from which to start the search (in bytes).

Note:

Using `offset` is not equivalent to passing `substr($subject, $offset)` to **preg_match()** in place of the subject string, because `pattern` can contain assertions such as *^*, *$* or *(?<=x)*. Compare:

```
<?php
$subject = "abcdef";
$pattern = '/^def/';
preg_match($pattern, $subject, $matches, PREG_OFFSET_CAPTURE, 3);
print_r($matches);
?>
```

The above example will output:

```
Array
(
)
```

while this example

```
<?php
$subject = "abcdef";
$pattern = '/^def/';
preg_match($pattern, substr($subject,3), $matches, PREG_OFFSET_CAPTURE);
print_r($matches);
?>
```

will produce

```
Array
(
    [0] => Array
        (
            [0] => def
            [1] => 0
        )

)
```

Alternatively, to avoid using substr(), use the `\G` assertion rather than the `^` anchor, or the `A` modifier instead, both of which work with the `offset` parameter.

Return Values ¶

preg_match() returns 1 if the `pattern` matches given `subject`, 0 if it does not, or **`false`** on failure.

Warning

This function may return Boolean **false**, but may also return a non-Boolean value which evaluates to **false**. Please read the section on Booleans for more information. Use the === operator for testing the return value of this function.

Errors/Exceptions ¶

If the regex pattern passed does not compile to a valid regex, an **E_WARNING** is emitted.

Changelog ¶

Version	Description
7.2.0	The **PREG_UNMATCHED_AS_NULL** is now supported for the `$flags` parameter.

Examples ¶

Example #1 Find the string of text "php"

```
<?php
// The "i" after the pattern delimiter indicates a case-insensitive search
if (preg_match("/php/i", "PHP is the web scripting language of choice.")) {
echo "A match was found.";
} else {
echo "A match was not found.";
}
?>
```

Example #2 Find the word "web"

```
<?php
/* The \b in the pattern indicates a word boundary, so only the distinct
* word "web" is matched, and not a word partial like "webbing" or "cobweb" */
if (preg_match("/\bweb\b/i", "PHP is the web scripting language of choice.")) {
echo "A match was found.";
} else {
echo "A match was not found.";
}

if (preg_match("/\bweb\b/i", "PHP is the website scripting language of choice.")) {
echo "A match was found.";
} else {
echo "A match was not found.";
}
?>
```

Example #3 Getting the domain name out of a URL

```
<?php
// get host name from URL
preg_match('@^(?:http://)?([^/]+)@i',
"http://www.php.net/index.html", $matches);
$host = $matches[1];

// get last two segments of host name
preg_match('/[^.]+\.[^.]+$/', $host, $matches);
echo "domain name is: {$matches[0]}\n";
?>
```

The above example will output:

```
domain name is: php.net
```

Example #4 Using named subpattern

```
<?php

$str = 'foobar: 2008';

preg_match('/(?P<name>\w+): (?P<digit>\d+)/', $str, $matches);

/* Alternative */
// preg_match('/(?<name>\w+): (?<digit>\d+)/', $str, $matches);

print_r($matches);

?>
```

The above example will output:

```
Array
(
    [0] => foobar: 2008
    [name] => foobar
    [1] => foobar
    [digit] => 2008
    [2] => 2008
)
```

Notes ¶

Tip

Do not use **preg_match()** if you only want to check if one string is contained in another string. Use strpos() instead as it will be faster.

See Also ¶

- PCRE Patterns
- preg_quote() - Quote regular expression characters
- preg_match_all() - Perform a global regular expression match
- preg_replace() - Perform a regular expression search and replace
- preg_split() - Split string by a regular expression
- preg_last_error() - Returns the error code of the last PCRE regex execution
- preg_last_error_msg() - Returns the error message of the last PCRE regex execution

+ add a note

User Contributed Notes 51 notes

up
down
918
force at md-t dot org ¶
11 years ago

```
Simple regex

Regex quick reference
[abc]     A single character: a, b or c
[^abc]     Any single character but a, b, or c
[a-z]     Any single character in the range a-z
[a-zA-Z]     Any single character in the range a-z or A-Z
^     Start of line
```

```
$     End of line
\A     Start of string
\z     End of string
.     Any single character
\s     Any whitespace character
\S     Any non-whitespace character
\d     Any digit
\D     Any non-digit
\w     Any word character (letter, number, underscore)
\W     Any non-word character
\b     Any word boundary character
(...)     Capture everything enclosed
(a|b)     a or b
a?     Zero or one of a
a*     Zero or more of a
a+     One or more of a
a{3}     Exactly 3 of a
a{3,}     3 or more of a
a{3,6}     Between 3 and 6 of a

options: i case insensitive m make dot match newlines x ignore whitespace in regex o perform #{...} substitutions only once
```

up
down
106
MrBull ¶
11 years ago

Sometimes its useful to negate a string. The first method which comes to mind to do this is: [^(string)] but this of course won't work. There is a solution, but it is not very well known. This is the simple piece of code on how a negation of a string is done:

(?:(?!string).)

?: makes a subpattern (see http://www.php.net/manual/en/regexp.reference.subpatterns.php) and ?! is a negative look ahead. You put the negative look ahead in front of the dot because you want the regex engine to first check if there is an occurrence of the string you are negating. Only if it is not there, you want to match an arbitrary character.

Hope this helps some ppl.

up
down
47

[ruakuu at NOSPAM dot com ¶](#)

13 years ago

```
Was working on a site that needed japanese and alphabetic letters and needed to
validate input using preg_match, I tried using \p{script} but didn't work:

<?php
$pattern ='/^([-a-zA-Z0-9_\p{Katakana}\p{Hiragana}\p{Han}]*)$/u'; // Didn't work
?>

So I tried with ranges and it worked:

<?php
$pattern ='/^[-a-zA-Z0-9_\x{30A0}-\x{30FF}'
         .'\x{3040}-\x{309F}\x{4E00}-\x{9FBF}\s]*$/u';
$match_string = '印刷最安 ニキビ跡除去 ゲームボーイ';

if (preg_match($pattern, $match_string)) {
    echo "Found - pattern $pattern";
} else {
    echo "Not found - pattern $pattern";
}
?>

U+4E00–U+9FBF Kanji
U+3040–U+309F Hiragana
U+30A0–U+30FF Katakana

Hope its useful, it took me several hours to figure it out.
```

[up](#)
[down](#)
32

[yofilter-php at yahoo dot co dot uk ¶](#)

9 years ago

```
There does not seem to be any mention of the PHP version of switches that can be used with regular expressions.

preg_match_all('/regular expr/sim',$text).

The s i m being the location for and available switches (I know about)
The i is to ignore letter cases (this is commonly known - I think)
The s tells the code NOT TO stop searching when it encounters \n (line break) - this is important with multi-line entries for example text from
an editor that needs search.
The m tells the code it is a multi-line entry, but importantly allows the use of ^ and $ to work when showing start and end.

I am hoping this will save someone from the 4 hours of torture that I endured, trying to workout this issue.
```

up
down
69
cebelab at gmail dot com ¶
12 years ago

```
I noticed that in order to deal with UTF-8 texts, without having to recompile php with the PCRE UTF-8 flag enabled, you can just add the
following sequence at the start of your pattern: (*UTF8)

for instance : '#(*UTF8)[[:alnum:]]#' will return TRUE for 'é' where '#[[:alnum:]]#' will return FALSE

found this very very useful tip after hours of research over the web directly in pcre website right here : http://www.pcre.org/pcre.txt
there are many further informations about UTF-8 support in the lib

hop that will help!

--
cedric
```

up
down
16
luc _ santeramo at t yahoo dot com ¶
13 years ago

```
If you want to validate an email in one line, use filter_var() function !
http://fr.php.net/manual/en/function.filter-var.php

easy use, as described in the document example :
var_dump(filter_var('bob@example.com', FILTER_VALIDATE_EMAIL));
```

up
down
37
arash dot hemmat at gmail dot com ¶
11 years ago

```
For those who search for a unicode regular expression example using preg_match here it is:
```

```
Check for Persian digits
preg_match( "/[^\x{06F0}-\x{06F9}\x]+/u" , '۱۲۳۴۵۶۷۸۹۰' );
```

up

down

14

ulli dot luftpumpe at murkymind dot de ¶

10 years ago

```
Matching a backslash character can be confusing, because double escaping is needed in the pattern: first for PHP, second for the regex engine
<?php
//match newline control character:
preg_match('/\n/','\n');   //pattern matches and is stored as control character 0x0A in the pattern string
preg_match('/\\\n/','\n'); //very same match, but is stored escaped as 0x5C,0x6E in the pattern string

//trying to match "\'" (2 characters) in a text file, '\\\'' as PHP string:
$subject = file_get_contents('myfile.txt');
preg_match('/\\\'/',$subject);    //DOESN'T MATCH!!! stored as 0x5C,0x27 (escaped apostrophe), this only matches apostrophe
preg_match('/\\\\\'/',$subject);  //matches, stored as 0x5C,0x5C,0x27 (escaped backslash and unescaped apostrophe)
preg_match('/\\\\\\\/',$subject); //also matches, stored as 0x5C,0x5C,0x5C,0x27 (escaped backslash and escaped apostrophe)

//matching "\n" (2 characters):
preg_match('/\\\\n/','\\n');
preg_match('/\\\n/','\\n'); //same match - 3 backslashes are interpreted as 2 in PHP, if the following character is not escapeable
?>
```

up

down

5

cmallabon at homesfactory dot com ¶

11 years ago

```
Just an interesting note. Was just updating code to replace ereg() with strpos() and preg_match and the thought occured that preg_match() could
be optimized to quit early when only searching if a string begins with something, for example
<?php
if(preg_match("/^http/", $url))
{
//do something
}
?>

vs
```

```
<?php
if(strpos($url, "http") === 0)
{
//do something
}
?>
```

As I guessed, strpos() is always faster (about 2x) for short strings like a URL but for very long strings of several paragraphs (e.g. a block of XML) when the string doesn't start with the needle preg_match as twice as fast as strpos() as it doesn't scan the entire string.

So, if you are searching long strings and expect it to normally be true (e.g. validating XML), strpos() is a much faster BUT if you expect if to often fail, preg_match is the better choice.

up
down
42

mohammad40g at gmail dot com ¶

11 years ago

This sample is for checking persian character:

```
<?php
   preg_match("/[\x{0600}-\x{06FF}\x]{1,32}/u", 'محمد');
?>
```

up
down
9

andre at koethur dot de ¶

9 years ago

Be aware of bug https://bugs.php.net/bug.php?id=50887 when using sub patterns: Un-matched optional sub patterns at the end won't show up in $matches.

Here is a workaround: Assign a name to all subpatterns you are interested in, and merge $match afterwards with an constant array containing some reasonable default values:

```
<?php
if (preg_match('/^(?P<lang>[^;*][^;]*){1}(?:;q=(?P<qval>[0-9.]+))?$/u', 'de', $match))
{
  $match = array_merge(array('lang' => '', 'qval' => ''), $match);
  print_r($match);
}
```

```
?>

This outputs:
Array
(
    [lang] => de
    [qval] =>
    [0] => de
    [1] => de
)

Instead of:
Array
(
    [0] => de
    [lang] => de
    [1] => de
)
```

up
down
21
daevid at daevid dot com ¶
13 years ago

```
I just learned about named groups from a Python friend today and was curious if PHP supported them, guess what -- it does!!!

http://www.regular-expressions.info/named.html

<?php
   preg_match("/(?P<foo>abc)(.*)(?P<bar>xyz)/",
                     'abcdefghijklmnopqrstuvwxyz',
                     $matches);
   print_r($matches);
?>

will produce:

Array
(
    [0] => abcdefghijklmnopqrstuvwxyz
    [foo] => abc
    [1] => abc
    [2] => defghijklmnopqrstuvw
    [bar] => xyz
    [3] => xyz
)

Note that you actually get the named group as well as the numerical key
value too, so if you do use them, and you're counting array elements, be
aware that your array might be bigger than you initially expect it to be.
```

up
down
13
solixmexico at outlook dot com ¶
6 years ago

```
To validate directorys on Windows i used this:

if( preg_match("#^([a-z]{1}\:{1})?[\\\/]?([\-\w]+[\\\/]?)*$#i",$_GET['path'],$matches) !== 1 ){
    echo("Invalid value");
}else{
    echo("Valid value");
}

The parts are:

#^ and $i             Make the string matches at all the pattern, from start to end for ensure a complete match.
([a-z]{1}\:{1})?        The string may starts with one letter and a colon, but only 1 character for eachone, this is for the drive letter (C:)
[\\\/]?              The string may contain, but not require 1 slash or backslash after the drive letter, (\/)
([\-\w]+[\\\/]?)*     The string must have 1 or more of any character like hyphen, letter, number, underscore, and may contain a slash or back
slash at the end, to have a directory like ("/" or "folderName" or "folderName/"), this may be repeated one or more times.
```

up
down
6
asdfasdasad34535 at iflow dot at ¶
9 years ago

```
Attention! PREG_OFFSET_CAPTURE not UTF-8 aware when using u modifier
and it's not a but, it's a feature:
https://bugs.php.net/bug.php?id=37391
```

```
Possible workaround: Use mb_strpos to get the correct offset, instead of the flag.

UTF-8 support would be nice.
```

up
down
31
jonathan dot lydall at gmail dot removethispart dot com ¶
14 years ago

```
Because making a truly correct email validation function is harder than one may think, consider using this one which comes with PHP through the
filter_var function (http://www.php.net/manual/en/function.filter-var.php):

<?php
$email = "someone@domain .local";

if(!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    echo "E-mail is not valid";
} else {
    echo "E-mail is valid";
}
?>
```

up
down
4
geompse at gmail dot com ¶
5 years ago

```
The function will return false and raise a warning if the input $subject is too long :
[PhpWarning] preg_match(): Subject is too long

I believe the limit is 1 or 2 GB because I was using a 2.2GB string.
While a parameter might exist to alter this limit, in my case it was possible and wiser to use <500MB strings instead.
```

up
down
3
workhorse at op dot pl ¶
11 years ago

```
Preg_match returns empty result trying to validate $subject with carriege returns (/n/r).
To solve it one need to use /s modifier in $pattern string.
<?php
$pattern='/.*/s';
```

```
$valid=preg_match($pattern, $subject, $match);
?>
```

up
down
5
aer0s ¶
10 years ago

```
Simple function to return a sub-string following the preg convention. Kind of expensive, and some might say lazy but it has saved me time.

# preg_substr($pattern,$subject,[$offset]) function
# @author   aer0s
#  return a specific sub-string in a string using
#   a regular expression
# @param   $pattern   regular expression pattern to match
# @param   $subject   string to search
# @param   [$offset]   zero based match occurrence to return
#
# [$offset] is 0 by default which returns the first occurrence,
# if [$offset] is -1 it will return the last occurrence

function preg_substr($pattern,$subject,$offset=0){
    preg_match_all($pattern,$subject,$matches,PREG_PATTERN_ORDER);
    return $offset==-1?array_pop($matches[0]):$matches[0][$offset];
}

example:

             $pattern = "/model(\s|-)[a-z0-9]/i";
             $subject = "Is there something wrong with model 654, Model 732, and model 43xl or is Model aj45B the preferred choice?";

             echo preg_substr($pattern,$subject);
             echo preg_substr($pattern,$subject,1);
             echo preg_substr($pattern,$subject,-1);

Returns something like:

             model 654
             Model 732
             Model aj45B
```

up
down
12
Kae Cyphet ¶
12 years ago

```
for those coming over from ereg, preg_match can be quite intimidating. to get started here is a migration tip.

<?php
if(ereg('[^0-9A-Za-z]',$test_string)) // will be true if characters arnt 0-9, A-Z or a-z.

if(preg_match('/[^0-9A-Za-z]/',$test_string)) // this is the preg_match version. the /'s are now required.
?>
```

up
down
4
Anonymous ¶
10 years ago

```
Here is a function that decreases the numbers inside a string (useful to convert DOM object into simplexml object)

e.g.: decremente_chaine("somenode->anode[2]->achildnode[3]") will return "somenode->anode[1]->achildnode[2]"

the numbering of the nodes in simplexml starts from zero, but from 1 in DOM xpath objects

<?php
function decremente_chaine($chaine)
    {
        //récupérer toutes les occurrences de nombres et leurs indices
        preg_match_all("/[0-9]+/",$chaine,$out,PREG_OFFSET_CAPTURE);
            //parcourir les occurrences
            for($i=0;$i<sizeof($out[0]);$i++)
            {
                $longueurnombre = strlen((string)$out[0][$i][0]);
                $taillechaine = strlen($chaine);
                // découper la chaine en 3 morceaux
                $debut = substr($chaine,0,$out[0][$i][1]);
                $milieu = ($out[0][$i][0])-1;
                $fin = substr($chaine,$out[0][$i][1]+$longueurnombre,$taillechaine);
                 // si c'est 10,100,1000 etc. on décale tout de 1 car le résultat comporte un chiffre de moins
                 if(preg_match('#[1][0]+$#', $out[0][$i][0]))
                {
                    for($j = $i+1;$j<sizeof($out[0]);$j++)
                    {
                        $out[0][$j][1] = $out[0][$j][1] -1;
                    }
                }
               $chaine = $debut.$milieu.$fin;
            }
        return $chaine;
    }
?>
```

up
down
7
sainnr at gmail dot com ¶
12 years ago

```
This sample regexp may be useful if you are working with DB field types.

(?P<type>\w+)($|\((?P<length>(\d+|(.*)))\))

For example, if you are have a such type as "varchar(255)" or "text", the next fragment

<?php
   $type = 'varchar(255)';  // type of field
   preg_match('/(?P<type>\w+)($|\((?P<length>(\d+|(.*)))\))/', $type, $field);
   print_r($field);
?>

will output something like this:
Array ( [0] => varchar(255) [type] => varchar [1] => varchar [2] => (255) [length] => 255 [3] => 255 [4] => 255 )
```

up
down
7
ian_channing at hotmail dot com ¶
12 years ago

```
When trying to check a file path that could be windows or unix it took me quite a few tries to get the escape characters right.

The Unix directory separator must be escaped once and the windows directory separator must be escaped twice.
```

```
This will match path/to/file and path\to\file.exe

preg_match('/^[a-z0-9_.\/\\\]*$/i', $file_string);
```

up
down
16
splattermania at freenet dot de ¶
13 years ago

```
As I wasted lots of time finding a REAL regex for URLs and resulted in building it on my own, I now have found one, that seems to work for all
kinds of urls:

<?php
    $regex = "((https?|ftp)\:\/\/)?"; // SCHEME
    $regex .= "([a-z0-9+!*(),;?&=\$_.-]+(\:[a-z0-9+!*(),;?&=\$_.-]+)?@)?"; // User and Pass
    $regex .= "([a-z0-9-.]*)\.([a-z]{2,3})"; // Host or IP
    $regex .= "(\:[0-9]{2,5})?"; // Port
    $regex .= "(\/([a-z0-9+\$_-]\.?)+)*\/?"; // Path
    $regex .= "(\?[a-z+&\$_.-][a-z0-9;:@&%=+\/\$_.-]*)?"; // GET Query
    $regex .= "(#[a-z_.-][a-z0-9+\$_.-]*)?"; // Anchor
?>

Then, the correct way to check against the regex ist as follows:

<?php
       if(preg_match("/^$regex$/", $url))
       {
               return true;
       }
?>
```

up
down
14
corey [works at] effim [delete] .com ¶
13 years ago

```
I see a lot of people trying to put together phone regex's and struggling (hey, no worries...they're complicated). Here's one that we use that's
pretty nifty. It's not perfect, but it should work for most non-idealists.

*** Note: Only matches U.S. phone numbers. ***
```

```
<?php

// all on one line...
$regex = '/^(?:1(?:[. -])?)?(?:\((?=\d{3}\)))?([2-9]\d{2})(?:(?<=\(\d{3})\))? ?(?:(?<=\d{3})[.-])?([2-9]\d{2})[. -]?(\d{4})(?: (?i:ext)\.? ?
(\d{1,5}))?$/';

// or broken up
$regex = '/^(?:1(?:[. -])?)?(?:\((?=\d{3}\)))?([2-9]\d{2})'
        .'(?:(?<=\(\d{3})\))? ?(?:(?<=\d{3})[.-])?([2-9]\d{2})'
        .'[. -]?(\d{4})(?: (?i:ext)\.? ?(\d{1,5}))?$/';

?>

If you're wondering why all the non-capturing subpatterns (which look like this "(?:", it's so that we can do this:

<?php

$formatted = preg_replace($regex, '($1) $2-$3 ext. $4', $phoneNumber);

// or, provided you use the $matches argument in preg_match

$formatted = "($matches[1]) $matches[2]-$matches[3]";
if ($matches[4]) $formatted .= " $matches[4]";

?>

*** Results: ***
520-555-5542 :: MATCH
520.555.5542 :: MATCH
5205555542 :: MATCH
520 555 5542 :: MATCH
520) 555-5542 :: FAIL
(520 555-5542 :: FAIL
(520)555-5542 :: MATCH
(520) 555-5542 :: MATCH
(520) 555 5542 :: MATCH
520-555.5542 :: MATCH
520 555-0555 :: MATCH
(520)5555542 :: MATCH
```

```
520.555-4523 :: MATCH
19991114444 :: FAIL
19995554444 :: MATCH
514 555 1231 :: MATCH
1 555 555 5555 :: MATCH
1.555.555.5555 :: MATCH
1-555-555-5555 :: MATCH
520-555-5542 ext.123 :: MATCH
520.555.5542 EXT 123 :: MATCH
5205555542 Ext. 7712 :: MATCH
520 555 5542 ext 5 :: MATCH
520) 555-5542 :: FAIL
(520 555-5542 :: FAIL
(520)555-5542 ext .4 :: FAIL
(512) 555-1234 ext. 123 :: MATCH
1(555)555-5555 :: MATCH
```

up
down
2
danielrydell at gmail dot com ¶
5 years ago

When trying to match accented characters, such as those found in Spanish, there seems to be a different internal interpretation when using character classes. So the best way is to add the u option (for unicode) after the delimiters.

```
<?php

//echoes 1 (adding u would not alter the result)
echo preg_match('/^áéíóúñ$/', 'áéíóúñ');

//echoes 0 (unless with [ó]+ or [ó]* or adding u)
echo preg_match('/^áéí[ó]úñ$/', 'áéíóúñ');

//so to match 'espana' or 'españa', add u or this won't match
//echoes 1
echo preg_match('/^espa[nñ]a$/u', 'españa');

?>
```

up
down

7
akniep at rayo dot info ¶
13 years ago

```
Bugs of preg_match (PHP-version 5.2.5)

In most cases, the following example will show one of two PHP-bugs discovered with preg_match depending on your PHP-version and configuration.

<?php

$text = "test=";
// creates a rather long text
for ($i = 0; $i++ < 100000;)
    $text .= "%AB";

// a typical URL_query validity-checker (the pattern's function does not matter for this example)
$pattern    = '/^(?:[;\/?:@&=+$,]|(?:[^\W_]|[-_.!~*\()\[\] ])|(?:%[\da-fA-F]{2}))*$/';

var_dump( preg_match( $pattern, $text ) );

?>

Possible bug (1):
=============
On one of our Linux-Servers the above example crashes PHP-execution with a C(?) Segmentation Fault(!). This seems to be a known bug (see
http://bugs.php.net/bug.php?id=40909), but I don't know if it has been fixed, yet.
If you are looking for a work-around, the following code-snippet is what I found helpful. It wraps the possibly crashing preg_match call by
decreasing the PCRE recursion limit in order to result in a Reg-Exp error instead of a PHP-crash.

<?php
[...]

// decrease the PCRE recursion limit for the (possibly dangerous) preg_match call
$former_recursion_limit = ini_set( "pcre.recursion_limit", 10000 );

// the wrapped preg_match call
$result = preg_match( $pattern, $text );

// reset the PCRE recursion limit to its original value
ini_set( "pcre.recursion_limit", $former_recursion_limit );
```

```
// if the reg-exp fails due to the decreased recursion limit we may not make any statement, but PHP-execution continues
if ( PREG_RECURSION_LIMIT_ERROR === preg_last_error() )
{
    // react on the failed regular expression here
    $result = [...];

    // do logging or email-sending here
    [...]
} //if

?>
```

```
Possible bug (2):
=============
On one of our Windows-Servers the above example does not crash PHP, but (directly) hits the recursion-limit. Here, the problem is that
preg_match does not return boolean(false) as expected by the description / manual of above.
In short, preg_match seems to return an int(0) instead of the expected boolean(false) if the regular expression could not be executed due to the
PCRE recursion-limit. So, if preg_match results in int(0) you seem to have to check preg_last_error() if maybe an error occurred.
```

up
down
3
Nimja ¶
10 years ago

```
When using a 'bad words reject string' filter, preg_match is MUCH faster than strpos / stripos. Because in the other cases, you would need to do
a foreach for each word. With efficient programming, the foreach is ONLY faster when the first word in the ban-list is found.

(for 12 words, 100,000 iterations, no word found)
stripos - Taken 1.4876 seconds.
strpos - Taken 1.4207 seconds.
preg_match - Taken 0.189 seconds.

Interesting fact:
With long words ('averylongwordtospitepreg'), the difference is only much less. Only about a 2/3rd of the time instead of 1/6th

<?php

$words = array('word1', 'word2', 'word3', 'word4', 'word5', 'word6', 'word7', 'word8', 'word9', 'word10', 'word11', 'word12' );
$teststring = 'ThIs Is A tEsTsTrInG fOr TeStInG.';
```

```
$count = 100000;
$find = 0;

$start = microtime(TRUE);
for ($i = 0; $i < $count; $i++) {
    foreach ($words as $word) {
        if (stripos($teststring, $word) !== FALSE) {
            $find++;
            break;
        }
    }
}
echo 'stripos - Taken ' . round(microtime(TRUE) - $start, 4) . ' seconds.' . PHP_EOL;

$start = microtime(TRUE);
for ($i = 0; $i < $count; $i++) {
    foreach ($words as $word) {
        if (strpos($teststring, $word) !== FALSE) {
            $find++;
            break;
        }
    }
}
echo 'strpos - Taken ' . round(microtime(TRUE) - $start, 4) . ' seconds.' . PHP_EOL;

$start = microtime(TRUE);
$pattern = '/';
$div = '';
foreach ($words as $word) {
    $pattern .= $div . preg_quote($word);
    $div = '|';
}
$pattern .= '/i';
//Pattern could easily be done somewhere else if words are static.
for ($i = 0; $i < $count; $i++) {
    if (preg_match($pattern, $teststring)) {
        $find++;
    }
}
```

```
$end = microtime(TRUE);
echo 'preg_match - Taken ' . round($end - $start, 4) . ' seconds.' . PHP_EOL;
?>
```

up
down
5

SoN9ne at gmail dot com ¶

12 years ago

```
I have been working on a email system that will automatically generate a text email from a given HTML email by using strip_tags().
The only issue I ran into, for my needs, were that the anchors would not keep their links.
I search for a little while and could not find anything to strip the links from the tags so I generated my own little snippet.
I am posting it here in hopes that others may find it useful and for later reference.

A note to keep in mind:
I was primarily concerned with valid HTML so if attributes do no use ' or " to contain the values then this will need to be tweaked.
If you can edit this to work better, please let me know.
<?php
/**
* Replaces anchor tags with text
* - Will search string and replace all anchor tags with text (case insensitive)
*
* How it works:
* - Searches string for an anchor tag, checks to make sure it matches the criteria
*         Anchor search criteria:
*             - 1 - <a (must have the start of the anchor tag )
*             - 2 - Can have any number of spaces or other attributes before and after the href attribute
*             - 3 - Must close the anchor tag
*
* - Once the check has passed it will then replace the anchor tag with the string replacement
* - The string replacement can be customized
*
* Know issue:
* - This will not work for anchors that do not use a ' or " to contain the attributes.
*         (i.e.- <a href=http: //php.net>PHP.net</a> will not be replaced)
*/
function replaceAnchorsWithText($data) {
    /**
     * Had to modify $regex so it could post to the site... so I broke it into 6 parts.
     */
    $regex  = '/(<a\s*'; // Start of anchor tag
    $regex .= '(.*?)\s*'; // Any attributes or spaces that may or may not exist
    $regex .= 'href=[\'"]+?\s*(?P<link>\S+)\s*[\'"]+?'; // Grab the link
    $regex .= '\s*(.*?)\s*>\s*'; // Any attributes or spaces that may or may not exist before closing tag
    $regex .= '(?P<name>\S+)'; // Grab the name
    $regex .= '\s*<\/a>)/i'; // Any number of spaces between the closing anchor tag (case insensitive)

    if (is_array($data)) {
        // This is what will replace the link (modify to you liking)
        $data = "{$data['name']}({$data['link']})";
    }
    return preg_replace_callback($regex, 'replaceAnchorsWithText', $data);
}

$input  = 'Test 1: <a href="http: //php.net1">PHP.NET1</a>.<br />';
$input .= 'Test 2: <A name="test" HREF=\'HTTP: //PHP.NET2\' target="_blank">PHP.NET2</A>.<BR />';
$input .= 'Test 3: <a hRef=http: //php.net3>php.net3</a><br />';
$input .= 'This last line had nothing to do with any of this';

echo replaceAnchorsWithText($input).'<hr/>';
?>
Will output:
Test 1: PHP.NET1(http: //php.net1).
Test 2: PHP.NET2(HTTP: //PHP.NET2).
Test 3: php.net3 (is still an anchor)
This last line had nothing to do with any of this

Posting to this site is painful...
Had to break up the regex and had to break the test links since it was being flagged as spam...
```

up
down
15
Yousef Ismaeil Cliprz ¶
9 years ago

```
Some times a Hacker use a php file or shell as a image to hack your website. so if you try to use move_uploaded_file() function as in example to
allow for users to upload files, you must check if this file contains a bad codes or not so we use this function. preg match

in this function we use
```

```
unlink() - http://php.net/unlink

after you upload file check a file with below function.

<?php

/**
* A simple function to check file from bad codes.
*
* @param (string) $file - file path.
* @author Yousef Ismaeil - Cliprz[at]gmail[dot]com.
*/
function is_clean_file ($file)
{
    if (file_exists($file))
    {
        $contents = file_get_contents($file);
    }
    else
    {
        exit($file." Not exists.");
    }

    if (preg_match('/(base64_|eval|system|shell_|exec|php_)/i',$contents))
    {
        return true;
    }
    else if (preg_match("#&\#x([0-9a-f]+);#i", $contents))
    {
        return true;
    }
    elseif (preg_match('#&\#([0-9]+);#i', $contents))
    {
        return true;
    }
    elseif (preg_match("#([a-z]*)=([\`\'\"]*)script:#iU", $contents))
    {
        return true;
    }
    elseif (preg_match("#([a-z]*)=([\`\'\"]*)javascript:#iU", $contents))
    {
        return true;
    }
    elseif (preg_match("#([a-z]*)=([\'\"]*)vbscript:#iU", $contents))
    {
        return true;
    }
    elseif (preg_match("#(<[^>]+)style=([\`\'\"]*).*expression\([^>]*>#iU", $contents))
    {
        return true;
    }
    elseif (preg_match("#(<[^>]+)style=([\`\'\"]*).*behaviour\([^>]*>#iU", $contents))
    {
        return true;
    }
    elseif (preg_match("#</*(applet|link|style|script|iframe|frame|frameset|html|body|title|div|p|form)[^>]*>#i", $contents))
    {
        return true;
    }
    else
    {
        return false;
    }
}
?>

Use

<?php
// If image contains a bad codes
$image   = "simpleimage.png";

if (is_clean_file($image))
{
    echo "Bad codes this is not image";
    unlink($image);
}
else
```

```
{
    echo "This is a real image.";
}
?>
```

up
down
6
Jonny 5 ¶
11 years ago

```
Workaround for getting the offset in UTF-8
(in some cases mb_strpos might be an option as well)

<?php
if(preg_match($pattern,$haystack,$out,PREG_OFFSET_CAPTURE)) {
    $offset = strlen(utf8_decode(substr($haystack,0,$out[0][1])));
}
?>
```

up
down
1
ASchmidt at Anamera dot net ¶
2 years ago

```
After the breaking change in 7.4, be aware that count( $matches ) may be different, depending on PREG_UNMATCHED_AS_NULL flag.

With PREG_UNMATCHED_AS_NULL, count( $matches ) will always be the maximum number of subpatterns.
However, without PREG_UNMATCHED_AS_NULL the $matches array will omit any unmatched subpatterns at the tail end.

Result forthe first two out of three matches with PREG_OFFSET_CAPTURE flag only:

array (size=3)
  0 =>
    array (size=2)
      0 => string 'may/02' (length=6)
      1 => int 0
  1 =>
    array (size=2)
      0 => string 'may' (length=3)
      1 => int 0
  2 =>
```

```
    array (size=2)
      0 => string '02' (length=2)
      1 => int 4

Result for two out of three matches with additional PREG_UNMATCHED_AS_NULL flags:

array (size=4)
  0 =>
    array (size=2)
      0 => string 'may/02' (length=6)
      1 => int 0
  1 =>
    array (size=2)
      0 => string 'may' (length=3)
      1 => int 0
  2 =>
    array (size=2)
      0 => string '02' (length=2)
      1 => int 4
  3 =>
    array (size=2)
      0 => null
      1 => int -1
```

up
down
7
ian_channing at hotmail dot com ¶
13 years ago

```
This is a function that uses regular expressions to match against the various VAT formats required across the EU.

<?php
/**
* @param integer $country Country name
* @param integer $vat_number VAT number to test e.g. GB123 4567 89
* @return integer -1 if country not included OR 1 if the VAT Num matches for the country OR 0 if no match
*/
function checkVatNumber( $country, $vat_number ) {
    switch($country) {
        case 'Austria':
            $regex = '/^(AT){0,1}U[0-9]{8}$/i';
            break;
        case 'Belgium':
            $regex = '/^(BE){0,1}[0]{0,1}[0-9]{9}$/i';
            break;
        case 'Bulgaria':
            $regex = '/^(BG){0,1}[0-9]{9,10}$/i';
            break;
        case 'Cyprus':
            $regex = '/^(CY){0,1}[0-9]{8}[A-Z]$/i';
            break;
        case 'Czech Republic':
            $regex = '/^(CZ){0,1}[0-9]{8,10}$/i';
            break;
        case 'Denmark':
            $regex = '/^(DK){0,1}([0-9]{2}[\ ]{0,1}){3}[0-9]{2}$/i';
            break;
        case 'Estonia':
        case 'Germany':
        case 'Greece':
        case 'Portugal':
            $regex = '/^(EE|EL|DE|PT){0,1}[0-9]{9}$/i';
            break;
        case 'France':
            $regex = '/^(FR){0,1}[0-9A-Z]{2}[\ ]{0,1}[0-9]{9}$/i';
            break;
        case 'Finland':
        case 'Hungary':
        case 'Luxembourg':
        case 'Malta':
        case 'Slovenia':
            $regex = '/^(FI|HU|LU|MT|SI){0,1}[0-9]{8}$/i';
            break;
        case 'Ireland':
            $regex = '/^(IE){0,1}[0-9][0-9A-Z\+\*][0-9]{5}[A-Z]$/i';
            break;
        case 'Italy':
        case 'Latvia':
            $regex = '/^(IT|LV){0,1}[0-9]{11}$/i';
            break;
        case 'Lithuania':
            $regex = '/^(LT){0,1}([0-9]{9}|[0-9]{12})$/i';
            break;
        case 'Netherlands':
            $regex = '/^(NL){0,1}[0-9]{9}B[0-9]{2}$/i';
            break;
        case 'Poland':
        case 'Slovakia':
            $regex = '/^(PL|SK){0,1}[0-9]{10}$/i';
            break;
        case 'Romania':
            $regex = '/^(RO){0,1}[0-9]{2,10}$/i';
            break;
        case 'Sweden':
            $regex = '/^(SE){0,1}[0-9]{12}$/i';
            break;
        case 'Spain':
            $regex = '/^(ES){0,1}([0-9A-Z][0-9]{7}[A-Z])|([A-Z][0-9]{7}[0-9A-Z])$/i';
            break;
        case 'United Kingdom':
            $regex = '/^(GB){0,1}([1-9][0-9]{2}[\ ]{0,1}[0-9]{4}[\ ]{0,1}[0-9]{2})|([1-9][0-9]{2}[\ ]{0,1}[0-9]{4}[\ ]{0,1}[0-9]{2}[\ ]{0,1}[0-
9]{3})|((GD|HA)[0-9]{3})$/i';
            break;
        default:
            return -1;
            break;
    }

    return preg_match($regex, $vat_number);
}
?>
```

up
down
1
jphansen at uga dot edu ¶
10 years ago

```
Here's a regex to validate against the schema for common MySQL
identifiers:
```

```
<?php
$string = "$table_name";
if (preg_match("/[^\\d\\sa-zA-Z$_]/", $string))
  echo "Failed validation";
?>
```

up
down
2
teracci2002 ¶
12 years ago

```
When you use preg_match() for security purpose or huge data processing,
mayby you should make consideration for backtrack_limit and recursion_limit.
http://www.php.net/manual/en/pcre.configuration.php

These limits may bring wrong matching result.
You can verify whether you hit these limits by checking preg_last_error().
http://www.php.net/manual/en/function.preg-last-error.php
```

up
down
2
ayman2243 at gmail dot com ¶
11 years ago

```
highlight Search Words

<?php
function highlight($word, $subject) {

    $split_subject = explode(" ", $subject);
    $split_word = explode(" ", $word);

    foreach ($split_subject as $k => $v){
           foreach ($split_word as $k2 => $v2){
               if($v2 == $v){

                   $split_subject[$k] = "<span class='highlight'>".$v."</span>";

               }
           }
```

```
        }

        return implode(' ', $split_subject);
}
?>
```

up
down
2
***Frank* ¶**
11 years ago

If someone is from a country that accepts decimal numbers in format 9.00 and 9,00 (point or comma), number validation would be like that:

```
<?php
$number_check = "9,99";
if (preg_match( '/^[\-+]?[0-9]*\.*\,?[0-9]+$/', $number_check)) {
    return TRUE;
}
?>
```

However, if the number will be written in the database, most probably this comma needs to be replaced with a dot.
This can be done with use of str_replace, i.e :

```
<?php
$number_database = str_replace("," , "." , $number_check);
?>
```

up
down
2
***matt* ¶**
13 years ago

To support large Unicode ranges (ie: [\x{E000}-\x{FFFD}] or \x{10FFFFF}) you must use the modifier '/u' at the end of your expression.
up
down
1
***Stefan* ¶**
13 years ago

I spent a while replacing all my ereg() calls to preg_match(), since ereg() is now deprecated and will not be supported as of v 6.0.

Just a warning regarding the conversion, the two functions behave very similarly, but not exactly alike. Obviously, you will need to delimit your pattern with '/' or '|' characters.

```
The difference that stumped me was that preg_replace overwrites the $matches array regardless if a match was found. If no match was found,
$matches is simply empty.

ereg(), however, would leave $matches alone if a match was not found. In my code, I had repeated calls to ereg, and was populating $matches with
each match. I was only interested in the last match. However, with preg_match, if the very last call to the function did not result in a match,
the $matches array would be overwritten with a blank value.

Here is an example code snippet to illustrate:

<?php
$test = array('yes','no','yes','no','yes','no');

foreach ($test as $key=>$value) {
  ereg("yes",$value,$matches1);
  preg_match("|yes|",$value,$matches2);
}
  print "ereg result: $matches1[0]<br>";
  print "preg_match result: $matches2[0]<br>";
?>

The output is:
ereg result: yes
preg_match result:

($matches2[0] in this case is empty)

I believe the preg_match behavior is cleaner. I just thought I would report this to hopefully save others some time.
```

up
down
2

wjaspers4 [at] gmail [dot] com ¶

13 years ago

```
I recently encountered a problem trying to capture multiple instances of named subpatterns from filenames.
Therefore, I came up with this function.

The function allows you to pass through flags (in this version it applies to all expressions tested), and generates an array of search results.

Enjoy!
```

```
<?php

/**
* Allows multiple expressions to be tested on one string.
* This will return a boolean, however you may want to alter this.
*
* @author William Jaspers, IV <wjaspers4@gmail.com>
* @created 2009-02-27 17:00:00 +6:00:00 GMT
* @access public
*
* @param array $patterns An array of expressions to be tested.
* @param String $subject The data to test.
* @param array $findings Optional argument to store our results.
* @param mixed $flags Pass-thru argument to allow normal flags to apply to all tested expressions.
* @param array $errors A storage bin for errors
*
* @returns bool Whether or not errors occurred.
*/
function preg_match_multiple(
  array $patterns=array(),
  $subject=null,
  &$findings=array(),
  $flags=false,
  &$errors=array()
) {
  foreach( $patterns as $name => $pattern )
  {
    if( 1 <= preg_match_all( $pattern, $subject, $found, $flags ) )
    {
      $findings[$name] = $found;
    } else
    {
      if( PREG_NO_ERROR !== ( $code = preg_last_error() ))
      {
        $errors[$name] = $code;
      } else $findings[$name] = array();
    }
  }
  return (0===sizeof($errors));
}
?>
```

up
down
2
itworkarounds at gmail dot com ¶
11 years ago

```
You can use the following code to detect non-latin (Cyrilic, Arabic, Greek...) characters:

<?php
preg_match("/^[a-zA-Z\p{Cyrillic}0-9\s\-]+$/u", "ABC abc 1234 АБВ абв");
?>
```

up
down
1
Anonymous ¶
12 years ago

```
The regular expression for breaking-down a URI reference into its components:

      ^(([^:/?#]+):)?(//([^/?#]*))?([^?#]*)(\?([^#]*))?(#(.*))?
       12            3  4          5       6  7        8 9

Source: ietf.org/rfc/rfc2396.txt
```

up
down
2
matt at proweb dot co dot uk ¶
1 year ago

```
pcre2-migration

Status: Implemented (in PHP 7.3)

SELinux will prevent PREG_* functions from working

Feb  8 12:40:51 servername setroubleshoot: SELinux is preventing httpd from using the execmem access on a process.

you need to add preg.jit=0 to php.ini or init_set('preg.jit', 0) if you can't do that

try the [PCRE] section so you can find it
```

up
down
2
skds1433 at hotmail dot com ¶
13 years ago
here is a small tool for someone learning to use regular expressions. it's very basic, and allows you to try different patterns and combinations. I made it to help me, because I like to try different things, to get a good understanding of how things work.

```
<?php
$search = isset($_POST['search'])?$_POST['search']:"//";
$match = isset($_POST['match'])?$_POST['match']:"<>";

echo '<form method="post">';
echo 's: <input style="width:400px;" name="search" type="text" value="'.$search.'" /><br />';
echo 'm:<input style="width:400px;" name="match" type="text" value="'.$match.'" /><input type="submit" value="go" /></form><br />';
if (preg_match($search, $match)){echo "matches";}else{echo "no match";}
?>
```

up
down
1
marcosc at tekar dot net ¶
13 years ago
When using accented characters and "ñ" (áéíóúñ), preg_match does not work. It is a charset problem, use utf8_decode/decode to fix.
up
down
0
chris at ocproducts dot com ¶
2 years ago
If PREG_OFFSET_CAPTURE is set then unmatched captures (i.e. ones with '?') will not be present in the result array. This is presumably because there is no offset, and thus the original PHP dev decided best to just leave it out.
up
down
0
ASchmidt at Anamera dot net ¶
2 years ago
Combining flags
PREG_OFFSET_CAPTURE | PREG_UNMATCHED_AS_NULL
will NOT result in a value of NULL for any submatched subpattern.

```
Instead still results in an array for each unmatched subpattern, in turn containing:

array (size=2)
      0 => null
      1 => int -1

Consequently your code needs to expect NULL as the string value:
$matches[ {subpattern+1} ][0] === null
and/or a negative string offset:
$matches[ {subpattern+1} ][1] < 0
to detect any unmatched subpattern!
```

up
down
1
***plasma* ¶**
12 years ago

```
To extract scheme, host, path, ect. simply use

<?php

  $url  = 'http://name:pass@';
  $url .= 'example.com:10000';
  $url .= '/path/to/file.php?a=1&amp;b=2#anchor';

  $url_data = parse_url ( $url );

  print_r ( $url_data );

?>
___
prints out something like:

Array
(
    [scheme] => http
    [host] => wild.subdomain.orgy.domain.co.uk
    [port] => 10000
    [user] => name
    [pass] => pass
    [path] => /path/to/file.php
    [query] => a=1&b=2
    [fragment] => anchor
)

In my tests parse_url is up to 15x faster than preg_match(_all)!
```

up
down
2
chat dot noir at arcor dot de ¶
5 years ago

```
Note that if a parenthesed group is not matched, its key may or may not be present in $matches. For instance,

<?php preg_match('/(foo)?(bar)?(baz)?/', 'bar', $matches);
print_r($matches);

// outputs
// Array
// (
//     [0] => bar
//     [1] =>
//     [2] => bar
// )
?>
Note that there is no element with key '3' in $matches, but an element with key '1' (the empty string). This inconsistent behavior also applies
to named groups.
```

up
down
-3
xcsv at gmx dot net ¶
2 years ago

```
As of PHP 7.2, you can use the following.

If you work with named subpatterns and dont want to bother with unnamed match result entries and unmatched subpatterns, just replace
preg_match() with named_preg_match(). This filters all unwanted stuff out.

<?php
function named_preg_match(string $pattern , string $subject, array &$matches = null, int $flags = 0, int $offset = 0) {
    $retval = preg_match($pattern, $subject, $localmatches, PREG_UNMATCHED_AS_NULL | $flags, $offset);
    if ($retval) {
        foreach ($localmatches as $key => $value) {
            if (is_int($key)) $value = null;
            if (is_null($value)) unset($localmatches[$key]);
        }
        $matches = $localmatches;
    }
    return $retval;
}
?>
```

Hope this will be useful.

up

down

-2

phil dot taylor at gmail dot com ¶

14 years ago

If you need to check for .com.br and .com.au and .uk and all the other crazy domain endings i found the following expression works well if you want to validate an email address. Its quite generous in what it will allow

```
<?php

        $email_address = "phil.taylor@a_domain.tv";

    if (preg_match("/^[^@]*@[^@]*\.[^@]*$/", $email_address)) {
        return "E-mail address";
    }

?>
```

up

down

-5

sun at drupal dot org ¶

11 years ago

Basic test for invalid UTF-8 that can hi-jack IE:

```
<?php
$valid = (preg_match('/^./us', $text) == 1);
?>
```

```
See http://api.drupal.org/api/drupal/includes--bootstrap.inc/function/drupal_validate_utf8/7 for details.

---

Test for valid UTF-8 and XML/XHTML character range compatibility:

<?php
$invalid = preg_match('@[^\x9\xA\xD\x20-\x{D7FF}\x{E000}-\x{FFFD}\x{10000}-\x{10FFFF}]@u', $text)
?>
Ref: http://www.w3.org/TR/2000/REC-xml-20001006#charsets
```

+ add a note

- PCRE Functions
  - preg_filter
  - preg_grep
  - preg_last_error_msg
  - preg_last_error
  - preg_match_all
  - preg_match
  - preg_quote
  - preg_replace_callback_array
  - preg_replace_callback
  - preg_replace
  - preg_split