PHP Regex

Intro PHP

Download example code from Nexus

A quick refresher

Regular expressions allow us to find patterns of characters within strings, or to match complete strings to a pattern.

What’s a pattern?

A pattern can be a simple literal string, or it can be a generic class of characters that may or may not repeat.

```
/my cat/ - matches first instance of‘my cat`
/^my cat$/ - matches if full string equals `my cat`
/dog|cat/ - matches first instance of `dog` or `cat`
/colou?r/ - matches US or Canadian spelling of `colour`
/[A-z]+/ - matches 1 or more alphabetical characters
/[0-9]$/ - matches a digit at end of string
/^[0-9]/ - matches a digit at start of string
```

01_refresher.php

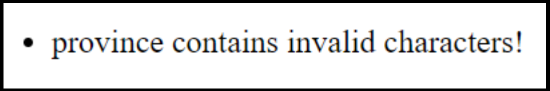
Validating user data with regex

Regular expressions can be a powerful tool when validating data that may be tainted.

```php
$errors = [];
$form = [
    'name' => 'John Davis-Walker 3rd',
    'city' => 'Winnipeg',
    'province' => 'Manitoba!'
];
$pattern = '/^[A-Za-z0-9\s\-\']{1,64}$/';

foreach($form as $key => $value) {
    if(!preg_match($pattern, $value)) {
        $errors[$key] = "$key contains invalid 
characters!";
    }
}
```

01_refresher.php

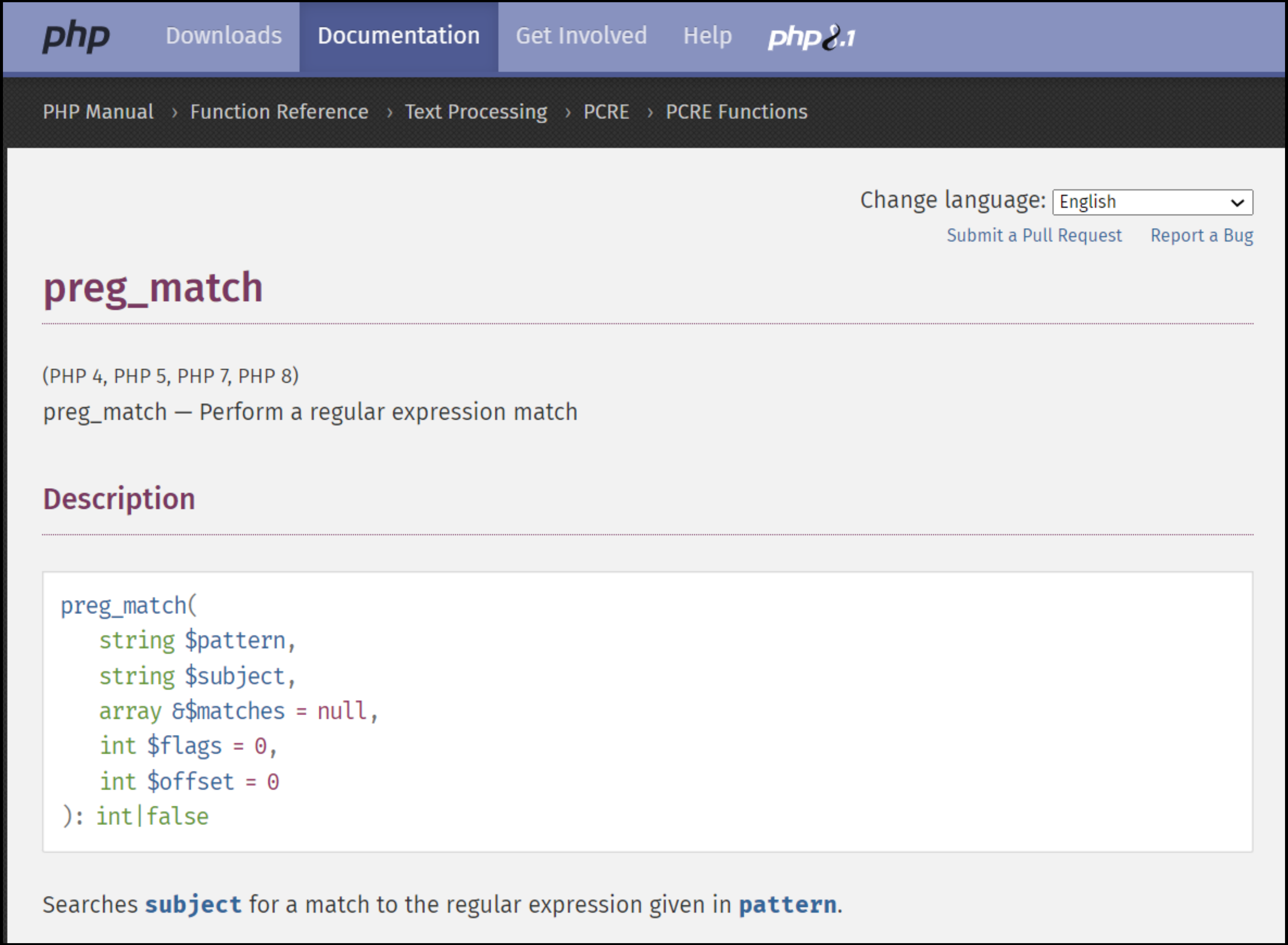
The preg_match() function allows us to do basic pattern matching in PHP.



- province contains invalid characters!

In this case, the province field value contains an invalid character: !

PHP Regex



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**.

Capture groups

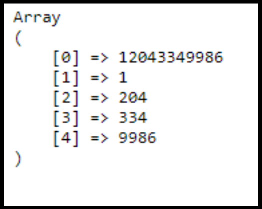
The $matches parameter in preg_match() allows us to capture the information we are trying to match,.

```
$pattern =
'/^([\d])?([\d]{3})([\d]{3})([\d]{4})$/';
    preg_match($pattern, $num, $matches);
```

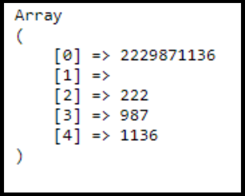
For the following numbers, $matches captures the results to the right. All three numbers are valid according to our pattern.

1 204.334 9986
222-987-1136
(512) 993 1234

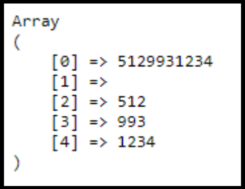
02_capture_groups.php



```
Array
(
    [0] => 12043349986
    [1] => 1
    [2] => 204
    [3] => 334
    [4] => 9986
)
```



```
Array
(
    [0] => 2229871136
    [1] =>
    [2] => 222
    [3] => 987
    [4] => 1136
)
```

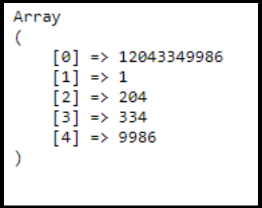


```
Array
(
    [0] => 5129931234
    [1] =>
    [2] => 512
    [3] => 993
    [4] => 1234
)
```

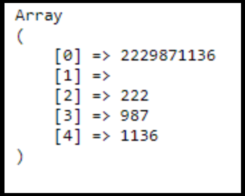
Capture groups

What does $matches contain?

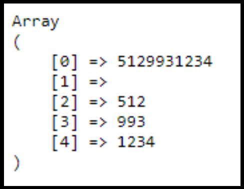
```
$pattern =
'/^([\d])?([\d]{3})([\d]{3})([\d]{4})$/';
    preg_match($pattern, $num, $matches);
```



```
Array
(
    [0] => 12043349986
    [1] => 1
    [2] => 204
    [3] => 334
    [4] => 9986
)
```



```
Array
(
    [0] => 2229871136
    [1] =>
    [2] => 222
    [3] => 987
    [4] => 1136
)
```



```
Array
(
    [0] => 5129931234
    [1] =>
    [2] => 512
    [3] => 993
    [4] => 1234
)
```

0 => the full match, if there is one
1 => the first capture group
2 => the second capture group
3 => the third capture group
4 => the fourth capture group

If $matches[0] is empty, then the string failed to match our pattern.

02_capture_groups.php

Assertions

Assertions allow us to inspect a string and determine that it meets certain requirements, before we begin testing the entire string for a match.

```
$pattern = '/(?=.*[A-Z]+)(?=.*[a-z]+)(?=.*[0-9]=)(?=.*[[:punct:]]+).{8,}/';
```

The pattern above contains four assertions before the basic string pattern. The assertions are contained within parenthesis and begin with ?=, which means ‘look ahead’. In other words, look ahead into the string and try and find this match. If the match is not found the expression fails without having to look any further.

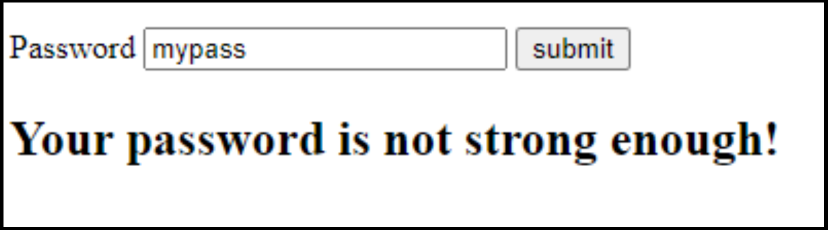
The assertions above are looking for: 1 or more uppercase letters, 1 or more lowercase letters, 1 or more digits, and 1 or more special characters (punctuation)… all within a string that must be at least 8 characters long.

03_assertions.php

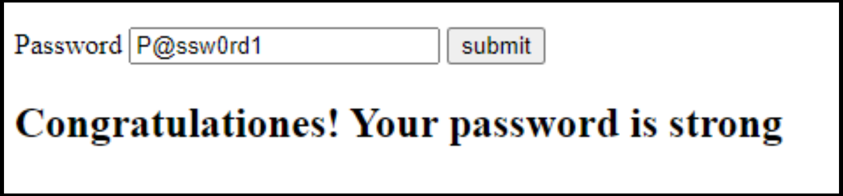
Assertions

In other words… a strong password!

```
$pattern = '/(?=.*[A-Z]+)(?=.*[a-z]+)(?=.*[0-9]=)(?=.*[[:punct:]]+).{8,}/';
```


Password mypass submit

Your password is not strong enough!


Password P@ssw0rd1 submit

Congratulationes! Your password is strong

03_assertions.php

preg_replace()

The extreme flexibility that regular expressions afford in matching classes of characters, allows us to perform some complicated search and replace functions with ease.

```
$pattern = '/[^\d]/';
$num = preg_replace($pattern, '', $phone);
```

The pattern above matches any character that is not a digit.

1 204.334 9986 will become 12043349986
222-987-1136 will become 2229871136
(512) 993 1234 will become 5129921234

In this case, a series of digits without any other characters are much easier to validated as a phone number than a string with many different character types.

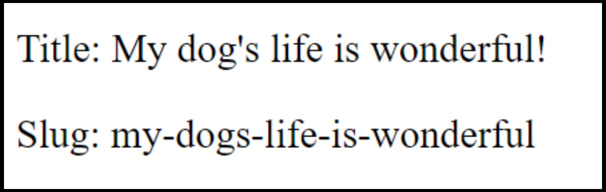
04_capture_groups.php

preg_replace()

Another common use case for preg_replace() is to generate URL friendly slugs from complex strings.

```php
$title = "My dog's     life    is     wonderful!";
$rep = preg_replace('/[^A-z\ \-]/', '', $title);
$slug = strtolower( preg_replace('/[\s]+/', '-', $rep) );
```

The pattern above matches any character that is not a alphabetical, a space, or a dash, and removes it.



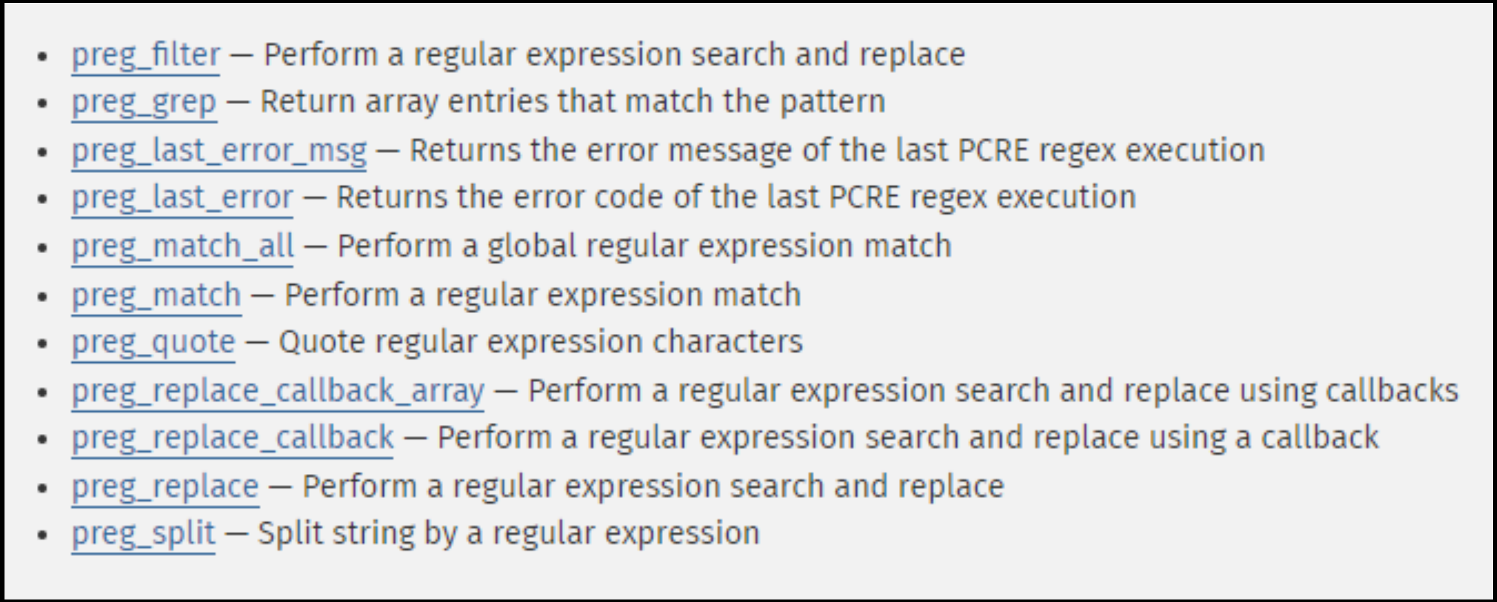
```
Title: My dog's life is wonderful!

Slug: my-dogs-life-is-wonderful
```

In this case, we can convert a messy string… for example, the title of an article or blog post… and convert it to a URL friendly slug.

04_capture_groups.php

Other PHP Regex Functions

PHP comes with a variety of regex functions that can perform useful tasks. We've covered only two in these slides, but you should explore the others and become familiar with them.



- preg_filter — Perform a regular expression search and replace
- preg_grep — Return array entries that match the pattern
- preg_last_error_msg — Returns the error message of the last PCRE regex execution
- preg_last_error — Returns the error code of the last PCRE regex execution
- preg_match_all — Perform a global regular expression match
- preg_match — Perform a regular expression match
- preg_quote — Quote regular expression characters
- preg_replace_callback_array — Perform a regular expression search and replace using callbacks
- preg_replace_callback — Perform a regular expression search and replace using a callback
- preg_replace — Perform a regular expression search and replace
- preg_split — Split string by a regular expression

https://www.php.net/manual/en/ref.pcre.php

Next:

Functions