Regular Expressions

Introduction

- A regular expression is a string that describes a set of matching rules.
- The regular expression are highly useful when we don't know the exact string we want to match.
- PHP supports PCRE (Perl Compatible Regular Expressions).

Delimiters

- Every regular expression must be delimited by a character we choose both for its starting and ending. It can be any character as long as it isn't a character we use within the regular expression itself.
- Using the forward slash '/' as a delimiter is a common convention.

Meta Characters

- A meta character represents a single character in the matched expression.
- The most popular meta characters are:

.	match any character
\s	match any whitespace character
\d	match any digit
\w	match any “word” character

Character Class

- A character class is a series of valid alternatives for a character.
- We denote a character class by using square brackets:

```
[a-d\d]       'a' or 'b' or 'c' or 'd' or a digit
[1-3\w]       any word character or '1' or '2' or '3'
```

- The character class can be part of a regular expression:

```
/ab[1-4]/     'ab1' or 'ab2' or 'ab3' or 'ab4'
/abc[#@!]/    'abc#' or 'abc@' or 'abc!'
```

Quantifiers

- A quantifier specifies the number of occurrences a specific character or a specific meta-character can appear in a matching string.
- There are four types of quantifier:

*	The character can appear zero or more times.
+	The character can appear one or more times.
?	The character can appear zero or one times.
{n,m}	The character can appear at least n times and no more than m. Omitting m means at least n times. Omitting n means no more than m times.

Sub Expressions

- A sub expression is a regular expression contained within another regular expression.
- A sub expression is defined by encapsulating it within parentheses.

```
/a(fgh.)f/
```

This regular expression will match any combination that starts with the letter 'a' following with “fgh” following with any character and ends with 'f'.

- A sub expression can be used in conjunction with a quantifier.

```
/a(fg)?b/
```

This regular expression will match “afgb” and “ab”.

The `preg_match()` Function

- The `preg_match()` function checks if there is a match between a regular expression and a given string.

```
int preg_match ( string $pattern , string $subject
        [, array &$matches [, int $flags [, int $offset ]]] )
```

This function checks if `$subject`, the string, matches `$pattern`, the regular expression.

The `preg_match()` Function

```
<?php
if (preg_match("/hello/", "Hello students! You had good time."))
{
    echo "A match was found.";
}
else
{
    echo "A match was not found.";
}
?>
```

The `preg_match()` Function

The `str_replace()` Function

- The `str_replace()` function replaces all matches with a string we specify.

```
mixed str_replace ( mixed $search , mixed $replace ,
                                    mixed $subject [, int &$count ] )
```

This function looks for strings matching with the `$search` text and replaces each one of them with the `$replace` text. This function works on the `$subject` text.

The `preg_replace()` Function

- The `preg_replace()` function replaces all matches with a regular expression we specify.

```
mixed preg_replace ( mixed $pattern , mixed $replacement ,
                     mixed $subject [, int $limit [, int &$count ]] )
```

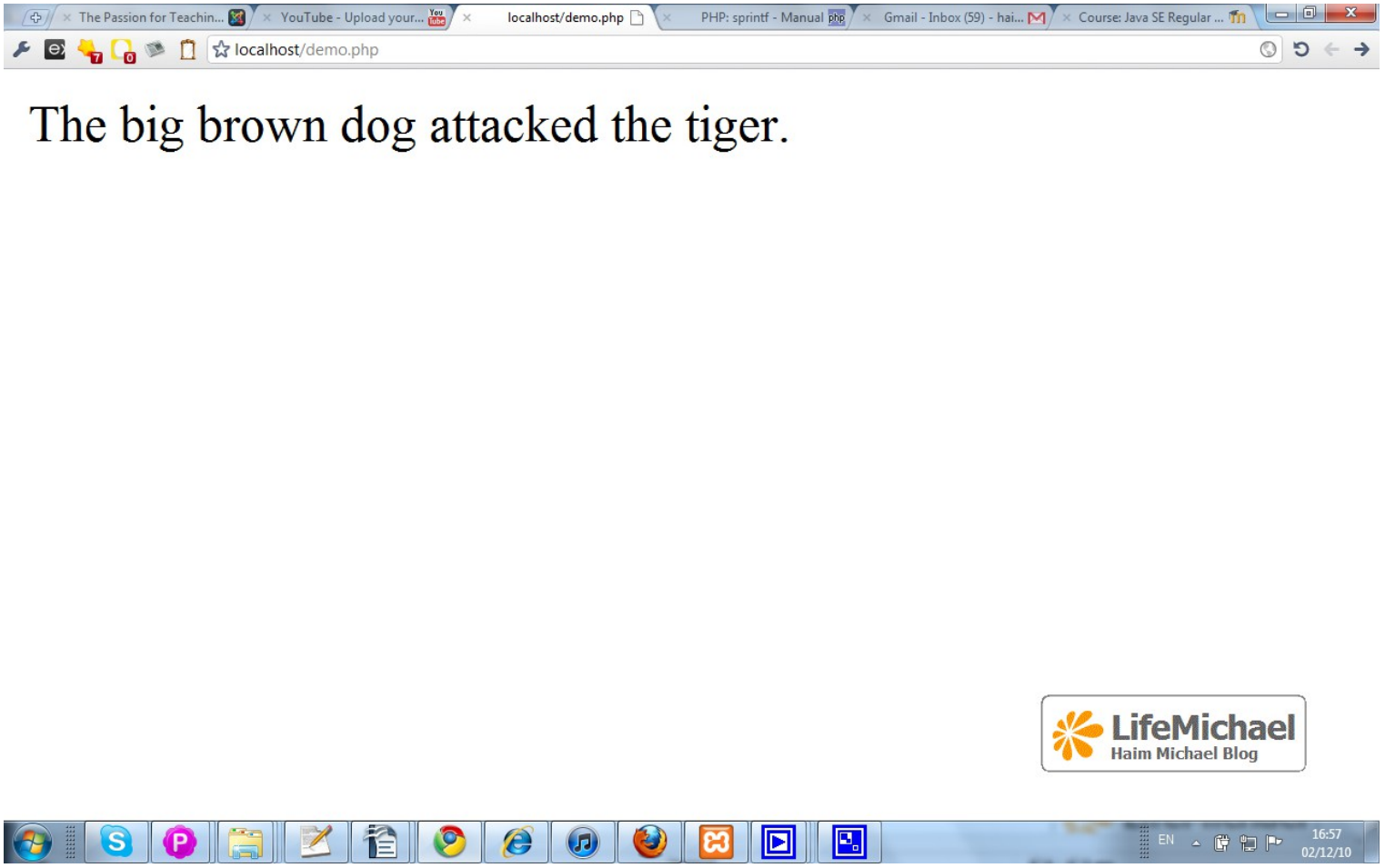
This function looks for matches with `$pattern` regular expression and replaces them with `$replacement` text. This function works on the $subject text.

- It is possible to pass arrays (instead of single values) for `$pattern` and `$replacement`.

The `preg_replace()` Function

```
<?php
$string = 'The small brown cat attacked the mouse.';
$patterns[0] = '/small/';
$patterns[1] = '/cat/';
$patterns[2] = '/mouse/';
$replacements[2] = 'big';
$replacements[1] = 'dog';
$replacements[0] = 'tiger';
echo preg_replace($patterns, $replacements, $string);
?>
```

The `preg_replace()` Function



The `preg_replace_callback_array()` Function

- As of PHP 7 we can call this function passing over an array as the first argument (keys are regular expressions and values are the functions we want to execute for each one of them) and the input string as the second one.

The `preg_replace_callback_array()` Function

```
<?php

$str = "we love java, php, samba, snowboarding and peace! jave";

preg_replace_callback_array( [

    '/php/' => function($match){
        echo "<h1>".var_dump($match)." match with php was found</h1>";
    },

    '/jav[a-g]/' => function($match){
        echo "<h1>".var_dump($match)."$match match with jav[a-g] was found</h1>";
    },

] , $str);


?>
```

The `preg_replace_callback_array()` Function