UNIT – IV PHP

Regular Expression

A regular expression is a sequence of characters that forms a search pattern. When you search for data in a text, you can use this search pattern to describe what you are searching for. A regular expression can be a single character, or a more complicated pattern. Regular expressions can be used to perform all types of text search and text replace operations.

In PHP, regular expressions are strings composed of delimiters, a pattern and optional modifiers.

$exp = "/w3schools/i";

Regular Expression Functions

PHP provides a variety of functions that allow you to use regular expressions.

The most common functions are:

Function	Description
preg_match()	Returns 1 if the pattern was found in the string and 0 if not
preg_match_all()	Returns the number of times the pattern was found in the string, which may als
preg_replace()	Returns a new string where matched patterns have been replaced with another s

Using preg_match()

The preg_match() function will tell you whether a string contains matches of a pattern.

Use a regular expression to do a case-insensitive search for "w3schools" in a string:

$str = "Visit W3Schools";

$pattern = "/w3schools/i";

echo preg_match($pattern, $str);

Using preg_match_all()

The preg_match_all() function will tell you how many matches were found for a pattern in a string.

Use a regular expression to do a case-insensitive count of the number of occurrences of "ain" in a string:

$str = "The rain in SPAIN falls mainly on the plains.";

$pattern = "/ain/i";

echo preg_match_all($pattern, $str);

Using preg_replace()

The preg_replace() function will replace all of the matches of the pattern in a string with another string.

Use a case-insensitive regular expression to replace Microsoft with W3Schools in a string:

$str = "Visit Microsoft!";

$pattern = "/microsoft/i";

echo preg_replace($pattern, "W3Schools", $str);

Regular Expression Modifiers

Modifiers can change how a search is performed.

Modifier	**Description**
I	Performs a case-insensitive search
M	Performs a multiline search (patterns that search for a match at the beginning or end of a string will now match the beginning or end of *each line*)
u	Enables correct matching of UTF-8 encoded patterns

Regular Expression Patterns

Brackets are used to find a range of characters:

Expression	**Description**
[abc]	Find one or many of the characters inside the brackets
[^abc]	Find any character NOT between the brackets
[a-z]	Find any character alphabetically between two letters
[A-z]	Find any character alphabetically between a specified upper-case letter and a specified lower-case letter
[A-Z]	Find any character alphabetically between two upper-case letters.
[123]	Find one or many of the digits inside the brackets
[0-5]	Find any digits between the two numbers
[0-9]	Find any digits

Metacharacters

Metacharacters are characters with a special meaning:

Metacharacter	**Description**
|	Find a match for any one of the patterns separated by | as in: cat|dog|fish
.	Find any character
^	Finds a match as the beginning of a string as in: ^Hello
$	Finds a match at the end of the string as in: World$
\d	Find any digits
\D	Find any non-digits
\s	Find any whitespace character
\S	Find any non-whitespace character
\w	Find any alphabetical letter (a to Z) and digit (0 to 9)
\W	Find any non-alphabetical and non-digit character

\b	Find a match at the beginning of a word like this: \bWORD, or at the end of a word like this: WORD\b
\uxxxx	Find the Unicode character specified by the hexadecimal number xxxx

Quantifiers

Quantifiers define quantities:

Quantifier	**Description**
n+	Matches any string that contains at least one *n*
*n**	Matches any string that contains zero or more occurrences of *n*
n?	Matches any string that contains zero or one occurrences of *n*
n{*3*}	Matches any string that contains a sequence of *3* *n*'s
n{*2*, *5*}	Matches any string that contains a sequence of at least 2, but not more that 5 *n*'s
n{3,}	Matches any string that contains a sequence of at least 3 *n*'s

Example : 1

```
<?php
$input = [
  "Red",
  "Pink",
  "Green",
  "Blue",
  "Purple"
];

$result = preg_grep("/^p/i", $input);
print_r($result);
?>
```

Example: 2

```php
<?php
$str = "The rain in SPAIN falls mainly on the plains.";
$pattern = "/ain/i";
if(preg_match_all($pattern, $str, $matches)) {
  print_r($matches);
}
?>
```