PHP Regular Expression Tutorial

1. Basic Syntax:

In PHP, regular expressions are typically enclosed in delimiters. The most common delimiters are forward slashes '/' or curly braces '{}' or pips '|':

```
$pattern = '/pattern/';
$pattern = '{pattern}';
$pattern = '|pattern|';
```

2. Matching Functions:

PHP provides several functions for working with regex:

```
- preg_match(): Checks if a pattern matches a string
- preg_match_all(): Finds all occurrences of a pattern in a string
- preg_replace(): Replaces matches with a specified string
- preg_split(): Splits a string by a regular expression
```

3. Simple Pattern Matching:

```
$text = "Hello, World!";
$pattern = '/Hello/';
if (preg_match($pattern, $text)) {
    echo "Match found!";
}
```

4. Character Classes:

- `[abc]`: Matches any single character in the set
- `[^abc]`: Matches any single character not in the set
- `[a-z]`: Matches any single character in the range

```
// Character Classes
echo "Character Classes:<br>";

// [abc]: Matches any single character in the set
$pattern = '/[aeiouAEIOU]/';
$text = "Hello World";
preg_match_all($pattern, $text, $matches);
echo "Text: " . $text . "<br>";
echo "[aeiou]: " . implode(', ', $matches[0]) . "<br><br>";

// [^abc]: Matches any single character not in the set
$pattern = '/[^aeiou]/';
$text = "Hello World";
preg_match_all($pattern, $text, $matches);
```

```php
echo "[^aeiou]: " . implode(', ', $matches[0]) . "<br><br>";

// [a-z]: Matches any single character in the range
$pattern = '/[a-mA-M]/';
$text = "Hello World";
preg_match_all($pattern, $text, $matches);
echo "[a-m]: " . implode(', ', $matches[0]) . "<br><br>";
```

5. Metacharacters:

- . (dot): Matches any single character except newline
- \d: Matches any digit (0-9)
- \w: Matches any word character (a-z, A-Z, 0-9, _)
- \s: Matches any whitespace character

```php
// Metacharacters
echo "Metacharacters:<br>";

// . (dot): Matches any single character except newline
$pattern = '/h.l/';
$text = "hello help hull h&ll";
preg_match_all($pattern, $text, $matches);
echo "Text: " . $text . "<br>";
echo ". (dot): " . implode(', ', $matches[0]) . "<br><br>";

// \d: Matches any digit (0-9)
$pattern = '/\d/';
$text = "There are 123 apples and 456 oranges";
preg_match_all($pattern, $text, $matches);
echo "Text: " . $text . "<br>";
echo "\\d: " . implode(', ', $matches[0]) . "<br><br>";

// \w: Matches any word character (a-z, A-Z, 0-9, _)
$pattern = '/\w/';
$text = "Hello_World 123!";
preg_match_all($pattern, $text, $matches);
echo "Text: " . $text . "<br>";
echo "\\w: " . implode(', ', $matches[0]) . "<br><br>";

// \s: Matches any whitespace character
$pattern = '/\s/';
$text = "Hello World\tTab\nNewline";
preg_match_all($pattern, $text, $matches);
echo "Text: " . $text . "<br>";
echo "\\s: " . count($matches[0]) . " whitespace characters found<br><br>";
```

6. Quantifiers:

- *: Matches 0 or more occurrences
- +: Matches 1 or more occurrences
- ?: Matches 0 or 1 occurrence
- {n}: Matches exactly n occurrences
- {n,}: Matches n or more occurrences
- {n,m}: Matches between n and m occurrences

```php
// Quantifiers
echo "Quantifiers:<br>";

// *: Matches 0 or more occurrences
$pattern = '/ab*c/';
$text = "ac abc abbc abbbc";
preg_match_all($pattern, $text, $matches);
echo "*: " . implode(', ', $matches[0]) . "<br><br>";

// +: Matches 1 or more occurrences
$pattern = '/ab+c/';
$text = "ac abc abbc abbbc";
preg_match_all($pattern, $text, $matches);
echo "+: " . implode(', ', $matches[0]) . "<br><br>";

// ?: Matches 0 or 1 occurrence
$pattern = '/ab?c/';
$text = "ac abc abbc";
preg_match_all($pattern, $text, $matches);
echo "?: " . implode(', ', $matches[0]) . "<br><br>";

// {n}: Matches exactly n occurrences
$pattern = '/a{3}/';
$text = "a aa aaa aaaa";
preg_match_all($pattern, $text, $matches);
echo "{n}: " . implode(', ', $matches[0]) . "<br><br>";

// {n,}: Matches n or more occurrences
$pattern = '/a{2,}/';
$text = "a aa aaa aaaa";
preg_match_all($pattern, $text, $matches);
echo "{n,}: " . implode(', ', $matches[0]) . "<br><br>";

// {n,m}: Matches between n and m occurrences
$pattern = '/a{2,3}/';
$text = "a aa aaa aaaa";
preg_match_all($pattern, $text, $matches);
```

```php
echo "{n,m}: " . implode(', ', $matches[0]) . "<br><br>";
```

7. Anchors:

- ^: Matches the start of a string
- $: Matches the end of a string

```php
// Anchors
echo "Anchors:<br>";

// ^: Matches the start of a string
$pattern = '/^Hello/';
$text = "Hello World\nHello Universe";
preg_match_all($pattern, $text, $matches);
echo "^: " . implode(', ', $matches[0]) . "<br><br>";

// $: Matches the end of a string
$pattern = '/World$/m';
$text = "Hello World\nHello Universe";
preg_match_all($pattern, $text, $matches);
echo "$: " . implode(', ', $matches[0]) . "<br><br>";
```

8. Example: Validating an Email Address

```php
$email = "user@example.com";
$pattern = '/^[\w\-\.]+@[\w\-\.]+\.\w{2,}$/';
if (preg_match($pattern, $email)) {
    echo "Valid email address";
} else {
    echo "Invalid email address";
}
```

9. Capturing Groups:

Use parentheses () to create capturing groups:

```php
$text = "John Doe";
$pattern = '/(\w+)\s(\w+)/';
preg_match($pattern, $text, $matches);
print_r($matches);
```

10. Modifiers:

```
Add modifiers after the closing delimiter:
- i: Case-insensitive matching
- m: Multi-line mode
- s: Dot matches newline
```

```php
$pattern = '/pattern/i'; // Case-insensitive
```

11. Matching Phone Numbers:

Let's create a pattern to match phone numbers in the format (XXX) XXX-XXXX.

```php
$phone_numbers = [
    "(123) 456-7890",
    "123-456-7890",
    "(800) 555-1234",
    "555-1234",
    "1234567890"
];

$pattern = '/^\(\d{3}\)\s\d{3}-\d{4}$/';

foreach ($phone_numbers as $number) {
    if (preg_match($pattern, $number)) {
        echo "$number is valid\n";
    } else {
        echo "$number is invalid\n";
    }
}
```

12. Extracting URLs from Text:

Let's extract all URLs from a given text.

```php
$text = "Visit https://www.example.com or http://subdomain.example.org. For more info, go to

$pattern = '/https?:\/\/[\w\-\.]+\.\w+(?:\/[\w\-\.\?=&]*)?/';

preg_match_all($pattern, $text, $matches);

echo "Found URLs:\n";
print_r($matches[0]);
```

13. Validating Passwords:

Create a pattern to validate passwords with at least 8 characters, containing at least one uppercase letter, one lowercase letter, one number, and one special character.

```php
$passwords = [
    "Weak123!",
    "StrongP@ssw0rd",
    "NoSpecialChar123",
    "short1!",
    "ALLUPPERCASEno123!",
    "NoNumbers!!!"
];

$pattern = '/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[!@#$%^&*]).{8,}$/';

foreach ($passwords as $password) {
    if (preg_match($pattern, $password)) {
        echo "$password is valid\n";
    } else {
        echo "$password is invalid\n";
    }
}
```

14. Replacing HTML Tags:

Let's replace all HTML tags with their content.

```php
$html = "<p>This is a <strong>bold</strong> and <em>emphasized</em> text.</p>";

$pattern = '/<[^>]+>(.*?)<\/[^>]+>/';

$stripped = preg_replace($pattern, '$1', $html);

echo "Original: $html\n";
echo "Stripped: $stripped\n";
```

15. Extracting Dates from Text:

Extract dates in the format MM/DD/YYYY from a given text.

```php
$text = "Important dates: 05/15/2023, 12/25/2023, and 01/01/2024.
         Invalid dates: 13/01/2023, 05/32/2023.";

$pattern = '/\b(0[1-9]|1[0-2])\/(0[1-9]|[12]\d|3[01])\/\d{4}\b/';

preg_match_all($pattern, $text, $matches);

echo "Found valid dates:\n";
print_r($matches[0]);
```

16. Splitting a String with Multiple Delimiters:

Split a string using multiple delimiters (comma, semicolon, or pipe).

```
$data = "apple,banana;cherry|date,elderberry|fig;grape";

$pattern = '/[,;|]/';

$fruits = preg_split($pattern, $data);

echo "Split result:\n";
print_r($fruits);
```

17. Matching and Replacing Email Domains:

Replace all email addresses from "example.com" domain with "newdomain.com".

```
$text = "Contact john@example.com or sarah@example.com for support.
         For sales, email sales@otherdomain.com.";

$pattern = '/(\w+@)example\.com\b/';
$replacement = '$1newdomain.com';

$new_text = preg_replace($pattern, $replacement, $text);

echo "Original: $text\n";
echo "Updated: $new_text\n";
```

These examples cover various common use cases for regular expressions in PHP. They demonstrate pattern matching, extraction, validation, and replacement techniques. You can run these examples to see how they work with the provided test data.