Server-Side Scripting using PHP

PHP

PHP is a popular general-purpose **scripting** language that is especially suited for web application development.

Fast, flexible and pragmatic, PHP powers everything from your blog to the most popular websites in the world.

PHP is a server-side scripting language

Scripting languages use a program known as an interpreter to translate commands and are **directly interpreted from source code**, **not requiring a compilation step**.

Other programming languages, on the other hand, may require a compiler to translate commands into machine code before it can execute those commands.

PHP

- PHP is an acronym for Hypertext Preprocessor.
- PHP is an open-source, interpreted, and object-oriented scripting language
- PHP is a widely-used **scripting language** that are executed on the server side.
- PHP borrowed its primary syntax from C++ & C.
- Many of the programming techniques you‟ve previously learned will work in PHP (assignments, comparisons, loops, etc.) with little to no syntax difference.
- There are, however, major changes in how data is manipulated in relationship to C/C++.
- C/C++ are type-specific languages, requiring the user to define a specific, singular type for each variable that they use.
- PHP commonly assigns its variables “by value”, meaning a variable takes on the value of the source variable (expression) and is assigned to the destination variable.
- A variable can therefore change its type “on the fly”.
- Therefore, variables are not declared as they are in most type-specific languages like C/C++.

Why PHP ?

- Generate pages and files dynamically.
- Create, open, read, write and close files on the server.
- Collect data from a web form such as user information, email, phone number, etc.
- Send emails to the users of your website.
- Send and receive cookies to track the visitor of your website.
- Store, delete, and modify information in your database.
- Restrict unauthorized access to your website.
- Encrypt data for safe transmission over internet.

Features

- Easy to learn – PHP is easy to learn and use. For beginner programmers who just started out in web development, PHP is often considered as the preferable choice of language to learn.
- Open source – PHP is an open-source project. It is developed and maintained by a worldwide community of developers who make its source code freely available to download and use.
- Portability – PHP runs on various platforms such as Microsoft Windows, Linus Mac OS, etc. and it is compatible with almost all servers used today such asApache, IIS, etc.
- Fast Performance – Scripts written in PHP usually execute or runs faster than those written in other scripting languages like ASP, Ruby, Python, Java, etc.
- Vast Community – Since PHP is supported by the worldwide community, finding help or documentation related to PHP online is extremely easy.

PHP can develop dynamic websites easily. But you must have the basic the knowledge of following technologies for web development as well.

HTML

CSS

JavaScript

Ajax

XML and JSON

jQuery

Prerequisite

Before learning PHP, you must have the basic knowledge of HTML, CSS, and JavaScript.

HTML - HTML is used to design static webpage.

CSS - CSS helps to make the webpage content more effective and attractive.

JavaScript - JavaScript is used to design an interactive website.

Syntax

PHP has quite easy syntax, if you are familiar with any C-type language.

- It has all the same structures that you are familiar with other programming languages.
- A PHP script can be placed anywhere in the document.
- It starts with <?php and ends with ?>

```
<?php
//PHP codes;
?>
```

- The default file extension for PHP files is “.php”.

Simple PHP Program

```
<html>
<head>
   <title>
         PHP
   </title>
</head>
<body>
<h1>First PHP Program</h1>
<?php
         echo "Hello World !";
?>
</body>
</html>
```

echo

PHP echo is a language construct, not a function. Therefore, you don't need to use parenthesis with it. But if you want to use more than one parameter, it is required to use parenthesis.

The syntax of PHP echo is given below:

void echo (string $arg1 [, string $...])

PHP echo statement can be used to print the string, multi-line strings, escaping characters, variable, array, etc. Some important points that you must know about the echo statement are:

echo is a statement, which is used to display the output.

echo can be used with or without parentheses: echo(), and echo.

echo does not return any value.

We can pass multiple strings separated by a comma (,) in echo.

echo is faster than the print statement.

Comments in PHP

 To write a single-line comment either start the line with two slashes (//) or a hash symbol (#).

 However to write multi-line comments, start the comment with a slash followed by an asterisk (/*) and end the comment with an asterisk followed by a slash (*/).

```
<?php
//This is a single line comment
#This is also a single line comment
/*
This is a multiple line comment block
that spans across more than
one line
*/
echo "Hello, PK!";
?>
```

PHP Variables

- In PHP, a variable **does not need to be declared** before adding a value to it.
- PHP automatically converts the variable to the correct **data type, depending on its value**.
- After declaring a variable it can be reused throughout the code.
- The assignment operator (=) is used to assign value to a variable.
- In PHP, variable can be declared as

  $var_name = value;

Naming Conventions

- All variables in PHP start with a $ sign, followed by the name of the variable.
- A variable name must start with a letter or the underscore character _.
- A variable name cannot start with a number.
- A variable name in PHP can only contain alpha-numeric characters and underscores (A-z, 0-9, and _).
- A variable name cannot contain spaces

PHP Case Sensitivity

In PHP, keyword (e.g., echo, if, else, while), functions, user-defined functions, classes are not case-sensitive. However, all variable names are case-sensitive.

Variable names in PHP are case-sensitive.

- As a result the variables $color, $Color, and $COLOR are treated as three different variables.

- However the keywords, functions and classes name are case-insensitive.

- As a result calling the gettype() or GETTYPE() produce the same result.

PHP Constant

- A constant is a name or an identifier for a fixed value.
- Constant are like variables, except that once they are defined, they cannot be undefined or changed.
- Constants are very useful for storing data that doesn‟t change while the script is running.
- Common examples of such data include configuration settings such as database username and password, website‟s base URL, company name, etc.
- Constants are defined using PHP‟s define() function, which accepts two arguments: the name of the constant, and its value.
- Naming conventions for PHP Constants are similar to that of PHP Variables.

Define Constant

PHP constants can be defined by 2 ways:

Using define() function
Using const keyword

define()

Use the define() function to create a constant. It defines constant at run time. Let's see the syntax of define() function in PHP.

define(name, value, case-insensitive)

name: It specifies the constant name.

value: It specifies the constant value.

case-insensitive: Specifies whether a constant is case-insensitive. Default value is false. It means it is case sensitive by default.

```
<?php
define("MESSAGE","Hello World");
echo MESSAGE;
?>
```

constant with case-insensitive:

```
<?php
define("MESSAGE","Hello World",true);//not case sensitive
echo MESSAGE, "</br>";
echo message;
?>
```

```
<?php
define("MESSAGE","Hello World",false);//case sensitive
echo MESSAGE;
echo message;
?>
```

PHP constant: const keyword

PHP has a keyword const to create a constant. The const keyword defines constants at compile time. It is a language construct, not a function. The constant defined using const keyword are case-sensitive.

```
<?php
const MESSAGE="Hello  World";
echo MESSAGE;
?>
```

language constructs is that language constructs are the most basic units of the language and cannot be broken down further by the PHP parser whereas functions have to be further broken down before being parsed, often into language constructs.

Data Types

PHP data types are used to hold different types of data or values. PHP supports eight primitive data types that can be categorized in following types:

Scalar Types (predefined)

Compound Types (user-defined)

Special Types

Scalar Types: It holds only single value. There are 4 scalar data types in PHP.

boolean, integer, float , string

Compound Types: It can hold multiple values. There are 2 compound data types in PHP.

Array, and object

Special Types: There are 2 special data types in PHP.

resource

NULL

Operators

Operators are symbols that tell the PHP processor to perform certain actions. For example, the addition (+) symbol is an operator that tells PHP to add two variables or values, while the greater-than (>) symbol is an operator that tells PHP to compare two values. In simple words, operators are used to perform operations on variables or values. For example:

$sum=10+20;//+ is the operator and 10,20 are operands

In the above example, + is the binary + operator, 10 and 20 are operands and $sum is variable.

PHP Operators can be categorized in following forms:

Arithmetic Operators
Assignment Operators
Bitwise Operators
Comparison Operators
Incrementing/Decrementing Operators
Logical Operators
String Operators
Array Operators
Type Operators
Execution Operators
Error Control Operators

Arithmetic Operators

The arithmetic operators are used to perform common arithmetical operations, such as addition, subtraction, multiplication etc. Here's a complete list of PHP's arithmetic operators:

Operator	Description	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y.
*	Multiplication	$x * $y	Product of $x and $y.
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y

```php
<?php
$x = 30; // Variable 1
$y = 5; // Variable 2

// Some arithmetic operations on
// these two variables
echo ($x + $y), "\n";
echo($x - $y), "\n";
echo($x * $y), "\n";
echo($x / $y), "\n";
echo($x % $y), "\n";
?>
```

Assignment Operators

The assignment operators are used to assign value to different variables. The basic assignment operator is "=".

Operator	Name	Example	Explanation
=	Assign	$a = $b	The value of right operand is assigned to the left operand.
+=	Add then Assign	$a += $b	Addition same as $a = $a + $b
-=	Subtract then Assign	$a -= $b	Subtraction same as $a = $a - $b
*=	Multiply then Assign	$a *= $b	Multiplication same as $a = $a * $b
/=	Divide then Assign (quotient)	$a /= $b	Find quotient same as $a = $a / $b
%=	Divide then Assign (remainder)	$a %= $b	Find remainder same as $a = $a % $b

```php
<?php
// Simple assign operator
$y = 75;
echo $y, "\n";

// Add then assign operator
$y = 100;
$y += 200;
echo $y, "\n";

// Subtract then assign operator
$y = 70;
$y -= 10;
echo $y, "\n";

// Multiply then assign operator
$y = 30;
$y *= 20;
echo $y, "\n";

// Divide then assign(quotient) operator
$y = 100;
$y /= 5;
echo $y, "\n";

// Divide then assign(remainder) operator
$y = 50;
$y %= 5;
echo $y;
?>
```

Logical or Relational Operators:

These are basically used to operate with conditional statements and expressions. Conditional statements are based on conditions. Also, a condition can either be met or cannot be met so the result of a conditional statement can either be true or false. Here are the logical operators along with their syntax and operations in PHP.

Operator	Name	Example	Explanation
and	And	$a and $b	Return TRUE if both $a and $b are true
Or	Or	$a or $b	Return TRUE if either $a or $b is true
xor	Xor	$a xor $b	Return TRUE if either $ or $b is true but not both
!	Not	! $a	Return TRUE if $a is not true
&&	And	$a && $b	Return TRUE if either $a and $b are true
||	Or	$a || $b	Return TRUE if either $a or $b is true

```php
<?php
$x = 50;
$y = 30;
if ($x == 50 and $y == 30)
    echo "and Success \n";

if ($x == 50 or $y == 20)
    echo "or Success \n";

if ($x == 50 xor $y == 20)
    echo "xor Success \n";

if ($x == 50 && $y == 30)
    echo "&& Success \n";

if ($x == 50 || $y == 20)
    echo "|| Success \n";

if (!$z)
    echo "! Success \n";
?>
```

Comparison Operators:

These operators are used to compare two elements and outputs the result in boolean form. Here are the comparison operators along with their syntax and operations in PHP.

```
<?php
$a = 80;
$b = 50;
$c = "80";

// Here var_dump function has been used to
// display structured information. We will learn
// about this function in complete details in further
// articles.
var_dump($a == $c) + "\n";
var_dump($a != $b) + "\n";
var_dump($a <> $b) + "\n";
var_dump($a === $c) + "\n";
var_dump($a !== $c) + "\n";
var_dump($a < $b) + "\n";
var_dump($a > $b) + "\n";
var_dump($a <= $b) + "\n";
var_dump($a >= $b);
?>
```

Operator	Name	Example	Explanation
==	Equal	$a == $b	Return TRUE if $a is equal to $b
===	Identical	$a === $b	Return TRUE if $a is equal to $b, and they are of same data type
!==	Not identical	$a !== $b	Return TRUE if $a is not equal to $b, and they are not of same data type
!=	Not equal	$a != $b	Return TRUE if $a is not equal to $b
<>	Not equal	$a <> $b	Return TRUE if $a is not equal to $b
<	Less than	$a < $b	Return TRUE if $a is less than $b
>	Greater than	$a > $b	Return TRUE if $a is greater than $b
<=	Less than or equal to	$a <= $b	Return TRUE if $a is less than or equal $b
>=	Greater than or equal to	$a >= $b	Return TRUE if $a is greater than or equal $b
<=>	Spaceship	$a <=>$b	Return -1 if $a is less than $b Return 0 if $a is equal $b Return 1 if $a is greater than $b

Increment/Decrement Operators

These are called the unary operators as they work on single operands. These are used to increment or decrement values.

```
<?php
$x = 2;
echo ++$x, " First increments then prints \n";
echo $x, "\n";

$x = 2;
echo $x++, " First prints then increments \n";
echo $x, "\n";

$x = 2;
echo --$x, " First decrements then prints \n";
echo $x, "\n";

$x = 2;
echo $x--, " First prints then decrements \n";
echo $x;
?>
```

Operator	Name	Example	Explanation
++	Increment	++$a	Increment the value of $a by one, then return $a
		$a++	Return $a, then increment the value of $a by one
--	decrement	--$a	Decrement the value of $a by one, then return $a
		$a--	Return $a, then decrement the value of $a by one

Strings

string is a sequence of characters i.e., used to store and manipulate text. PHP supports only 256-character set and so that it does not offer native Unicode support. There are 4 ways to specify a string literal in PHP.

single quoted

double quoted

heredoc syntax

newdoc syntax

Single Quoted

You can create a string in PHP by enclosing the text in a single-quote. It is the easiest way to specify string in PHP.

For specifying a literal single quote, escape it with a backslash (\) and to specify a literal backslash (\) use double backslash (\\). All the other instances with backslash such as \r or \n, will be output same as they specified instead of having any special meaning.

```
<?php
$str1='Hello World
This is a line with text
Another line within single quoted string';
$str2='Using double "quote" directly inside single quoted string';
$str3='Using escape sequences \n in single quoted string';
echo "$str1 <br/> $str2 <br/> $str3";
?>
```

Double Quoted

In PHP, we can specify string through enclosing text within double quote also. But escape sequences and variables will be interpreted using double quote PHP strings.

```
<?php
$str1="Using double "quote" directly inside double quoted string";
echo $str1;
?>
```

```
<?php
$str1="Hello text
multiple line
text within double quoted string";
$str2="Using double \"quote\" with backslash inside double quoted string";
$str3="Using escape sequences \n in double quoted string";
echo "$str1 <br/> $str2 <br/> $str3";
?>
```

String functions

The strtolower() function returns string in lowercase letter.

```
<?php
$str="Hello PK";
$str=strtolower($str);
echo $str;
?>
```

The strtoupper() function returns string in uppercase letter.

```
$str=strtoupper($str);
echo $str;
```

The ucfirst() function returns string converting first character into uppercase. It doesn't change the case of other characters.

```
$str=ucfirst($str);
```

The strlen() function returns length of the string.

```
$len=strlen($str);
echo $len;
```

Counting Number of Words in a String str_word_count()

```
<?php
$my_str = 'The quick brown fox jumps over the lazy dog.';
echo str_word_count($my_str);
?>
```

Replacing Text within Strings str_replace()

```
<?php
$my_str = 'If the facts do not fit the theory, change the
    facts.';
echo str_replace("facts", "truth", $my_str, $count);
echo "<br>";
echo "The text was replaced $count times.";
?>
```

Reversing a String – strrev()

```
<?php
$my_str = 'You can do anything, but not everything.';
echo strrev($my_str);
?>
```

Conditional Statement

PHP if else statement is used to test condition. There are various ways to use if statement in PHP.

- if
- if-else
- if-else-if
- nested if

```php
<?php
$num=12;
if($num<100){
echo "$num is less than 100";
}
?>
```

```php
<?php
$num=12;
if($num%2==0){
echo "$num is even number";
}else{
echo "$num is odd number";
}
?>
```

File Handling

File handling simply means to open a file and to process it according to the required tasks. PHP facilitates several functions to create, read, write, append, delete and close files.

PHP supports the following file formats for reading and write operations.

Text Files: Files with extension .txt

Log Files: Files with extension .log

Custom Extensions: Files with a custom extension like .xyz

CSV Files: Files with extension .csv

Image Files: Files with extension .jpg/png/gif

File with Initialization Setting: Files with extension .ini

PHP File System allows us to create file, read file line by line, read file character by character, write file, append file, delete file and close file.

File Handling in PHP – Opening

- The PHP fopen() function is used to open a file.
- The basic syntax of this function is:

fopen(filename, mode)

- The first parameter passed to fopen() specifies the name of the file you want to open and the second parameter specifies which mode the file should be opened.

For example:

```
<?php
$handle = fopen("data.txt", "r");
echo "File opened successfully.";
?>
```

File Handling in PHP – Opening

- If you try to open a file that doesn‟t exist, PHP will generate a warning message.
- To avoid these error messages you should always implement a simple check whether a file or directory exists or not before trying to access it, with the PHP file_exists() function.

```
<?php
$file = "data.txt";
// Check the existence of file
if(file_exists($file)){
echo "The file $file exists." . "<br>";
// Attempt to open the file
$handle = fopen($file, "r") or die("ERROR: Cannot open the file.");
if($handle){
echo "File opened successfully.";
// Closing the file handle
fclose($handle);
}
} else{
echo "ERROR: The file $file does not exist.";
}
?>
```

File Handling in PHP – Opening

The file may be opened in one of the following modes:

r

Open the file for reading only

r+

Open the file for reading and writing

w

Open the file for writing only and clears the contents of file. If the file doesn‟t exist, PHP will attempt to create it.

w+

Open the file for reading and writing and clears the contents of file. If the file doesn‟t exist, PHP will attempt to create it.

a

Append. Opens the file for writing only. Preserves file content by writing to the end of the file. If the file doesn‟t exist, PHP will attempt to create it.

a+

Read/Append. Opens the file for reading and writing. Preserves file content by writing to the end of the file. If the file doesn‟t exist, PHP will attempt to create it.

x

Opens the file for writing only. Return FALSE and generates an error if the file already exists. If the file doesn‟t exist, PHP will attempt to create it.

x+

Open the file for reading and writing; otherwise it has the same behavior as „x‟.

File Handling in PHP – Closing

- Once you‟ve finished working with a file, it needs to be closed.
- The fclose() function is used to close the file, as seen in the previous example.
- Although PHP automatically closes all open files when script terminates, but it‟s a good practice to close a file after performing all the operations

File Handling in PHP – Reading

- The fread() function can be used to read a specified number of characters from a file.
- The basic syntax of this function can be given as:

fread(file handle, length in bytes)

- This function takes two parameter – a file handle and the number of bytes to read.
- The following example reads 20 bytes from the “data.txt” file including spaces.

fread($handle, "20");

- The fread() function can be used in conjugation with the filesize() function to read the entire file at once.
- The fiesize() function returns the size of the file in bytes.

fread($handle, filesize($file));

File Handling in PHP – Reading

- The easiest way to read the entire contents of a file in PHP is with the readfile() function.
- This function allows you to read the contents of a file without needing to open it.

readfile($file)

- Another way to read the whole content of a file without needing to open it is with the file_get_contents() function.

file_get_contents($file)

File Handling in PHP – Reading

```
<?php
$file = "data.txt";
// Check the existence of file
if(file_exists($file)){
// Open the file for reading
$handle = fopen($file, "r");
// Reading the entire file
$content = fread($handle, filesize($file));
// Display the file content
echo $content;
// Closing the file handle
fclose($handle);
} else{
echo "ERROR: File does not exist.";
}
?>
```

readfile()

The readfile() function in PHP is an inbuilt function which is used to read a file and write it to the output buffer. The filename is sent as a parameter to the readfile() function and it returns the number of bytes read on success, or FALSE and an error on failure.

By adding an '@' in front of the function name the error output can be hidden.

```
$myfile = @readfile("gfg.txt");
```

```
<?php
$file = "data.txt";
// Check the existence of file
if(file_exists($file)){
// Reads and outputs the entire file
readfile($file);
} else{
echo "ERROR: File does not exist.";
}
?>
```

File Handling in PHP – Writing

- The fwrite() function can be used to write data to a file or append to an existing file using PHP.
- The basic syntax of this function can be given as:

fwrite(file handle, string)

- This function takes two parameter – a file handle and the string of data that is to be written.
- The following example writes the message to the “data.txt” file.

fwrite($handle, "This line will be written in the file");

- An alternative way is using the file_put_contents() function which is the counterpart of file_get_contents() function and provides an easy method of writing the data to a file without needing to open it.

File Handling in PHP – Renaming

The rename() function can be used to rename a file or directory using PHP.

```
<?php
$file = "data.txt";
// Check the existence of file
if(file_exists($file)){
// Attempt to rename the file
if(rename($file, "newfile.txt")){
echo "File renamed successfully.";
} else{
echo "ERROR: File cannot be renamed.";
}
} else{
echo "ERROR: File does not exist.";
}
?>
```

File Handling in PHP – Deleting

The unlink() function can be used to delete a file or directory using PHP.

```
<?php
$file = "newfile.txt";
// Check the existence of file
if(file_exists($file)){
// Attempt to delete the file
if(unlink($file)){
echo "File removed successfully.";
} else{
echo "ERROR: File cannot be removed.";
}
} else{
echo "ERROR: File does not exist.";
}
?>
```

Form Handling in PHP

```
<html>
<head><title>Contact Form</title> </head>
<body>
    <h2>Contact Us</h2>
    <p>Please fill in this form and send us.</p>
      <form action="process-form.php" method="post">
          <p>
              <label for="inputName">Name:<sup>*</sup></label>
              <input type="text" name="name" id="inputName">
          </p>
          <p>
              <label for="inputEmail">Email:<sup>*</sup></label>
              <input type="text" name="email" id="inputEmail">
          </p>
          <p>
          <label for="inputSubject">Subject:</label>
          <input type="text" name="subject" id="inputSubject">
          </p>
          <p>
          <label for="inputComment">Message:<sup>*</sup></label>
          <textarea name="message" id="inputComment" rows="5" cols="30"></textarea>
          </p>
        <input type="submit" value="Submit">
         <input type="reset" value="Reset">
      </form>
</body>
```

Form Handling in PHP

```
<html>
<head>
<title>Contact Form</title>
</head>
<body>
<h1>Thank You</h1>
<p>Here is the information you have submitted:</p>
<ol>
<li><em>Name:</em> <?php echo $_POST["name"]?></li>
<li><em>Email:</em> <?php echo $_POST["email"]?></li>
<li><em>Subject:</em> <?php echo $_POST["subject"]?></li>
<li><em>Message:</em> <?php echo $_POST["message"]?></li>
</ol>
</body>
</html>
```

Cookies

- A cookie is a small text file that lets you store a small amount of data (nearly 4KB) on the user‟s computer.
- They are typically used to keeping track of information such as username that the site can retrieve to personalize the page when user visits the website next time.
- Each time the browser requests a page to the server, all the data in the cookie is automatically sent to the server within the request.

Setting a Cookie in PHP

- The setcookie() function is used to set a cookie in PHP.
- Make sure you call the setcookie() function before any output generated by your script otherwise cookie will not set.
- The basic syntax of this function is as follow:

setcookie(name, value, expire, path, domain, secure);

- If the expiration time of the cookie is set to 0, or omitted, the cookie will expire at the end of the session. i.e. when the browser closes.

```
<?php
// Setting a cookie
setcookie("username", "John Nash", time()+30*24*60*60);
echo “Cookie has been set.”;
?>
```

Accessing Cookies Values in PHP

- The PHP $_COOKIE superglobal variable is used to retrieve a cookie value.
- The individual cookie value can be accessed using standard array notation.

```
<?php
// Accessing an individual cookie value
echo $_COOKIE["username"];
?>
```

- An example for accessing cookie:

```
<?php
// Verifying whether a cookie is set or not
if(isset($_COOKIE["username"])){
echo "Hi " . $_COOKIE["username"];
} else{
echo "Welcome Guest!";
}
?>
```

Removing Cookies in PHP

- You can delete a cookie by calling the same setcookie() function with the cookie name and any value (such as an empty string).
- However this time you need to set the expiration date in the past.

```
<?php
// Deleting a cookie
setcookie("username", "", time()-3600);
?>
```

- You should pass exactly the same path, domain, and other arguments that you have used when you first created the cookie in order to ensure that the correct cookie is deleted.