B.C.A. Semester – VI
US06CBCA21
Web Programming using PHP

Unit-1

1. Introduction to PHP
2. History of PHP,
3. Features
4. Merits and Demerits of PHP,
5. General structure of PHP,
6. Displaying Output,
7. Escaping Special Characters,
8. Comments,
9. Variables – (Declaring, Assigning, Destroying),
10. Datatypes, Setting and Testing Datatypes –
11. Constants –
12. Operators (Arithmetic, Comparison, Logical, Assignment, Concatenation) –
13. Superglobal variables

1.0 Introduction to PHP

1.1 What is PHP?

1. PHP is an acronym for "PHP: Hypertext Preprocessor"
2. PHP is a widely-used, open source scripting language
3. PHP scripts are executed on the server
4. PHP is free to download and use

1.2 What is a PHP File?

1. PHP files can contain text, HTML, CSS, JavaScript, and PHP code
2. PHP code is executed on the server, and the result is returned to the browser as plain HTML
3. PHP files have extension "xxx.php"

1.3 What Can PHP Do?

1. PHP can generate dynamic page content
2. PHP can create, open, read, write, delete, and close files on the server
3. PHP can collect form data
4. PHP can send and receive cookies
5. PHP can add, delete, modify data in your database
6. PHP can be used to control user-access
7. PHP can encrypt data

With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

1.4 Why PHP?

1. PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
2. PHP is compatible with almost all servers used today (Apache, IIS, etc.)

3. PHP supports a wide range of databases
4. PHP is free.
5. PHP is easy to learn and runs efficiently on the server side

2.0 History of PHP

1. PHP 1 [PHP: Hypertext Preprocessor] was created by RasmusLerdorf in 1994. It was initially developed for HTTP usage logging and server-side form generation in Unix.

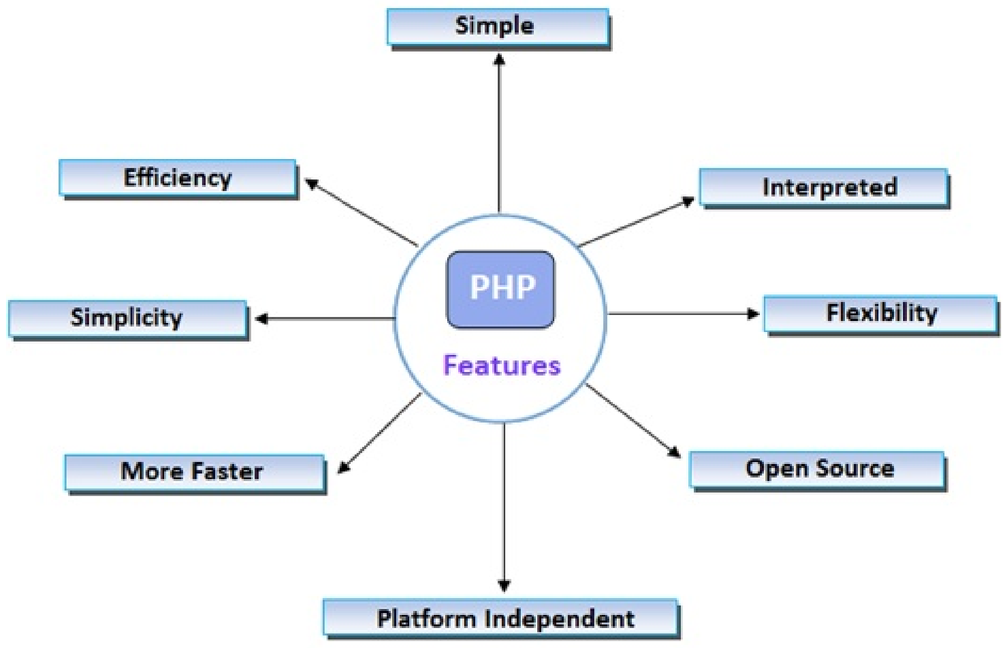
2. PHP 2 (1995) transformed the language into a Server-side embedded scripting language. Added database support, file uploads, variables, arrays, recursive functions, conditionals, iteration, regular expressions, etc.

3. PHP 3 [1998] added support for ODBC data sources, multiple platform support, email protocols, and new parser written by ZeevSuraski .

4. PHP 4 (2000) became an independent component of the web server for added efficiency. The parser was renamed the Zend Engine. Many security features were added.

5. PHP 5 (2004) adds Zend Engine II with object oriented programming robust XML support using the libxrnl2 library, SQLite has been bundled with PHP.

3.0 Features of PHP



Source: http://www.tracesoftware.in/phps-main-features/

It is most popular and frequently used worldwide scripting language, the main reason of popularity is; It is open source and very simple.

1. Simple and Faster
2. Interpreted and Open Source
3. Case Sensitive and Simplicity
4. Efficiency

5. Platform Independent
6. Security, Familiarity and Flexibility

(1) Simple, Familiar and ease of use: Its popularly known for its simplicity, familiarity and easy to learn the language as the syntax is similar to that of 'C' or Pascal language. So the language is very logical and well organized general-purpose programming language. Even people with a normal programming background can easily understand and capture the use of language. PHP is very advantageous for new users as it's a very reliable, fluent, organized, clean, demandable and efficient.
The main strength of PHP is the availability of rich pre-defined functions. The core distribution helps the developers implement dynamic websites very easily with secured data. PHP applications are very easy to optimize.

(2) Flexibility: PHP is known for its flexibility and embedded nature as it can be well integrated with **HTML**, **XML**, **JavaScript** and many more. PHP can run on multiple operating systems like **Windows**, **Unix**, **Mac OS**, **Linux**, **etc**. The PHP scripts can easily run on any device like laptops, mobiles, tablets, and computer. It is very comfortably integrated with various Databases. Desktop applications are created using advanced PHP features. The executable PHP can also be run on command-line as well as directly on the machine. Heavyweight applications can be created without a server or browser.It also acts as an excellent interface with relational databases.

(3) Open Source: All PHP frameworks are open sources, no payment is required for the users and its completely free. User can just download PHP and start using for their applications or projects. Even in companies, the total cost is reduced for software development providing morereliability and flexibility.It supports a popular range of databases like MySQL, SQLite, Oracle, Sybase,Informix, and PostgreSQL.

(4) PHP provides libraries to access these databases to interact with web servers. Developers are free to post errors, inspect codes and can contribute to code as well as bug fixing. Many frameworks like Codeignitor, Zend Framework, CakePHP make use of PHP.Even many popular content management systems like WordPress, Joomla and Drupal use PHP as prime language.
Because of the above reasons many web hosting companies and Internet Service providers prefers PHP.

(5) Cross-platform compatibility: PHP is multi-platform and known for its portability as it can run on any operating System and windows environments. The most common are XAMPP (**Windows**, **Apache Server**, **MySQL**, **Perl**, and **PHP**) and LAMP (**Linux**, **Apache**, **MySQL**, **PHP**). As PHP is platform-independent, it's very easy to integrate with various databases and other technologies without re-implementation. It effectively saves a lot of energy, time and money.

(6) Fast and efficient performance: Users generally prefer fast loading websites. For any web development, speed becomes an important aspect which is taken care of by PHP.

(7) PHP scripts are faster than other scripting languages like **ASP.NET**, **PERL**, and **JSP**. The memory manager of PHP 7 is very optimized and fast as compared to older versions of PHP. Even connecting to the database and loading of required data from tables, are faster than other programming languages. It provides a built-in module for easy and efficient database management system. The high speed of PHP is advantageous for users for its server administration and mail functionality. Also, it supports session management and removing of unwanted memory allocation.

Some More Features

(8) **Error reporting and exceptions:** PHP supports much errors reporting constants to generate errors and relevant warnings at run time.
For,example **E_ERROR, E_WARNING, E_PARSE, E_STRICT**.

(9) **Active community support:** PHP is very rich with many diverse online community developers to help beginners for web-based applications. These worldwide volunteers contribute many features as well as new versions for PHP libraries. Even they contribute a translation in different languages to help out programmers. There is a bundle of third-party open-source libraries which provide basic functionalities. Even the documentation given by the official site helps in implementing new features providing access to a variety of creative imagination.

(10) **Maintenance:** When dealing with big projects, maintenance of code is also an important aspect of the web development process. There are many PHP frameworks for example MVC (Model View Controller) which makes development and maintenance of code easier. Files belonging to the different module are maintained separately.

(11) **Third-party application support and security:** Many PHP's predefined functions support data encryption options keeping it more secure. Even the users can use third-party applications to secure data.

(12) **Real time access monitoring:** PHP also provides a summary of user's recent logging accesses.

(13) **Memory and CPU usage information:** PHP can provide memory usage information from functions like **memory_get_usage()** or **memory_get_peak_usage()**, which can help the developers optimize their code. In the similar way, the CPU power consumed by any script can be retrieved for further optimization.

(14) **Object oriented features:** PHP supports object-oriented programming features, resulting in increased speed and introducing added features like data encapsulation and inheritance at many levels.

(15) **Loosely typed language:** PHP encourages the use of variables without declaring its data types. So this is taken care at the execution time depending on the value assigned to the variable. Even the variable name can be changed dynamically.

4.0 Merits and Demerits of PHP

Merits

1. PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
2. PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
3. It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle,

Sybase, Informix, and Microsoft SQL Server.
4. PHP is pleasingly(satisfying) zippy(fresh) in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.
5. PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.
6. PHP is forgiving(understanding): PHP language tries to be as forgiving as possible. PHP Syntax is C-Like.

Demerits

1. **Security:** Since it is open sourced, all people can see the source code. If there are bugs in the source code, it can be used by people to explore the weakness of it.
2. **Not suitable of large applications:** It will be difficult to use it for programming huge applications. Since the programming language is not highly modular, huge applications created out of the programming language will be difficult to maintain.
3. **Weak type:** Implicit conversion may surprise unwary programmers and lead to unexpected bugs. Confusion between arrays and hash tables. This is slow and could be faster. There are often a few ways to accomplish a task. It is not strongly typed. It is interpreted and uses curly braces.
4. **Poor Error Handling Method:** The framework has a bad error handling method. It is not a proper solution for the developers. Therefore, as a qualified PHP developer, you will have to overcome it.
5. **PHP is unable to handle large number of apps:** The technology is helpless to support a bunch of apps. It is highly tough to manage because, it is not competent modular. It already imitates the features of Java language.

4.1 Applications of PHP

1. PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
2. PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
3. You add, delete, modify elements within your database through PHP.
4. Access cookies variables and set cookies.
5. Using PHP, you can restrict users to access some pages of your website.
6. It can encrypt data.

5.0 General structure of PHP

PHP code is always separated from HTML code with symbols
<?php (LESS-THAN sign followed by question mark and the word php) and
?> (a question mark followed by a GREATER-THAN sign).

<?phpindicates the start of a PHP block or code and tells the web server thatall statements that follow until **?>** are PHP statements.

```
<?php

//<some valid PHP syntax>
```

```
?>
```

6.0 Displaying Output

PHP echo and print Statements

echo and print are more or less the same. They are both used to output data to the screen.

The differences are small: echo has no return value while print has a return value of 1 so it can be used in expressions. echo can take multiple parameters (although such usage is rare) while print can take one argument. echo is marginally faster than print

PHP echo Statement

The echo statement can be used with or without parentheses: echo or echo().

Display Text

The following example shows how to output text with the echo command.

```
<?php
echo "<h2>PHP is Fun!</h2>";
echo "Hello world!<br>";
echo "I'm about to learn PHP!<br>";
echo "This ", "string ", "was ", "made ", "with multiple parameters.";
?>
```

PHP print Statement

The print statement can be used with or without parentheses: print or print().

Display Text

The following example shows how to output text with the print command.

```
<?php
print "<h2>PHP is Fun!</h2>";
print "Hello world!<br>";
print "I'm about to learn PHP!";
?>
```

7.0 Escaping Special Characters

Escaping to PHP:

The PHP parsing engine needs a way to differentiate PHP code from other elements in the page. The mechanism for doing so is known as 'escaping to PHP'.

1. **Canonical PHP tags: <?php … ?>**
   If you use this style, you can be positive that your tags will always be correctly interpreted.

2. **Short-open (SGML-style) tags: <? … ?>**
   Short tags are, as one might expect, the shortest option You must do one of two things to enable PHP to recognize the tags −
   Choose the --enable-short-tags configuration option when you're building PHP.
   Set the short_open_tag setting in your php.ini file to on. This option must be disabled to parse XML with PHP because the same syntax is used for XML tags.

3. **ASP-style tags: <% … %>**
   ASP-style tags mimic(copy) the tags used by Active Server Pages to delineate(describe) code blocks.

4. **HTML script tags: <script language= “PHP”> …. </script>**

The escape-sequence replacements are −

\n is replaced by the newline character

\r is replaced by the carriage-return character

\t is replaced by the tab character

\$ is replaced by the dollar sign itself ($)

\" is replaced by a single double-quote (")

\\ is replaced by a single backslash (\)

8.0 Comments

A comment is the portion of a program that exists only for the human reader and stripped out(take away from) before displaying the programs result. There are two commenting formats in PHP −

1. Single Line comments:
They are generally used for short explanations or notes relevant to the local code. Here are theexamples of single line comments.

```
<?
  # This is a comment, and
  # This is the second line of the comment

  // This is a comment too. Each style comments only
  print "An example with single line comments";
?>
```

2. Multi-lines comments:
They are generally used to provide pseudocode algorithms and more detailed explanations when necessary. The multiline style of commenting is the same as in C. Here is the example of multi lines comments.

```
<?
  /* This is a comment with multiline
Author:Ivan Bayross
    Purpose: Multiline Comments Demo
    Subject: PHP
  */
    print "An example with multi line comments";
?>
```

9.0 Variables

The main way to store information in the middle of a PHP program is by using a variable.

Here are the most important things to know about variables in PHP.

1. All variables in PHP are denoted with a leading dollar sign ($).
2. The value of a variable is the value of its most recent assignment.
3. Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.
4. Variables can, but do not need, to be declared before assignment.
5. Variables in PHP do not have intrinsic types - a variable does not know in advance whether it will be used to store a number or a string of characters.
6. Variables used before they are assigned have default values.
7. PHP does a good job of automatically converting types from one to another when necessary.
8. PHP variables are Perl-like.

Variable Scope:

Scope can be defined as the range of availability a variable has to the program in

which it is declared. PHP variables can be one of four scope types −

1. Local Variables

2. Function Parameters

3. Global Variables

4. Static Variables

Rules for PHP variables:

1. A variable start with the $ sign, followed by the name of the variable.
2. A variable name must start with a letter or the underscore character.
3. A variable name cannot start with a number.
4. A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _).
5. Variable names are case-sensitive ($age and $AGE are two different variables)

Declaring and Assign PHP Variables

In PHP, a variable starts with the $ sign, followed by the name of the variable:

```
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

Destroy PHP Variables

Unset () Description

unset (mixed $var , mixed ...$vars) : void

unset () destroys the specified variables. The behavior of unset () inside of a function can vary depending on what type of variable you are attempting to destroy. If a globalized variable is unset () inside of a function, only the local variable is destroyed. The variable in the calling environment will retain the same value as before unset () was called.

```
<?php
function destroy_foo()
{  global $foo;
   unset($foo); }
$foo = 'bar';
destroy_foo();
echo $foo;
?>
```

10.0 Data Types

PHP supports the following data types:

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

PHP String

A string is a sequence of characters, like "Hello world!".

A string can be any text inside quotes. You can use single or double quotes:

```
<?php
$x = "Hello world!";
$y = 'Hello world!';

echo $x;
echo "<br>";
echo $y;
?>
```

PHP Integer

An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.

Rules for integers:

- An integer must have at least one digit
- An integer must not have a decimal point
- An integer can be either positive or negative
- Integers can be specified in: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation

In the following example $x is an integer. The PHP var_dump() function returns the data type and value:

```
<?php
$x = 5985;
var_dump($x);
?>
```

PHP Float

A float (floating point number) is a number with a decimal point or a number in exponential form.

In the following example $x is a float. The PHP var_dump() function returns the data type and value:

```
<?php
$x = 10.365;
var_dump($x);
?>
```

PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

```
$x = true;
$y = false;
```

Booleans are often used in conditional testing. You will learn more about conditional testing in a later chapter of this tutorial.

PHP Array

An array stores multiple values in one single variable.

In the following example $cars is an array. The PHP var_dump() function returns the data type and value:

```
<?php
$cars = array("Volvo","BMW","Toyota");
var_dump($cars);
?>
```

PHP Object

Classes and objects are the two main aspects of object-oriented programming.

A class is a template for objects, and an object is an instance of a class.

When the individual objects are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

Let's assume we have a class named Car. A Car can have properties like model, color, etc. We can define variables like $model, $color, and so on, to hold the values of these properties.

When the individual objects (Volvo, BMW, Toyota, etc.) are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

If you create a __construct() function, PHP will automatically call this function when you create an object from a class.

```
<?php
class Car {
  public $color;
  public $model;
  public function __construct($color, $model) {
    $this->color = $color;
    $this->model = $model;
  }
  public function message() {
    return "My car is a " . $this->color . " " . $this->model . "!";
  }
}

$myCar = new Car("black", "Volvo");
echo $myCar -> message();
echo "<br>";
$myCar = new Car("red", "Toyota");
echo $myCar -> message();
?>
```

PHP NULL Value

Null is a special data type which can have only one value: NULL.

A variable of data type NULL is a variable that has no value assigned to it.

Variables can also be emptied by setting the value to NULL:

```
<?php
$x = "Hello world!";
$x = null;
var_dump($x);
?>
```

PHP Resource

The special resource type is not an actual data type. It is the storing of a reference to functions and resources external to PHP.

A common example of using the resource data type is a database call.

We will not talk about the resource type here, since it is an advanced topic.

11.0 PHP Constants

Constants are like variables except that once they are defined they cannot be changed or undefined.
A constant is an identifier (name) for a simple value. The value cannot be changed during the script.
A valid constant name starts with a letter or underscore (no $ sign before the constant name).

Create a PHP Constant

To create a constant, use the define() function.

Syntax

define (*name*, *value*, *case-insensitive*)

Parameters:

- *name*: Specifies the name of the constant
- *value*: Specifies the value of the constant
- *case-insensitive*: Specifies whether the constant name should be case-insensitive. Default is false

```
<?php
```

```
define("GREETING", "Welcome to W3Schools.com!");
echo GREETING;
?>
<?php
define("GREETING", "Welcome to W3Schools.com!", true);
echo greeting;
?>
```

PHP Constant Arrays

```
<?php
define("cars", [
  "Alfa Romeo",
  "BMW",
  "Toyota"
]);
echo cars[0];
?>
```

Constants are Global

Constants are automatically global and can be used across the entire script.
This example uses a constant inside a function, even if it is defined outside the function:

```
<?php
define("GREETING", "Welcome to W3Schools.com!");
function myTest() {
  echo GREETING;
}
myTest();
?>
```

12.0 Operators

Operators are used to perform operations on variables and values.

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Array operators
- Conditional assignment operators

PHP Arithmetic Operators

The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

Operator	Name	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
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power

PHP Assignment Operators

The PHP assignment operators are used with numeric values to write a value to a variable.

The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

Assignment	**Same as...**	**Description**
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

PHP Comparison Operators

The PHP comparison operators are used to compare two values (number or string):

Operator	**Name**	**Example**	**Result**
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y

>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y
<=>	Spaceship	$x <=> $y	Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7.

PHP Increment / Decrement Operators

The PHP increment operators are used to increment a variable's value.

The PHP decrement operators are used to decrement a variable's value.

Operator	**Name**	**Description**
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

PHP Logical Operators

The PHP logical operators are used to combine conditional statements.

Operator	**Name**	**Example**	**Result**
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

PHP String Operators

PHP has two operators that are specially designed for strings.

Operator	**Name**	**Example**	**Result**

.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1

PHP Array Operators

The PHP array operators are used to compare arrays.

Operator	**Name**	**Example**	**Result**
+	Union	$x + $y	Union of $x and $y
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types
!=	Inequality	$x != $y	Returns true if $x is not equal to $y
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y

PHP Conditional Assignment Operators

The PHP conditional assignment operators are used to set a value depending on conditions:

Operator	**Name**	**Example**	**Result**
?:	Ternary	$x = *expr1* ? *expr2* : *expr3*	Returns the value of $x. The value of $x is *expr2* if *expr1* = TRUE. The value of $x is *expr3* if *expr1* = FALSE
??	Null coalescing	$x = *expr1* ?? *expr2*	Returns the value of $x. The value of $x is *expr1* if *expr1* exists, and is not NULL. If *expr1* does not exist, or is NULL, the value of $x is *expr2*. Introduced in PHP 7

PHP Concatenation Operators

PHP Compensation Operator is used to combine character strings.

Operator	Description
.	The PHP concatenation operator (.) is used to combine two string values to create one string.
.=	Concatenation assignment.

```
<?php
   $name="Karamchand";
   $lastName="Gandhi";
   echo $name." ".$lastName; // Outputs Karamchand Gandhi
   $a="Hello";
   $a .= " Gandhi!";
   echo $a; // Outputs Hello Gandhi!
?>
```

13.0 Global Variables - Superglobals

Some predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

The PHP superglobal variables are:

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET

(1) PHP $GLOBALS

$GLOBALS is a PHP super global variable which is used to access global variables from anywhere in the PHP script (also from within functions or methods).

PHP stores all global variables in an array called $GLOBALS[*index*]. The *index* holds the name of the variable.

The example below shows how to use the super global variable $GLOBALS:

```
<?php
$x = 75;
$y = 25;
 function addition() {
  $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}
addition();
echo $z;
?>
```

(2) PHP $_SERVER

$_SERVER is a PHP super global variable which holds information about headers, paths, and script locations.

The example below shows how to use some of the elements in $_SERVER:

```
<?php
echo $_SERVER['PHP_SELF'];
echo "<br>";
echo $_SERVER['SERVER_NAME'];
echo "<br>";
echo $_SERVER['HTTP_HOST'];
echo "<br>";
echo $_SERVER['HTTP_REFERER'];
echo "<br>";
echo $_SERVER['HTTP_USER_AGENT'];
echo "<br>";
echo $_SERVER['SCRIPT_NAME'];
?>
```

(3) PHP $_REQUEST

PHP $_REQUEST is a PHP super global variable which is used to collect data after submitting an HTML form.

The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to this file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $_REQUEST to collect the value of the input field:

```
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  // collect value of input field
  $name = $_REQUEST['fname'];
  if (empty($name)) {
    echo "Name is empty";
  } else {
    echo $name;
  }
}
?>
```

```
</body>
</html>
```

(4) PHP $_POST

PHP $_POST is a PHP super global variable which is used to collect form data after submitting an HTML form with method="post". $_POST is also widely used to pass variables.

The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to the file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $_POST to collect the value of the input field:

```
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  // collect value of input field
  $name = $_POST['fname'];
  if (empty($name)) {
    echo "Name is empty";
  } else {
    echo $name;
  }
}
?>

</body>
</html>
```

(5) PHP $_GET

PHP $_GET is a PHP super global variable which is used to collect form data after submitting an HTML form with method="get".

$_GET can also collect data sent in the URL.

Assume we have an HTML page that contains a hyperlink with parameters:

```
<html>
<body>

<a href="test_get.php?subject=PHP&web=W3schools.com">Test $GET</a>

</body>
</html>
```

When a user clicks on the link "Test $GET", the parameters "subject" and "web" are sent to "test_get.php", and you can then access their values in "test_get.php" with $_GET.

The example below shows the code in "test_get.php":

```
<html>
<body>

<?php
echo "Study " . $_GET['subject'] . " at " . $_GET['web'];
?>

</body>
</html>
```