INTRODUCTION TO PHP

INTRODUCTION TO PHP

PHP was originally created by **Rasmus Lerdorf** in 1994. PHP is one of the most widely used server-side scripting language for web development. Popular websites like Facebook, Yahoo are developed using PHP. PHP stands for **Hypertext Pre-Processor**. it is a scripting language used to develop static and dynamic webpages and web applications. Here are a few important things you must know about PHP:

1. PHP is an Interpreted language; it doesn't need a compiler.
2. To run and execute PHP code, we need a Web server.
3. PHP is a server-side scripting language, which means that PHP is executed on the server and the result is sent to the browser.
4. PHP is open source and free.

PHP VARIAVLE

Variables are used to store data, like string, Text and numbers. Variable values can change over the course of a script.

Important things to know about variables:

1. All variables in PHP are signified with dollar sign ($).
2. The value of a variable is the value of its assignment.
3. Variables are assigned with the = operator, with the variable on the left-hand side and the expression on the right.
4. Variables used before they are assigned have default values.
5. PHP does a good job of automatically converting types from one to another when necessary

Ex:1

```
<?php
$txt = "Hello World!";
$number = 10;
echo $txt;
echo $number;
?>
```

PHP OPERATORS:

Operators are used to perform operations on some values. In other words, we can describe operators as something that takes some values, performs some operation on them and gives a result. From example: 1 + 2 = 3 this expression is an operator.

Just like any other programming language, PHP also supports various types of operations like the arithmetic operations (addition, subtraction, others), logical operations (AND, OR), Increment/Decrement Operations. PHP provides us with many operators to perform such operations on various operands and values.

Below are the groups of operators:

- Arithmetic Operators
- Logical Operators
- Comparison Operators
- Conditional Operators
- Assignment Operators
- Array Operators
- Increment/Decrement Operators
- String Operators
- Spaceship Operators (Introduced in PHP 7)

Arithmetic Operators

Operator	Description	Example
+	Addition	$x + $y
-	Subtraction	$x - $y
*	Multiplication	$x * $y
/	Division	$x / $y
%	Modulus	$x % $y

Ex:1

```
<?php
error_reporting();
$x = $_POST['fst'];
$y = $_POST['snd'];
$a = $x + $y;
$b = $x - $y;
$c = $x * $y;
$d = $x / $y;
$e = $x % $y;
?>
<html>
<head>
<title>Operators</title>
<style>
input[type=text] {
  width: 100%;
  padding: 12px 20px;
  margin: 8px 0;
```

```
  display: inline-block;
  border: 1px solid #ccc;
  border-radius: 4px;
  box-sizing: border-box;
}
input[type=submit] {
  width: 20%;
  background-color: #4CAF50;
  color: white;
  padding: 14px 20px;
  margin: 8px 0;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  float:right;
}
</style>
</head>
<body>
<table style="width:40%; margin:0 auto; background-color:brown; margin-top:40px;
color:#fff;">
<form action="" method="POST">

<tr>
<td><h3>Enter No1</h3></td>
<td><input type="text" name="fst" /></td>
</tr>
<tr>
<td><h3>Enter No2</h3></td>
<td><input type="text" name="snd" /></td>
</tr>
```

```
<tr>
<td><h1>Result</h1></td>

<td>
<?php echo "Add:$a <br />"; ?>
<?php echo "Sub:$b <br />"; ?>
<?php echo "Mul:$c <br />"; ?>
<?php echo "Div:$d <br />"; ?>
<?php echo "Moduler:$e <br />"; ?>
</td>
</tr>
<tr>
<td colspan="2">
<input type="submit" value="RESULT">
</td>
</tr>
</form>
</table>
</body>
</html>
```

PHP Assignment Operators

The assignment operators are used to assign values to variables.

Operator	Description	Example
=	Assign	$x = $y
+=	Add and assign	$x += $y
-=	Subtract and assign	$x -= $y
*=	Multiply and assign	$x *= $y
/=	Divide and assign quotient	$x /= $y
%=	Divide and assign modulus	$x %= $y

EX:1

```
<html>

  <head>
    <title>Assignment Operators</title>
  </head>

  <body>

    <?php

       $a = 20;
       $b = 10;
        echo "<br />";
                  echo "Value A : 20 <br />";
                  echo "Value B : 10 <br />";
       $c = $a + $b;
       echo "total: $c <br/><br/>";

       $c += $a;
       echo "Total + a: $c <br/>";

       $c -= $a;
       echo "Total -a : $c <br/>";

       $c *= $a;
       echo "Multiple a*: $c <br/>";
```

```
    $c /= $a;
    echo "Division a/: $c <br/>";

    $c %= $a;
    echo "Modulus a%: $c <br/>";
  ?>

 </body>
</html>
```

PHP Comparison Operators

The comparison operators are used to compare two values in a Boolean fashion.

Operator	Name	Example
==	Equal	$x == $y
===	Identical	$x === $y
!=	Not equal	$x != $y
<>	Not equal	$x <> $y
!==	Not identical	$x !== $y
<	Less than	$x < $y
>	Greater than	$x > $y
>=	Greater than or equal to	$x >= $y
<=	Less than or equal to	$x <= $y

EX:1

```
<html>
  <head>
    <title>Comparison Operators</title>
  </head>

  <body>

    <?php

       $a = 42;
       $b = 20;

           echo "A value:$a <br />";
           echo "B value:$b <br />";

       if( $a == $b ) {
         echo " a==b: is equal to b<br/>";
       }else {
         echo " a==b: is not equal to b<br/>";
       }

       if( $a > $b ) {
         echo " a>b: is greater than  b<br/>";
       }else {
         echo "a>b: is not greater than b<br/>";
       }

       if( $a < $b ) {
         echo "a<b: is less than  b<br/>";
       }else {
         echo "a<b: a is not less than b<br/>";
       }

       if( $a != $b ) {
         echo " a!=b : a is not equal to b<br/>";
       }else {
         echo " a!=b : a is equal to b<br/>";
       }

       if( $a >= $b ) {
         echo " a>=b: a is either greater than or equal to b<br/>";
       }else {
         echo "a>=b : a is neither greater than nor equal to b<br/>";
```

```
      }

      if( $a <= $b ) {
        echo "a<=b : a is either less than or equal to b<br/>";
      }else {
        echo "a<=b : a is neither less than nor equal to b<br/>";
      }
    ?>

  </body>
</html>
```

PHP Incrementing and Decrementing Operators

There are two types of incrementing and decrementing operators, post incrementing/decrementing operators and pre incrementing /decrementing operators. Post incrementing operators are placed after a variable name, and pre incrementing /decrementing operators are placed before the name.

Operator	Name
++$x	Pre-increment
$x++	Post-increment
--$x	Pre-decrement
$x--	Post-decrement

Incrementing and Decrementing Operators

Examples:

Post Incrementing

Ex:1

```
<?php
$a = 8;
$b = $a++;
echo "$b <br />";
echo $a;
?>
```

Post Decrementing

Ex:2

```
<?php
$a = 10;
$b = $a--;
echo "$b <br />";
echo $a;
?>
```

Pre-Incrementing

Ex:3

```
<?php
$a = 8;
$b = ++$a;
echo "$b <br />";
echo $a;
?>
```

Pre-Decrementing
Ex:4

```
<?php
$a = 10;
$b = --$a;
echo "$b <br />";
echo $a;
?>
```

PHP Logical Operators

The logical operators are typically used to combine conditional statements.

Operator	Name	Example
and	And	$x and $y
or	Or	$x or $y
xor	Xor	$x xor $y
&&	And	$x && $y
||	Or	$x || $y
!	Not	!$x

AND

Ex:1

```
<html>
<body>
<?php
$x = 100;
$y = 50;

if ($x == 100 and $y == 50) {
   echo "Both and different No:";
}
?>

</body>
</html>
```

NOR

Ex: 1

```
<html>
<body>

<?php
$x = 100;
$y = 50;

if ($x == 100 xor $y == 80) {
   echo "Hello world!";
}
?>

</body>
</html>
```

NOT

Ex :1

```
<html>
<body>

<?php
$x = 100;

if ($x !== 100) {
   echo "Hello world!";
}
?>
```

PHP String Operators

These operators are tokens of a programming language that allow some operation on variables. Strings in any programming language can be combined to a variable. PHP String operators are (.) dot operator and (.=) concatenation assignment operator .

Operator	Description	Example
.	Concatenation	$str1. $str2
.=	Concatenation assignment	$str1. = $str2

Dot Operator (.)

Ex:1

```
<?php
$my_hello='Hello ';
$my_world='World ';
$my_welcome='Welcome ';
$my_php= 'Come and learn ';
$my_site= "String Operator ";

echo $my_hello.$my_world.$my_welcome.$my_php.$my_site.'<br>';
?>
```

Concatenation Assignment Operator (.=)

Concatenation Assignment operator behaves like an assignment operator that combines the string on the right-hand side with the variable value on the left-hand side and updates the variable's value with concatenated string

Ex: 1

```
<?php
$my_msg="";
$hello='Hello ';
$world='World ';

$welcome='Welcome ';

$php= 'PHP programming ';
```

```
$site= "Application ";

$my_msg.=$hello;
echo $my_msg.'<br>';
$my_msg.=$world;
echo $my_msg.'<br>';
$my_msg.=$welcome;
echo $my_msg.'<br>';
$my_msg.=$site.'<br>';
echo $my_msg.'<br>';
?>
```

PHP Array Operators

Arrays is a type of data structure that allows us to store multiple elements of similar data type under a single variable.

. The arrays are helpful to create a list of elements of similar types, which can be accessed using their index or key.

Suppose we want to store five names and print them accordingly. This can be easily done by the use of five different string variables. But if instead of five, the number rises to a hundred, then it would be really difficult for the developer to create so many different variables.

Here array comes into play and helps us to store every element within a single variable and also allows easy access using an index or a key. An array is created using an **array()** function in PHP.

There are basically three types of arrays in PHP:

- **Indexed or Numeric Arrays:** An array with a numeric index where values are stored linearly.
- **Associative Arrays:** An array with a string index where instead of linear storage, each value can be assigned a specific key.
- **Multidimensional Arrays:** An array which contains single or multiple array within it and can be accessed via multiple indices.

The array operators are used to compare arrays:

Operator	Name	Example
+	Union	$x + $y
==	Equality	$x == $y
===	Identity	$x === $y
!=	Inequality	$x != $y
<>	Inequality	$x <> $y
!==	Non-identity	$x !== $y

Ex:1

```
<?php
 $arr_variable = array(
 1 => "PHP",
 2 => "MySQL",
 3 => "Java",
$pro = "Learning",
 );
 foreach( $arr_variable as $arritem )
 {
 echo "$arritem <br />";
 }
 ?>
```

Numeric Array

These arrays can store numbers, strings and any object but their index will be represented by numbers

Ex:1

```
<html>
  <body>
  <?php
     $num = array( 1, 2, 3, 4, 5);
       foreach( $num as $value ) {
         echo "$value <br />";
       }
       $num[0] = "one";
       $num[1] = "two";
       $num[2] = "three";
       $num[3] = "four";
      $num[4] = "five";
        foreach( $num as $value ) {
         echo "Value carry: $value <br />";
       }
    ?>
 </body>
</html>
```

Associative Arrays

The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index. Associative array will have their index as string so that you can establish a strong association between key and values.

Ex:1

```
<html>
<body>
<?php
      /* First method to associate create array. */
      $salaries = array("admin" => 2000, "acc" => 1000, "teach" => 500);
        echo "Salary  ". $salaries['admin'] . "<br />";
      echo "Salary  ".  $salaries['acc']. "<br />";
      echo "Salary ".  $salaries['teach']. "<br />";
       /* Second method to create array. */
      $salaries['admin'] = "high";
      $salaries['acc'] = "medium";
      $salaries['teach'] = "low";
      echo "Salary Admin ". $salaries['admin'] . "<br />";
      echo "Salary Acc ".  $salaries['acc']. "<br />";
      echo "Salary Teach ".  $salaries['teach']. "<br />";
   ?>
 </body>
</html>
```

Multidimensional Arrays

A multi-dimensional array each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are accessed using multiple index.

Ex:1

```
<html>
  <body>

    <?php
      $marks = array(
        "student1" => array (
          "physics" => 35,
          "maths" => 30,
          "chemistry" => 39
        ),

        "student2" => array (
          "physics" => 30,
          "maths" => 32,
          "chemistry" => 29
        ),

        "student3" => array (
          "physics" => 31,
          "maths" => 22,
```

```
            "chemistry" => 39
         )
      );

      /* Accessing multi-dimensional array values */
      echo "Marks for student1 in physics : " ;
      echo $marks['student1']['physics'] . "<br />";

      echo "Marks for student2 in maths : ";
      echo $marks['student2']['maths'] . "<br />";

      echo "Marks for student3 in chemistry : " ;
      echo $marks['student3']['chemistry'] . "<br />";
   ?>

  </body>
</html>
```

PHP PROGRAMMING (INSERT DATA INTO DATABASE)

Create Table

```
CREATE TABLE tbldemo (
id INT(6) AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
)
```

PHP BACKEND PROGRAMMING

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "demo";
$conn = mysqli_connect($servername, $username, $password, $dbname);
if (!$conn) {
  echo "Failed Connection";
}
$sql = "INSERT INTO tbdemo(firstname, lastname, email)
VALUES ('Dev', 'Panjiyar', 'dn@gmail.com.com')";
$res = mysqli_query($conn,$sql);
if(!$res)
{        echo "failed to send data";}
else
{echo "Success";}
?>
```

FRONTEND IN PHP

```
<html>
        <head>
<title>wt</title>
</head>
        <body>
        <h1>INSERT DATA INTO</h1>
        <form action="back.php" method="POST">
        <table><tr>
        <td>Enter Firstname</td>
        <td><input type="text" name="f"></td>
        </tr><tr>
        <td>Enter Lastname</td>
        <td><input type="text" name="l"></td>
        </tr>
        <tr>
        <td>Enter E-mail</td>
        <td><input type="text" name="e"></td>
        </tr>
        <tr>
        <td></td>
        <td><input type="submit" value="INSERT"></td>
        </tr>
        </td>
        </tr>
        </table>
</form>
</body>
</html>
```

PHP DISPLAY RECORDS FROM DATABASE

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "demo";

$conn = mysqli_connect($servername, $username, $password, $dbname);

if (!$conn) {
  echo "no connected!";
}

$sql = "SELECT id, firstname, lastname FROM tbdemo";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {

  while($row = mysqli_fetch_array($result)) {
    echo "id:" . $row["id"]. "Name:" . $row["firstname"]. " " . $row["lastname"]. "<br>";
  }
}
 else
{
  echo "0 results";
}

mysqli_close($conn);
?>
```

Role Based Access Control

Role-Based Access Control (RBAC) is a security model whereby users are granted access to resources based on their role in the company

.

The basic principle of Role-Based Access Control is simple: The Finance department can't see HR data.

When implemented correctly, RBAC will be transparent to the users. Role assignment happens behind the scenes, and each user has access to the applications and data that they need to do their job.

RBAC Example: The Basics

1. **Role assignment:** A subject can exercise a permission only if the subject has selected or been assigned a role.
2. **Role authorization:** A subject's active role must be authorized. That is, I can't just assign myself to a role. I need authorization.
3. **Permission authorization:** A subject can exercise a permission only if the permission is authorized for the subject's active role. With rules 1 and 2, this rule ensures that users can only exercise permissions for which they are authorized.

Benefits Of RBAC

1. RBAC helps maximize operational efficiency, protects your data from being leaked or stolen, reduces admin and IT support work, and makes it easier to meet audit requirements.
2. RBAC also reduces IT and administrative load across the organization and increases the productivity of the users. IT doesn't have to manage personalized permissions for every user, and it's easier for the right users to get to the right data.
3. Managing new users and guest users without time-consuming and difficult

4. companies can implement RBAC systems to meet the regulatory and statutory requirements for confidentiality and privacy because executives and IT departments can more effectively manage how the data is accessed and used

XML

XML stands for Extensible Markup Language. It is a text-based markup language derived from Standard Generalized Markup Language (SGML). XML tags identify the data and are used to store and organize the data, rather than specifying how to display it like HTML tags, which are used to display the data. XML is not going to replace HTML in the near future, but it introduces new possibilities by adopting many successful features of HTML.

There are three important characteristics of XML that make it useful in a variety of systems and solutions:

- XML is extensible: XML allows you to create your own self-descriptive tags or language, that suits your application.
- XML carries the data, does not present it: XML allows you to store the data irrespective of how it will be presented.
- XML is a public standard: XML was developed by an organization called the World Wide Web Consortium (W3C) and is available as an open standard.

XML Usage

- XML can work behind the scene to simplify the creation of HTML documents for large web sites.
- XML can be used to exchange the information between organizations and systems.
- XML can be used for offloading and reloading of databases.
- XML can be used to store and arrange the data, which can customize your data handling needs.
- XML can easily be merged with style sheets to create almost any desired output.
- Virtually, any type of data can be expressed as an XML document.

An XML document is a basic unit of XML information composed of elements and other markup in an orderly package. An XML document can contain a wide variety of data. For

example, database of numbers, numbers representing molecular structure or a mathematical equation.

XML Document Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
  <to>To Dev</to>
  <from>You are </from>
  <heading>Remindered</heading>
  <body>Don't forget me this weekend! </body>
</note>
```