Module I: Overview of PHP, Benefits, and drawbacks in running PHP as a Sever Side Script, PHP Language Basics: The building blocks of PHP: variables, global & super global Data types: Set type, type casting, test type, Operators & Expressions, Flow control functions in PHP, Functions: Defining a function variable scope, calling a function returning values, setting default values for arguments, passing variable reference, built in functions

MODULE 1- OVERVIEW OF PHP

What is PHP?

• PHP is an acronym for "PHP: Hypertext Preprocessor"

• PHP is a widely-used, open source scripting language

• PHP scripts are executed on the server

• PHP is free to download and use

What is a PHP File?

• PHP files can contain text, HTML, CSS, JavaScript, and PHP code

• PHP code is executed on the server, and the result is returned to the browser as plain HTML

• PHP files have extension ".php"

What Can PHP Do?

• PHP can generate dynamic page content

• PHP can create, open, read, write, delete, and close files on the server

• PHP can collect form data

• PHP can send and receive cookies

• PHP can add, delete, modify data in your database

• PHP can be used to control user-access

• PHP can encrypt data

Basic PHP Syntax

• A PHP script can be placed anywhere in the document.

• A PHP script starts with <?php and ends with ?>:

• The default file extension for PHP files is ".php".

• A PHP file normally contains HTML tags, and some PHP scripting code.

```
<?php
// PHP code goes here
?>
```

Advantages of Using PHP Frameworks

1.Speed up custom web application development

The tools, features, and code snippets provided by PHP frameworks help developers to accelerate custom web application development.

2.Simplify web application maintenance

The PHP frameworks simplify web application development and maintenance by supporting modelview-controller (MVC) architecture.

3.No need to write additional code

PHP, unlike other programming languages, does not allow programmers to express concepts without writing longer lines of code.

5.Work with databases more efficiently

Most PHP frameworks allow programmers to work with several widely used relational databases.

6.Protect websites from targeted security attacks

PHP is one of the most unsecured programming languages. Often programmers must explore ways to protect the PHP applications from various security attacks.

Disadvantages of Using PHP Frameworks

1.Programmers need to learn PHP frameworks instead of PHP

The PHP frameworks enable programmers to add functionality to a web application without writing additional code. But the programmers have to put some time and effort to learn the PHP framework.

2.Lack of option to modify core behavior

In addition to proving a basic structure for web application development, the PHP frameworks further accelerate custom web application development.

3.Affect Speed and performance of websites

Most PHP frameworks come with robust features and tools to accelerate development of large and complex websites.

Variable in PHP

Variables are used to store data, like string of text, numbers, etc. Variable values can change over the course of a script. Here're some important things to know about 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 (=) used to assign value to a variable.

In PHP variable can be declared as: $var_name = value;

Example $a=10;

Super Global Variables in PHP

PHP super global variable is used to access global variables from anywhere in the PHP script. PHP Super global variables is accessible inside the same page that defines it, as well as outside the page. while local variable's scope is within the page that defines it.Several predefined variables in PHP are "superglobals", which means they are available in all scopes throughout a script. There is no need to do global $variable; to access them within functions or methods.

These superglobal variables are:

• $GLOBALS

• $_SERVER

• $_GET

• $_POST

• $_FILES

• $_COOKIE

• $_SESSION

• $_REQUEST

• $_ENV

Typecasting in Php

The meaning of type casting is to use the value of a variable with different data type. In other word typecasting is a way to utilize one data type variable into the different data type. Typecasting is the explicit conversion of data type because user explicitly defines the data type in which he wants to cast. In this tutorial, we will explore various aspects of PHP Type casting.PHP does not require or support type definition of the variable. In PHP we never define data type while declaring the variable. In PHP variables automatically decide the data type on the basis of the value assignment or context. For example:

```
<?php

$i =1;

var_dump($i); //$i is integer

$i = 2.3;

var_dump($i); //$i is float

$i = "php type casting";

var_dump($i); //$i is string

?>
```

settype() Function

The settype() function is a built-in function in PHP. The settype() function is used to the set the type of a variable. It is used to set type or modify type of an existing variable.

Syntax:

```
boolean settype($variable_name, $type)
```

Parameters:

The settype() function accepts two parameters as shown in above syntax and are described below.

```
 <?php

$var1 = "123xyz";

$var2 = 3;

$r = true;
```

```
settype($var1, "integer");
settype($var2, "float");
settype($r, "string");
echo $var1."\n";
echo $var2."\n";
echo $r."\n";
 ?>
```

Output:

```
123
3
1
```

PHP Operators

What is Operator? Simple answer can be given using expression 4 + 5 is equal to 9. Here 4 and 5 are called operands and + is called operator. PHP language supports following type of operators.

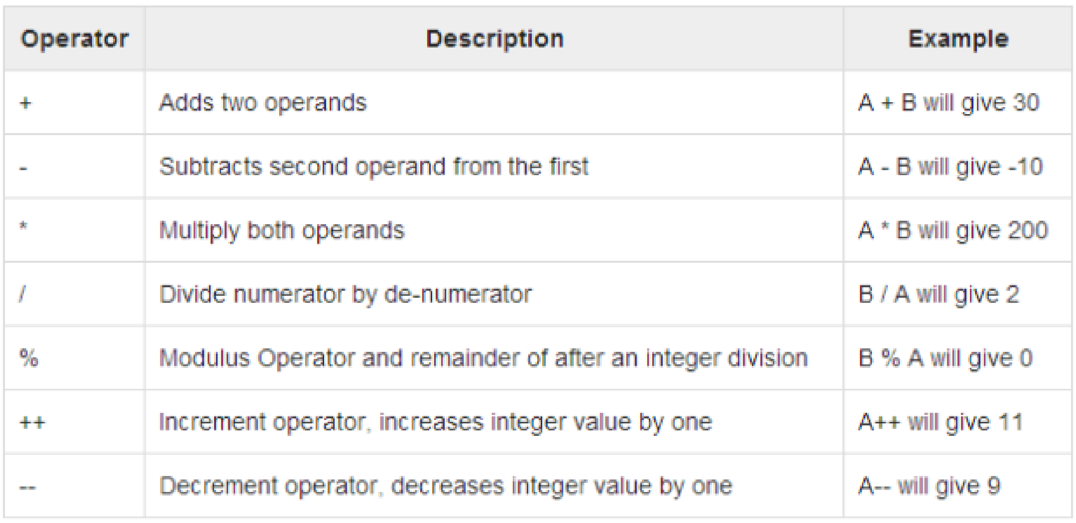
1. Arithmetic Operators
2. Comparison Operators
3. Logical (or Relational) Operators
4. Assignment Operators
5. Conditional (or ternary) Operators

Lets have a look on all operators one by one.

1.Arithmetic Operators:-

• There are following arithmetic operators supported by PHP language −

• Assume variable A holds 10 and variable B holds 20 then –

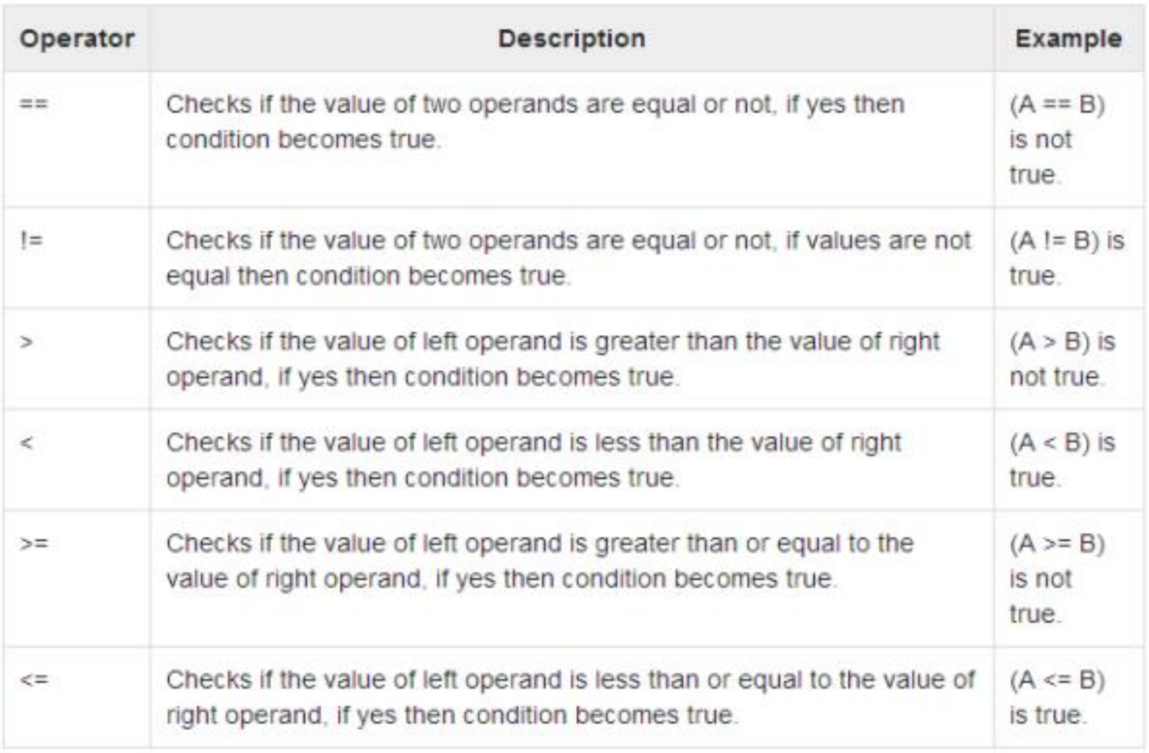


Operator	Description	Example
+	Adds two operands	A + B will give 30
-	Subtracts second operand from the first	A - B will give -10
*	Multiply both operands	A * B will give 200
/	Divide numerator by de-numerator	B / A will give 2
%	Modulus Operator and remainder of after an integer division	B % A will give 0
++	Increment operator, increases integer value by one	A++ will give 11
--	Decrement operator, decreases integer value by one	A-- will give 9

2.Comparison Operators

• There are following comparison operators supported by PHP language

• Assume variable A holds 10 and variable B holds 20 then –

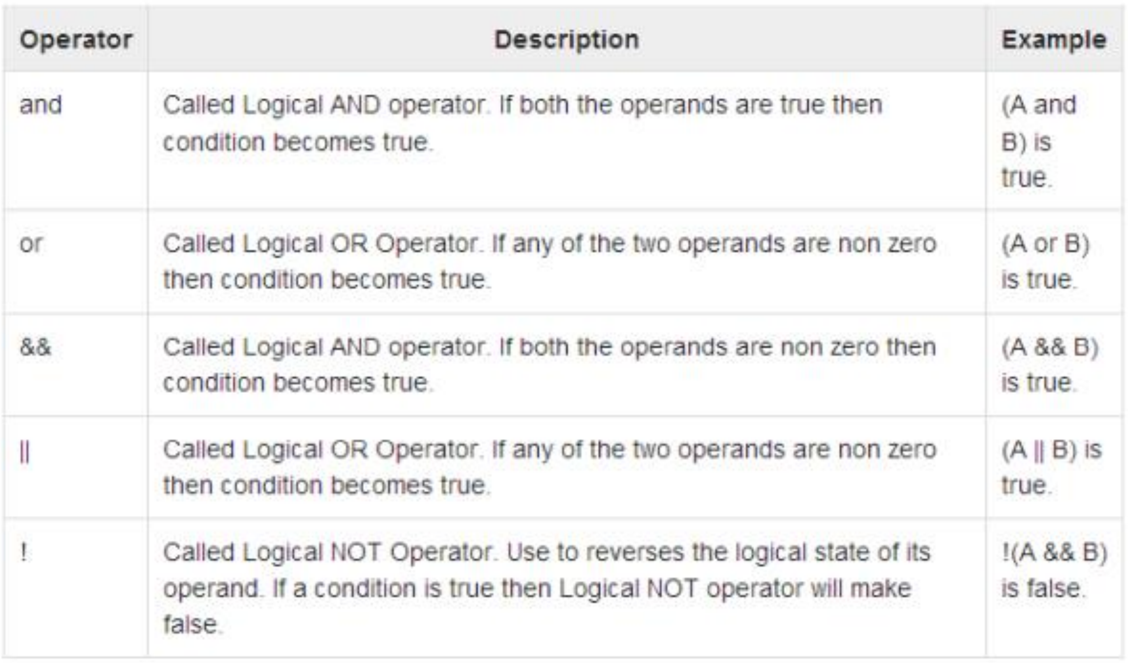


Operator	Description	Example
==	Checks if the value of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

3.Logical Operators

• There are following logical operators supported by PHP language

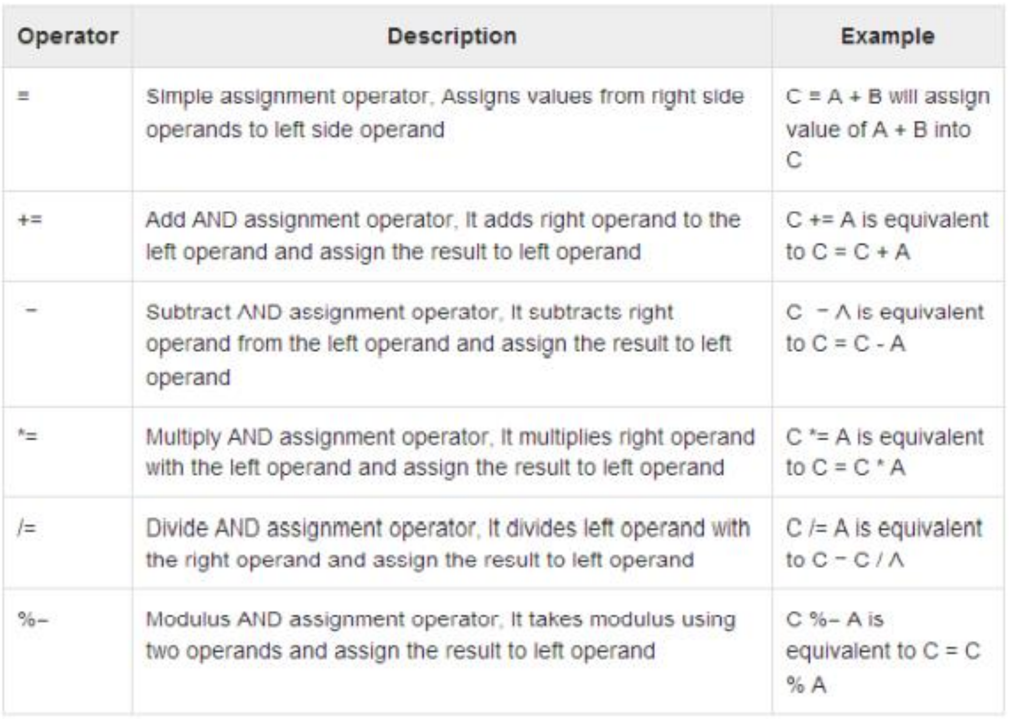
• Assume variable A holds 10 and variable B holds 20 then



Operator	Description	Example
and	Called Logical AND operator. If both the operands are true then condition becomes true.	(A and B) is true.
or	Called Logical OR Operator. If any of the two operands are non zero then condition becomes true.	(A or B) is true.
&&	Called Logical AND operator. If both the operands are non zero then condition becomes true.	(A && B) is true.
||	Called Logical OR Operator. If any of the two operands are non zero then condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is false.

4.Assignment Operators

There are following assignment operators supported by PHP language –



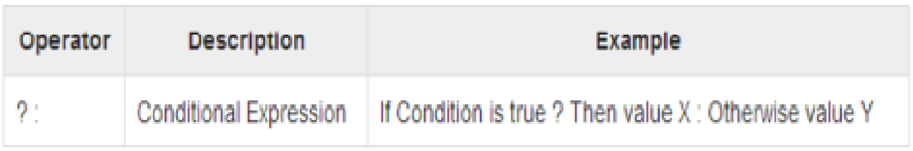
Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assign value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
−	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C − A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C − C / A
%−	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	C %− A is equivalent to C = C % A

5.Conditional Operator

There is one more operator called conditional operator. This first evaluates an expression for a true or

false value and then execute one of the two given statements depending upon the result of the

evaluation. The conditional operator has this syntax −



Operator	Description	Example
? :	Conditional Expression	If Condition is true ? Then value X : Otherwise value Y

Expressions in PHP

• Expressions are the most important building blocks of PHP. In PHP, almost anything you write is an expression. The simplest yet most accurate way to define an expression is "anything that has a value".

• The most basic forms of expressions are constants and variables. When you type "$a = 5", you're assigning '5' into$a. '5', obviously, has the value 5, or in other words '5' is an expression with the value of 5 (in this case, '5' is an integer constant).

• Example $s=$a+$b;

PHP - Decision Making

The ifelse, elseif and switch statements are used to take decision based on the different condition.You can use conditional statements in your code to make your decisions. PHP supports following three decision making statements −

1.if...else statement − use this statement if you want to execute a set of code when a condition is true and another if the condition is not true

2.elseif statement − is used with the if...else statement to execute a set of code if oneof the several condition is true

3.switch statement − is used if you want to select one of many blocks of code to be executed, use the Switch statement. The switch statement is used to avoid long blocks of if..elseif..else code.

1.if...Else Statement

If you want to execute some code if a condition is true and another code if a condition is false,

use the if....else statement.

Syntax

```
if (condition)
 code to be executed if condition is true;
else
 code to be executed if condition is false;
```

Example

```
<html>
 <body>
  <?php
 $d = 10;
  if ($d %2==0
 echo "EVEN”;
  else
 echo "ODD
 ?>
  </body>
</html>
```

It will produce the following result −

```
EVEN
```

2. else If Statement

If you want to execute some code if one of the several conditions are true use the elseif statement

Syntax

```
if (condition)
 code to be executed if condition is true;
```

```
elseif (condition)
 code to be executed if condition is true;
else
 code to be executed if condition is false;
```

Example

Output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current

time is less than 20. Otherwise it will output "Have a good night!":

```
<?php
$t = date("H");
if ($t < "10")
{
 echo "Have a good morning!";
}
elseif ($t < "20")
{
 echo "Have a good day!";
}
Else
{
 echo "Have a good night!";
}
?>
```

3.Switch Statement

If you want to select one of many blocks of code to be executed, use the Switch statement.

The switch statement is used to avoid long blocks of if..elseif..else code.

Syntax

```
switch (expression){
 case label1:
 code to be executed if expression = label1;
 break;
  case label2:
 code to be executed if expression = label2;
 break;
 default:
  code to be executed
 if expression is different
 from both label1 and label2;
}
```

Example

The switch statement works in an unusual way. First it evaluates given expression then seeks a lable to match the resulting value. If a matching value is found then the code associated with the matching label will be executed or if none of the lable matches then statement will execute any specified default code.

```
<html>
 <body>


 <?php
 $d = date("D");
  switch ($d){
 case "Mon":
 echo "Today is Monday";
 break;
case "Tue":
 echo "Today is Tuesday";
 break;
  case "Wed":
 echo "Today is Wednesday";
 break;
  case "Thu":
 echo "Today is Thursday";
 break;
  case "Fri":
 echo "Today is Friday";
 break;
  case "Sat":
 echo "Today is Saturday";
 break;
  case "Sun":
 echo "Today is Sunday";
 break;
  default:
 echo "Wonder which day is this ?";
 }
 ?>
  </body>
</html>
```

It will produce the following result −

```
Today is Monday
```

PHP - Looping

Loops in PHP are used to execute the same block of code a specified number of times. PHP supports following four loop types.

• for − loops through a block of code a specified number of times.

• while − loops through a block of code if and as long as a specified condition is true

• do...while − loops through a block of code once, and then repeats the loop as long as a special condition is true.

• foreach − loops through a block of code for each element in an array.

We will discuss about continue and break keywords used to control the loops execution.

1.for loop statement

The for statement is used when you know how many times you want to execute a statement or a block of statements.

Syntax

```
for (initialization; condition; increment)
{
 code to be executed;
}
```

The initializer is used to set the start value for the counter of the number of loop iterations. A variable may be declared here for this purpose and it is traditional to name it $i.

Example

```
<html>
 <body>
  <?php
  for( $i = 0; $i<5; $i++ )
 {
 echo”Kiran”;
 }
  ?>
  </body>
</html>
```

Output :This program will print Kiran 5 times

2.while loop statement

The while statement will execute a block of code if and as long as a test expression is true. If the test expression is true then the code block will be executed. After the code has executed the test expression will again be evaluated and the loop will continue until the test expression is found to be false

Syntax

```
while (condition)
{
 code to be executed;
}
```

Example

```
<html>
 <body>
  <?php
 $i=0;
 while( $i < 5)
 {
echo “Kiran”
 }
 ?>
  </body>
</html>
```

Output :This program will print Kiran 5 times

3.do...while loop statement

The do...while statement will execute a block of code at least once - it then will repeat the loop as long as a condition is true.

Syntax

```
do
```

```
{
 code to be executed;
}
while (condition);
```

Example:

```
<html>
 <body>
  <?php
 $i = 0;
 do
 {
 echo “Kiran”;
 }
while( $i < 5 );
 ?>
  </body>
</html>
```

Output :This program will print Kiran 5 times

4.foreach loop statement

The foreach statement is used to loop through arrays. For each pass the value of the current array element is assigned to $value and the array pointer is moved by one and in the next pass next element will be processed.

Syntax

```
foreach (array as value)
{
 code to be executed;
}
```

Example

Try out following example to list out the values of an array.

```
<html>
 <body>
  <?php
 $array = array( 1, 2, 3, 4, 5);
  foreach( $array as $value ) {
 echo "Value is $value <br />";
 }
 ?>
  </body>
</html>
```

This will produce the following result −

```
Value is 1
Value is 2
Value is 3
Value is 4
Value is 5
```

Break statement

The PHP break keyword is used to terminate the execution of a loop prematurely.

The break statement is situated inside the statement block. It gives you full control and whenever you want to exit from the loop you can come out. After coming out of a loop immediate statement to the loop will be executed.

Example

In the following example condition test becomes true when the counter value reaches 3 and loop terminates.

```
<html>
 <body>
 <?php
 for($i=0;$i<=10;$i++)
{
If($i==4)
{
break;
}
echo $i;
}
 ?>
  </body>
</html>
```

This will produce the following result − 0 1 2 3

Loop stopped at i = 4

Continue statement

The PHP continue keyword is used to halt the current iteration of a loop but it does not terminate the loop. Just like the break statement the continue statement is situated inside the statement block containing the code that the loop executes, preceded by a conditional test. For the pass encountering continue statement, rest of the loop code is skipped and next pass starts.

Example

In the following example loop prints the value of array but for which condition becomes true it just skip the code and next value is printed.

```
<?php
for($i=0;$i<10;$i++)
{
If($i==4)
{
continue;
}
echo $i;
}
 ?>
```

This will produce the following result –

```
0 1 2 3 5 6 7 8 9
```

PHP - Functions

• A function is a block of statements that can be used repeatedly in a program.

• A function will not execute automatically when a page loads.

• A function will be executed by a call to the function.

You already have seen many functions like fopen() and fread() etc. They are built-in functions but PHP gives you option to create your own functions as well.

There are two parts which should be clear to you −

• Creating a PHP Function

• Calling a PHP Function

Create a User Defined Function in PHP

A user-defined function declaration starts with the word function:

Syntax

```
function functionName()
{
 code to be executed;
}
```

Example

```
function display()
{
echo “Hai welcome”;
}
```

PHP Functions with Parameters

Information can be passed to functions through arguments. An argument is just like a variable. Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma

Syntax

```
function functionName(parameter list)
{
 code to be executed;
}
```

Example

```
function display($a,$b)
{
$s=$a+$b;
echo $s;
}
```

Passing Arguments by Reference

It is possible to pass arguments to functions by reference. This means that a reference to the variable is manipulated by the function rather than a copy of the variable's value.Any changes made to an argument in these cases will change the value of the original variable. You can pass an argument by reference by adding an ampersand to the variable name in eithr the function call or the function definition.

Example:

```
<?php
 function calculate(&$a)
 {
 $a++;
 }
 $a=5;
 calculate($a);
 echo $a;
?>
```

Output:6

PHP Functions returning value

A function can return a value using the return statement in conjunction with a value or object. return stops the execution of the function and sends the value back to the calling code. Following example takes two integer parameters and add them together and then returns their sum to the calling program. Note that return keyword is used to return a value from a function.

```
<?php
 function addFunction($num1, $num2)
 {
 $sum = $num1 + $num2;
 return $sum;
 }
 $return_value = addFunction(10, 20);
 echo "Returned value from the function : $return_value";
 ?>
```

This will display following result −

```
Returned value from the function : 30
```

MODULE 2

ARRAYS

Arrays are indexed, which means that each entry is made up of a key and a value. The key is the index position, beginning with 0. The value is whatever value you associate with that position—a string, an integer, or whatever you want.

Creating Arrays

You can create an array using either the array() function or the array operator [].The array() function is usually used when you want to create a new array and populate it with more than one element at the same time. The array operator is used when you want to create a new array with just one element (for now), or when you want to add to an existing element.The following code snippet shows how to create an array called $rainbow, containing all its various colors:

```
$rainbow = array("red", "orange", "yellow", "green", "blue", "indigo", "violet");
```

The following snippet shows the same array being created incrementally using the array operator:

```
$rainbow[] = "red";

$rainbow[] = "orange";

$rainbow[] = "yellow";

$rainbow[] = "green";

$rainbow[] = "blue";

$rainbow[] = "indigo";

$rainbow[] = "violet";
```

Both create a seven-element array called $rainbow, with values starting at index position 0 and ending at index position 6. If you wanted to be literal about it, you could have specified the index positions, such as in this code:

However, PHP does this for you when positions are not specified, and that eliminates the possibility that you will misnumber your elements, as in this example:

```
$rainbow[0] = "red";

$rainbow[1] = "orange";

$rainbow[2] = "yellow";

$rainbow[5] = "green";
```

$rainbow[6] = “blue”;

$rainbow[7] = “indigo”;

$rainbow[8] = “violet”;

Regardless of whether you initially create your array with the array() function or the array operator, you can still add to it using the array operator:

$rainbow = array(“red”, “orange”, “yellow”, “green”, “blue”, “indigo”);

$rainbow[] = “violet”

Creating Associative Arrays

Whereas numerically indexed arrays use an index position as the key—0, 1, 2,and so forth—associative arrays utilize actual named keys. The following example demonstrates this by creating an array called $character with four elements:

$character = array(“name” => “Bob”,“occupation” => “superhero”,“age” => 30,“special power” => “x-ray vision”);

The four keys in the $character array are called name, occupation, age, and special power. The associated values are Bob, superhero, 30, and x-ray vision. You can reference specific elements of an associative array using the specific key,such as in this example:

<?php

echo $character[‘occupation’];

?>

The output of this snippet is this:

superhero

As with numerically indexed arrays, you can use the array operator to add to an associative array:

$character[‘supername’] = “Mega X-Ray Guy”;

This example adds a key called supername with a value of Mega X-Ray Guy.

Creating Multidimensional Arrays

The first two types of arrays hold strings and integers, whereas this third type holds other arrays. If each set of key/value pairs constitutes a dimension, a multidimensional array holds more than one series of these key/value pairs. In the given example defines a multidimensional array called $characters, each element of which contains an associative array

```
1:  <?php
2:  $characters = array(
3:                   array(
4:                     "name" => "Bob",
5:                     "occupation" => "superhero",
6:                     "age" => 30,
7:                     "special power" => "x-ray vision"
8:                    ),
9:                   array(
10:                    "name" => "Sally",
11:                    "occupation" => "superhero",
12:                    "age" => 24,
13:                    "special power" => "superhuman strength"
14:                   ),
15:                   array(
16:                    "name" => "Jane",
17:                    "occupation" => "arch villain",
18:                    "age" => 45,
19:                    "special power" => "nanotechnology"
20:                   )
21:                );
22: ?>
```

In line 2, the $characters array is initialized using the array() function. Lines3–8 represent the first element, lines 9–14 represent the second element, and lines15?20 represent the third element. These elements can be referenced as $characters[0], $characters[1], and $characters[2].However, if you attempt to print these elements:

<?php

echo $character[2];

?>

the output will be this:

Array

To access specific information found within the array element, you need to access the element index position and the associative name of the value you want to view. Take a look at this example:

<?php

echo $character[2][‘occupation’];

?>

It will print this:

Superhero

Some Array-Related Functions

Approximately 60 array-related functions are built into PHP, which you can read about in detail at `http://www.php.net/array`. Some of the more common (and useful) functions are explained in this section.

- `count()` and `sizeof()`—Each counts the number of elements in an array. Given the following array:

  ```
  $colors = array("blue", "black", "red", "green");
  ```

  both `count($colors);` and `sizeof($colors);` return a value of `4`.

- `each()` and `list()`—These usually appear together, in the context of stepping through an array and returning its keys and values. The following example steps through an associative array called `$character`, printing its key, the text `has a value of`, and the value, followed by an HTML break.

  ```
  while (list($key, $val) = each($characater)) {
       echo "$key has a value of $val <br>";
  }
  ```

- `foreach()`—This is also used to step through an array, assigning the value of an element to a given variable. The following example steps through an associative array called `$character`, prints some text, the value, and finally an HTML break.

  ```
  foreach($characater as $c) {
       echo "The value is $c <br>";
  }
  ```

- `reset()`—This rewinds the pointer to the beginning an array, as in this example:

  ```
  reset($character);
  ```

  This function is useful when you are performing multiple manipulations on an array, such as sorting, extracting values, and so forth.

- `array_push()`—This adds one or more elements to the end of an existing array, as in this example:

  ```
  array_push($existingArray, "element 1", "element 2", "element 3");
  ```

- `array_pop()`—This removes (and returns) the last element of an existing array, as in this example:

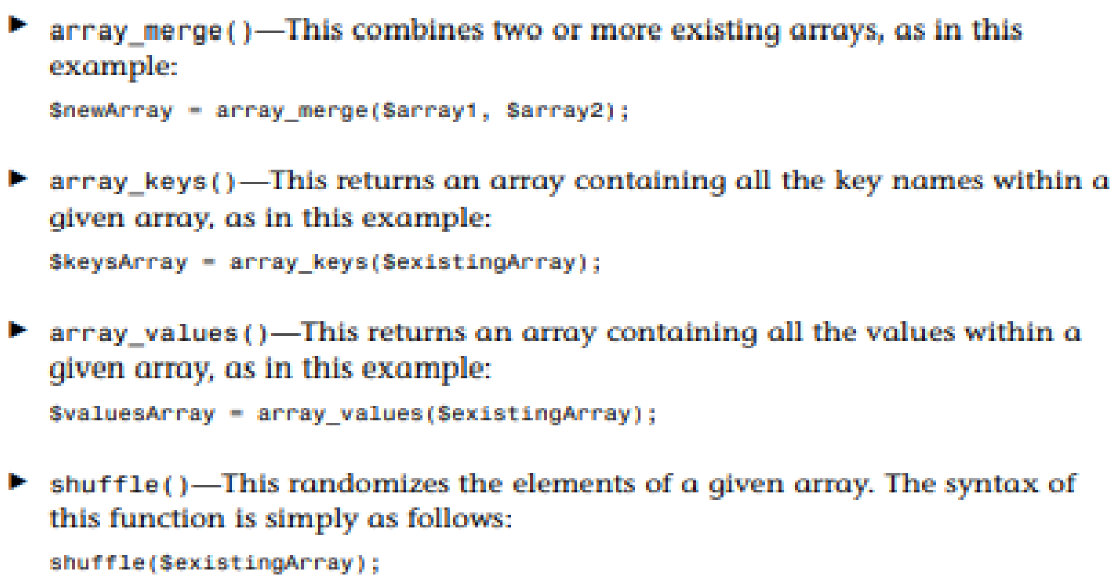
  ```
  $last_element = array_pop($existingArray);
  ```

- `array_unshift()`—This adds one or more elements to the beginning of an existing array, as in this example:

  ```
  array_unshift($existingArray, "element 1", "element 2", "element 3");
  ```

- `array_shift()`—This removes (and returns) the first element of an existing array, as in this example:

  ```
  $first_element = array_shift($existingArray);
  ```



- `array_merge()`—This combines two or more existing arrays, as in this example:

  ```
  $newArray = array_merge($array1, $array2);
  ```

- `array_keys()`—This returns an array containing all the key names within a given array, as in this example:

  ```
  $keysArray = array_keys($existingArray);
  ```

- `array_values()`—This returns an array containing all the values within a given array, as in this example:

  ```
  $valuesArray = array_values($existingArray);
  ```

- `shuffle()`—This randomizes the elements of a given array. The syntax of this function is simply as follows:

  ```
  shuffle($existingArray);
  ```

OOP Case

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

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

Define a Class

A class is defined by using the class keyword, followed by the name of the class and a pair of curly braces ({}). All its properties and methods go inside the braces:

Syntax

```
<?php
class Fruit {
  // code goes here...
}
?>
```

Below we declare a class named Fruit consisting of two properties ($name and $color) and two methods set_name() and get_name() for setting and getting the $name property:

```
<?php
class Fruit {
  // Properties
  public $name;
  public $color;
  // Methods
  function set_name($name) {
```

```
    $this->name = $name;
  }
  function get_name() {
    return $this->name;
  }
}
?>
```

Define Objects

Classes are nothing without objects! We can create multiple objects from a class. Each object has all the properties and methods defined in the class, but they will have different property values.

Objects of a class is created using the new keyword.In the example below, $apple and $banana are instances of the class Fruit:Example

```
<?php
class Fruit {
  // Properties
  public $name;
  public $color;
  // Methods
  function set_name($name) {
    $this->name = $name;
  }
  function get_name() {
    return $this->name;
  }
}
$apple = new Fruit();
$banana = new Fruit();
$apple->set_name('Apple');
$banana->set_name('Banana');
echo $apple->get_name();
echo "<br>";
echo $banana->get_name();
?>
```

Will give

Apple

Banana

In the example below, we add two more methods to class Fruit, for setting and getting the $color property:

```
<?php
class Fruit {
  // Properties
  public $name;
  public $color;

  // Methods
  function set_name($name) {
    $this->name = $name;
  }
  function get_name() {
    return $this->name;
  }
  function set_color($color) {
    $this->color = $color;
  }
  function get_color() {
    return $this->color;
  }
}

$apple = new Fruit();
$apple->set_name('Apple');
$apple->set_color('Red');
echo "Name: " . $apple->get_name();
echo "<br>";
echo "Color: " . $apple->get_color();
?>
```

Willgive

Name: Apple
Color: Red

PHP - The $this Keyword

The $this keyword refers to the current object, and is only available inside methods.

Look at the following example:

```
<?php
class Fruit {
  public $name;
}
$apple = new Fruit();
?>
```

So, where can we change the value of the $name property? There are two ways:

1. Inside the class (by adding a set_name() method and use $this):

```
<?php
class Fruit {
  public $name;
  function set_name($name) {
    $this->name = $name;
  }
}
$apple = new Fruit();
$apple->set_name("Apple");

echo $apple->name;
?>
```

Will give

```
Apple
```

2. Outside the class (by directly changing the property value):

```
<?php
class Fruit {
  public $name;
}
$apple = new Fruit();
$apple->name = "Apple";

echo $apple->name;
?>
```

Will give

```
Apple
```

PHP - The __construct Function

A constructor allows you to initialize an object's properties upon creation of the object.If you create a `__construct()` function, PHP will automatically call this function when you create an object from a class.

Notice that the construct function starts with two underscores (__)!We see in the example below, that using a constructor saves us from calling the set_name() method which reduces the amount of code:

Example

```
<?php
class Fruit {
  public $name;
  public $color;

  function __construct($name) {
    $this->name = $name;
  }
  function get_name() {
    return $this->name;
  }
}

$apple = new Fruit("Apple");
```

```
echo $apple->get_name();
?>
```

Will give apple

PHP - The __destruct Function

A destructor is called when the object is destructed or the script is stopped or exited.

If you create a __destruct() function, PHP will automatically call this function at the end of the script.Notice that the destruct function starts with two underscores (__)!

The example below has a __construct() function that is automatically called when you create an object from a class, and a __destruct() function that is automatically called at the end of the script:

Example

```
<?php
class Fruit {
  public $name;
  public $color;

  function __construct($name) {
    $this->name = $name;
  }
  function __destruct() {
    echo "The fruit is {$this->name}.";
  }
}

$apple = new Fruit("Apple");
?>
```

Will give

```
The fruit is Apple.
```

String Handling In PHP

In PHP, when we use pieces of text, it is called string. It is one of the data types in PHP. String may contain letters, numbers, punctuations, spaces, tabs or any other type of character. In PHP strings are enclosed in double quotes or in single quotes.

1. Getting length of a String

PHP has a predefined function to get the length of a string. Strlen() displays the length of any string.

```
$a="hai";

$l=strlen(a);
```

value of l is 3

2. Counting of the number of words in a String

Another function which enables display of the number of words in any specific string is str_word_count().

$a="I love my india";

&c=str_word_count($a);

value of c is 4

3. Reversing a String

Strrev() is used for reversing a string. You can use this function to get the reverse version of any string.

$a="abhi";

strrev($a);

now $a become ihba

4. Finding Text Within a String

Strpos() enables searching particular text within a string. It works simply by matching the specific text in a string. If found, then it returns the specific position. If not found at all, then it will return “False”.

$a=strpos(“Welcome to Cloudways”,”Cloudways”);

$ahae value 11

5. Replacing text within a string

Str_replace() is a built-in function, basically used for replacing specific text within a string.

Syntax

Str_replace(string to be replaced,text,string)

Example

str_replace(“cloudways”, “the programming world”, “Welcome to cloudways”);

Output

Welcome to the programming world

6. Converting lowercase into Title Case

Ucwords() is used to convert first alphabet of every word into uppercase.

Syntax

Ucwords(string)

Example

ucwords(“welcome to the php world”);

Output

Welcome To The Php World

7. Converting a whole string into UPPERCASE

Strtoupper() is used to convert a whole string into uppercase.

Syntax

Strtoupper(string);

Example

strtoupper(“welcome to cloudways”);// It will convert all letters of string into uppercase

Output

WELCOME TO CLOUDWAYS

8. Converting whole String to lowercase

Strtolower() is used to convert a string into lowercase.

Syntax

Strtolower(string)

Example

strtolower(“WELCOME TO CLOUDWAYS”);

Output

welcome to cloudways

9. Comparing Strings

You can compare two strings by using strcmp(). It returns output either greater than zero, less than zero or equal to zero. If string 1 is greater than string 2 then it returns greater than zero. If string 1 is less than string 2 then it returns less than zero. It returns zero, if the strings are equal.

Syntax

Strcmp(string1,string2)

Example

```
<?php
echo strcmp(“Cloudways”,”CLOUDWAYS”);
echo “<br>”;
echo strcmp(“cloudways”,”cloudways”);//Both the strings are equal
echo “<br>”;
echo strcmp(“Cloudways”,”Hosting”);
echo “<br>”;
echo strcmp(“a”,”b”);//compares alphabetically
echo “<br>”;
echo strcmp(“abb baa”,”abb baa caa”);//compares both strings and returns the result in terms of
number of characters.
?>
```

Output

```
1
0
-1
-1
-4
```

10. Displaying part of String

Through substr() function you can display or extract a string from a particular position.

Syntax

substr(string,start,length)

Example

```
<?php
echo substr(“Welcome to Cloudways”,6).”<br>”;
echo substr(“Welcome to Cloudways”,0,10).”<br>”;
?>
```

Output

```
e to Cloudways
Welcome to
```

12. Removing white spaces from a String

Trim() is dedicated to remove white spaces and predefined characters from a both the sides of a string.

Syntax

trim(string,charlist)

Example

```
<?php
$str = “Wordpess Hosting”;
echo $str . “<br>”;
echo trim(“$str”,”Wording”);
?>
```

DATE Function in PHP

The PHP date() function return the current date and time according to the built-in clock of the web server on which the script has been executed.

```
<?php
```

```
echo date("d/m/Y") ;

echo date("d-m-Y") ;

echo date("d.m.Y");

?>
```

output

```
26/11/2019

26-11-2019

26.11.2019
```

PHP time() Function:

The time() function is used to get the current time as a Unix timestamp (the number of seconds since the beginning of the Unix epoch: January 1, 1970, 00:00:00 GMT).

.

FORM Tag in PHP

A form is an HTML tag that contains graphical user interface items such as input box, check boxes radio buttons etc.The form is defined using the <form>...</form> tags and GUI items are defined using form elements such as input.

When and why we are using forms?

- Forms come in handy when developing flexible and dynamic applications that accept user input.

Create a form

We will use HTML tags to create a form. Below is the minimal list of things you need to create a form.

- Opening and closing form tags <form>…</form>
- Form submission type POST or GET
- Submission URL that will process the submitted data
- Input fields such as input boxes, text areas, buttons,checkboxes etc.

```
<form action="registration_form.php" method="POST"> First name:

        <input type="text" name="t1"> <br> Last name:

        <input type="text" name="t2">

        <input type="submit" value="Submit">

</form>
```

Submitting the form data to the server

The action attribute of the form specifies the submission URL that processes the data. The method attribute specifies the submission type.

POST method

- This is the built in PHP super global array variable that is used to get values submitted via HTTPPOST method.
- The array variable can be accessed from any script in the program; it has a global scope.
- This method is ideal when you do not want to display the form post values in the URL.

- A good example of using post method is when submitting login details to the server.

It has the following syntax.

```
<?php
            $_POST['variable_name'];
?>
```

HERE,

- “$_POST[…]” is the PHP array
- “'variable_name'” is the URL variable name.

PHP GET method

- This is the built in PHP super global array variable that is used to get values submitted via HTTPGET method.
- The array variable can be accessed from any script in the program; it has a global scope.
- This method displays the form values in the URL.
- It’s ideal for search engine forms as it allows the users to book mark the results.

It has the following syntax.

```
<?php
$_GET['variable_name'];
?>
```

HERE,

- “$_GET[…]” is the PHP array
- “'variable_name'” is the URL variable name.

GET vs POST Methods

POST	GET
Values not visible in the URL	Values visible in the URL

Has not limitation of the length of the values since they are submitted via the body of HTTP	Has limitation on the length of the values usually 255 characters. This is because the values are displayed in the URL. Note the upper limit of the characters is dependent on the browser.
Has lower performance compared to Php_GET method due to time spent encapsulation the Php_POST values in the HTTP body	Has high performance compared to POST method dues to the simple nature of appending the values in the URL.
Supports many different data types such as string, numeric, binary etc.	Supports only string data types because the values are displayed in the URL
Results cannot be book marked	Results can be book marked due to the visibility of the values in the URL

How to make a redirect in PHP?

Redirection from one page to another in PHP is commonly achieved using the following way:

Using Header Function in PHP:

The header() function is an inbuilt function in PHP which is used to send the raw HTTP (Hyper Text Transfer Protocol) header to the client.

Syntax:

```
header( $header, $replace, $http_response_code )
```

Parameters: This function accepts three parameters as mentioned above and described below:

- **$header:** This parameter is used to hold the header string.
- **$replace:** This parameter is used to hold the replace parameter which indicates the header should replace a previous similar header, or add a second header of the same type. It is optional parameter.
- **$http_response_code:** This parameter hold the HTTP response code.

Below program illustrates the header() function in PHP:

Program:

```
<?php

header("Location: http://www.geeksforgeeks.org");

?>
```

Combining HTML & PHP code on a single page

We can include PHP code inside HTML code without redirecting in a separate page. Such a combination can be useful if you need to present the same form to the user more than once.

First you need to set your action in the form field to blank or the name of the current file so either a POST or GET is sent to the same file.

So when you click on submit a request will be sent to the same file where the html is, we need to handle the request, the easiest way to do it is to add an isset($_POST){} to the top of the file. Your logic will go inside this is set.

Here is the code snippet:

```
<?php

if (isset($_POST["t1"]))

{
        echo $_POST["t1"];
}
?>
<html>
<body>
<form action="" method="post">
        <input type="text" name="t1" />
        <input type="submit" value="submit" />
</form>
</body>
</html>
```

The Program will print the name that given to the text box

Creating A Send Mail Form

PHP mail() is the built in PHP function that is used to send emails from PHP scripts.

The mail function accepts the following format

mail("$to", $subject, $message, $headers);

- $to variable is to store reciever’s email id.
- $subject is a variable to store mail subject.
- $message is a variable to store user’s message.
- $headers contains other email parameters like BCc, Cc etc.

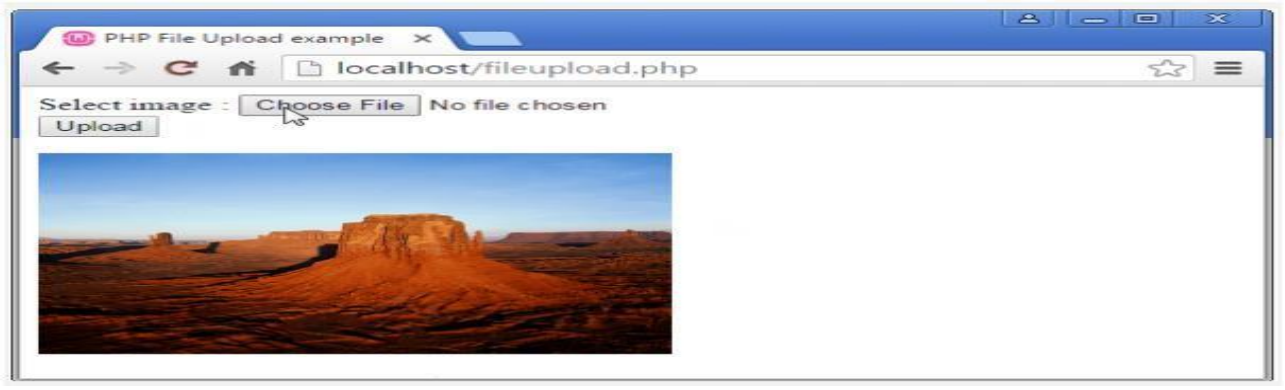
It’s a cost effective way of notifying users on important events.Let users contact you via email by providing a contact us form on the website that emails the provided content.Developers can use it to receive system errors by emailYou can use it to email your newsletter subscribers.

You can use it to send password reset links to users who forget their passwords
You can use it to email activation/confirmation links. This is useful when registering users and verifyingtheir email addresses.

Uploading and display Image in PHP

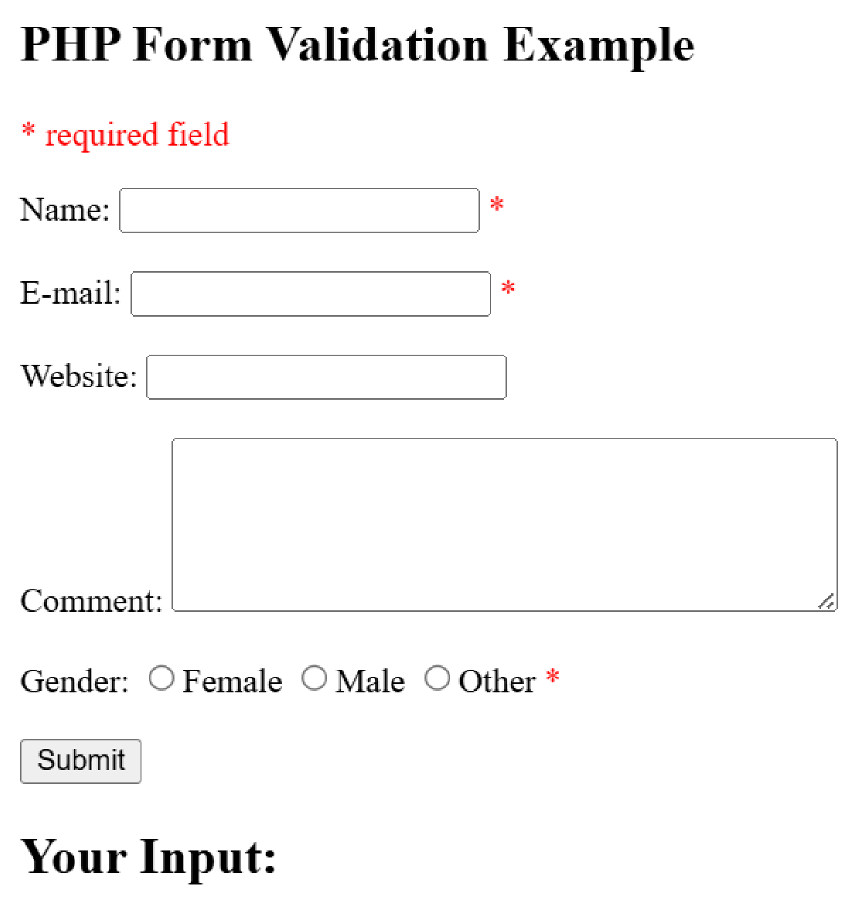
In this php tutorial example we do upload image using file upload and display image on same page in img control after uploading image.

```
<html>
<head>
<title>PHP File Upload example</title>
</head>
<body>

<form action="fileupload.php" method="post">
Select image :
<input type="file" name="file"><br/>
<input type="submit" value="Upload" name="Submit1"> <br/>
</form>
 <?php
if(isset($_POST['Submit1']))
{
$filepath = "images/" . $_FILES["file"]["name"];

if(move_uploaded_file($_FILES["file"]["tmp_name"], $filepath))
{
echo "<img src=".$filepath." height=200 width=300 />";
}
else
{
echo "Error !!";
}
}
?>

</body>
</html>
```



Form Validation

```
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>

<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
    // check if name only contains letters and whitespace
    if (!preg_match("/^[a-zA-Z-' ]*$/",$name)) {
      $nameErr = "Only letters and white space allowed";
    }
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
    // check if e-mail address is well-formed
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
      $emailErr = "Invalid email format";
    }
  }

  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
    // check if URL address syntax is valid (this regular expression also allows dashes in
the URL)
    if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-
9+&@#\/%=~_|]/i",$website)) {
      $websiteErr = "Invalid URL";
    }
  }

  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);
  }

  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>

<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field</span></p>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"
]);?>">
  Name: <input type="text" name="name" value="<?php echo $name;?>">
  <span class="error">* <?php echo $nameErr;?></span>
  <br><br>
  E-mail: <input type="text" name="email" value="<?php echo $email;?>">
  <span class="error">* <?php echo $emailErr;?></span>
  <br><br>
  Website: <input type="text" name="website" value="<?php echo $website;?>">
  <span class="error"><?php echo $websiteErr;?></span>
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40"><?php echo $comment;?
></textarea>
  <br><br>
  Gender:
  <input type="radio" name="gender" <?php if (isset($gender) &&
$gender=="female") echo "checked";?> value="female">Female
  <input type="radio" name="gender" <?php if (isset($gender) &&
$gender=="male") echo "checked";?> value="male">Male
  <input type="radio" name="gender" <?php if (isset($gender) &&
$gender=="other") echo "checked";?> value="other">Other
  <span class="error">* <?php echo $genderErr;?></span>
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>
</body>
</html>
```



PHP Form Validation Example

* required field

Name: *

E-mail: *

Website:

Comment:

Gender: ○ Female ○ Male ○ Other *

Submit

Your Input:

Cookies in PHP

- A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.
- Cookies are text files stored on the client computer and they are kept of use tracking purpose.
- Server script sends a set of cookies to the browser. For example name, age, or identification number etc.
- Browser stores this information on local machine for future use.
- When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.

There are 2 types of cookies:

(1) Session Based which expire at the end of the session.
(2) Persistent cookies which are written on hard disk.

Session cookies expire at the end of the session. This means, when you close your browser window, the session cookie is deleted. This website only uses session cookies. This type of cookies are temporary and are expire as soon as the session ends or the browser is closed.

Persistent cookies do not expire at the end of the session. It will retain its value at end of the time you specified. To make a cookie persistent we must provide it with an expiration time. Then the cookie will only expire after the given expiration time, until then it will be a valid cookie.

Create/Setting Cookies with PHP

A cookie is created with the setcookie() function.

Syntax : *setcookie(name, value, expire, path, domain, security);*

1. **Name** − This sets the name of the cookie and is stored in an environment variable called HTTP_COOKIE_VARS. This variable is used while accessing cookies.
2. **Value** − This sets the value of the named variable and is the content that you actually want to store.
3. **Expiry** − This specify a future time in seconds .After this time cookie will become inaccessible. Ifthis parameter is not set then cookie will automatically expire when the Web Browser is closed.
4. **Path** − This specifies the directories for which the cookie is valid
5. **Domain** − This can be used to specify the domain name in very large domains and must contain at least two periods to be valid. All cookies are only valid for the host and domain which created them.
6. **Security** − This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which mean cookie can be sent by regular HTTP.

Example:

```
<?php
  setcookie("name", "John Watkin", time()+3600, "/","", 0);
?>
```

Accessing Cookies with PHP

- PHP provides many ways to access cookies.
- Simplest way is to use either $_COOKIE or $HTTP_COOKIE_VARS variables.

Syntax:

```
$_COOKIE[“cookie _name”];
Or
$HTTP_COOKIE_VARS[“cookie_name”];
```

Example:

```
<?php
      echo $_COOKIE["name"]. "<br
          />";or
       echo $HTTP_COOKIE_VARS["name"]. "<br />";
   ?>
```

isset() Method in cookie

You can use isset() function to check if a cookie is set or not

```
<?php

      if( isset($_COOKIE["name"]))
         echo "Welcome " . $_COOKIE["name"] . "<br
      />";else
         echo "Sorry... Not recognized" . "<br />";
?>
```

Deleting Cookie with PHP

There is no special dedicated function provided in PHP to delete a cookie. All we have to do is to update the expire-time value of the cookie by setting it to a past time using the setcookie() function. A very simple way of doing this is to deduct a few seconds from the current time.

setcookie("gfg", "", time() - 3600);

Program:

```
<?php
// set the expiration date to one hour ago
setcookie("user", "", time() - 3600);
?>
<html>
<body>
<?php
echo "Cookie 'user' is deleted.";
?>
</body>
</html>
```

PHP program to creating and retrieving cookies

```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
?>
<html>
<body>
<?php
if(!isset($_COOKIE[$cookie_name])) {
  echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
  echo "Cookie '" . $cookie_name . "' is set!<br>";
  echo "Value is: " . $_COOKIE[$cookie_name];
}
?>
</body>
</html>
```

Modify a Cookie Value

To modify a cookie, just set (again) the cookie using the setcookie() function:
Example

```
<?php
$cookie_name = "user";
$cookie_value = "Alex Porter";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>
<html>
<body>
<?php
if(!isset($_COOKIE[$cookie_name])) {
  echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
  echo "Cookie '" . $cookie_name . "' is set!<br>";
  echo "Value is: " . $_COOKIE[$cookie_name];
}
?>
</body>
</html>
```

Check if Cookies are Enabled

The following example creates a small script that checks whether cookies are enabled. First, try to create a test cookie with the setcookie() function, then count the $_COOKIE array variable:

Example

```
<?php
setcookie("test_cookie", "test", time() + 3600, '/');
?>
<html>
<body>
<?php
if(count($_COOKIE) > 0) {
  echo "Cookies are enabled.";
} else {
  echo "Cookies are disabled.";
}
?>

</body>
</html>
```

Session in PHP

- Session is a way to store user variable and access it in multiple pages in a single application.
- Unlike a cookie, the information is not stored on the users computer.
- A session creates a file in a temporary directory on the server where registered session variables and their values are stored. This data will be available to all pages on the site during that visit

Starting a PHP Session

Before you can store any information in session variables, you must first start up the session. To begin a new session, simply call the PHP ***session_start()*** function. It will create a new session and generate a unique session ID for the user.

```
<?php
session_start();
?>
```

Storing Session Data

You can store all your session data as key-value pairs in the $_SESSION[] array. The stored data can beaccessed during lifetime of a session.

Syntax:

```
$_SESSION["session_name"]=value;
```

Example:

```
$_SESSION["firstname"] = "Kiran";
```

Program :

```
<?php
// Start the session
session_start();
?>
<html>
<body>

<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>
</html>
```

Accessing Session Data

Session variable can be accessed in any ware within the application using name of the session.

Syntax:

$_SESSION[“session_name”]

Example:

Page1: *$_SESSION[“fname”]=”Priya”;*
Page2: echo “User name is”. *$_SESSION["fname"]* ;

Modify a PHP Session Variable

To change a session variable, just overwrite it:

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// to change a session variable, just overwrite it
$_SESSION["favcolor"] = "yellow";
print_r($_SESSION);
?>

</body>
</html>
```

Destroying a Session

If you want to remove certain session data, simply unset the corresponding key of the $_SESSION associative array. The session_unset() function frees all session variables currently registered.

Syntax:

unset($_SESSION["session_name"]);

Example:

unset($_SESSION["fname"]);

To remove all global session variables and destroy the session, use session_unset() and session_destroy():

```
Example
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// remove all session variables
session_unset();

// destroy the session
session_destroy();
?>

</body>
</html>
```

Passing Session IDs in the Query String

We can pass session variable through query string (script request to another page or server. This can be achieved by passing the session ID from script to script embedded in a query string. PHP makes a name/value pair available in a constant called SID if a cookie value for a session ID cannot be found. Youcan add this string to any HTML links in session-enabled pages:

```
<a href="anotherpage.html?<?php print SID; ?>">Another page</a>
```

It will reach the browser as

```
<a href="anotherpage.html?PHPSESSID=08ecedf791">Anotherpage</a>
```

The session ID passed in this way will automatically be recognized in the target page when session_start() is called, and you will have access to session variables in the usual way.

Database concepts

- MySQL is an open-source relational database management system (RDBMS). It is the most popular database system used with PHP. MySQL is developed, distributed, and supported by Oracle Corporation.
- The data in a MySQL database are stored in tables which consists of columns and rows.
- MySQL is a database system that runs on a server.
- MySQL is ideal for both small and large applications.
- MySQL is very fast, reliable, and easy to use database system.It uses standard SQL
- MySQL compiles on a number of platforms.

MySQL Features

1. **Relational Database Management System (RDBMS):** MySQL is a relational database management system.
2. **Easy to use:** MySQL is easy to use. You have to get only the basic knowledge of SQL. You can build and interact with MySQL with only a few simple SQL statements.
3. **It is secure:** MySQL consist of a solid data security layer that protects sensitive data from intruders. Passwords are encrypted in MySQL.
4. **Client/ Server Architecture:** MySQL follows a client /server architecture. There is a database server (MySQL) and arbitrarily many clients (application programs), which communicate with the server; that is, they query data, save changes, etc.
5. **Free to download:** MySQL is free to use and you can download it from MySQL official website.
6. **It is scalable:** MySQL can handle almost any amount of data, up to as much as 50 million rows or more. The default file size limit is about 4 GB. However, you can increase this number to a theoretical limit of 8 TB of data.
7. **Compatibale on many operating systems:** MySQL is compatible to run on many operating systems, like Novell NetWare, Windows* Linux*, many varieties of UNIX* (such as Sun* Solaris*, AIX, and DEC* UNIX), OS/2, FreeBSD*, and others. MySQL also provides a facility that the clients can run on the same computer as the server or on another computer (communication via a local network or the Internet).
8. **High Performance:** MySQL is faster, more reliable and cheaper because of its unique storage engine architecture.
9. **High Flexibility:** MySQL supports a large number of embedded applications which makes MySQL very flexible.
10. **High Productivity:** MySQL uses Triggers, Stored procedures and views which allows the developer to give a higher productivity.

Disadvantages / Drawback of MySQL:
Following are the few disadvantages of MySQL:

- MySQL version less than 5.0 doesn't support ROLE, COMMIT and stored procedure.
- MySQL does not support a very large database size as efficiently.
- MySQL doesn't handle transactions very efficiently and it is prone to data corruption.
- MySQL is accused that it doesn't have a good developing and debugging tool compared to paid databases.
- MySQL doesn't support SQL check constraints.

MySQL Data Types

A database table contains multiple columns with specific data types such as numeric or string. MySQL provides more data types other than just numeric or string. Each data type in MySQL can be determined by the following characteristics:

- The kind of values it represents.
- The space that takes up and whether the values is a fixed-length or variable length.
- The values of the data type can be indexed or not.
- How MySQL compares the values of a specific data type.

MySQL uses many different data types broken into three categories −

1. Numeric
2. Date and Time
3. String Types.

1.Numeric Data Types

The following list shows the common numeric data types and their descriptions −

INT − A normal-sized integer that can be signed or unsignedYou can specify a width of up to 11 digits.

TINYINT − A very small integer that can be signed or unsigned. If signed, the allowable range is from -128 to 127. If unsigned, the allowable range is from 0 to 255. You can specify a width of up to 4 digits.

SMALLINT − A small integer that can be signed or unsigned. You can specify a width of up to 5 digits.

MEDIUMINT − A medium-sized integer that can be signed or unsigned You can specify a width of up to 9 digits.

BIGINT − A large integer that can be signed or unsigned You can specify a width of up to 20 digits.

FLOAT(M,D) − A floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 10,2, where 2 is the number of decimals and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a FLOAT.

DOUBLE(M,D) − A double precision floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a DOUBLE. REAL is a synonym for DOUBLE.

DECIMAL(M,D) − An unpacked floating-point number that cannot be unsigned. In the unpacked decimals, each decimal corresponds to one byte. Defining the display length (M) and the number of decimals (D) is required. NUMERIC is a synonym for DECIMAL.
Date and Time Types

2. Date and Time

The MySQL date and time datatypes are as follows −

DATE − A date in YYYY-MM-DD format For example, December 30th, 1973 would be stored as 1973-12-30.

DATETIME − A date and time combination in YYYY-MM-DD HH:MM:SS format,. For example, 3:30 in the afternoon on December 30th, 1973 would be stored as 1973-12-30 15:30:00.

TIMESTAMP − A timestamp between midnight, January 1st, 1970 and sometime in 2037.
TIME − Stores the time in a HH:MM:SS format.

YEAR(M) − Stores a year in a 2-digit or a 4-digit format. If the length is specified as 2 (for example YEAR(2)), YEAR can be between 1970 to 2069 (70 to 69). If the length is specified as 4, then YEAR can be 1901 to 2155. The default length is 4.

3. String Types

Although the numeric and date types are fun, most data you'll store will be in a string format. This list describes the common string datatypes in MySQL.

CHAR(M) − A fixed-length string between 1 and 255 characters in length (for example CHAR(5)).
VARCHAR(M) − A variable-length string between 1 and 255 characters in length. For example, VARCHAR(25). You must define a length when creating a VARCHAR field.
BLOB or TEXT − A field with a maximum length of 65535 characters. BLOBs are "Binary Large Objects" and are used to store large amounts of binary data, such as images or other types of files. Fields defined as TEXT also hold large amounts of data. The difference between the two is that the sorts and comparisons on the stored data are case sensitive on BLOBs and are not case sensitive in TEXT fields. You do not specify a length with BLOB or TEXT.

TINYBLOB or TINYTEXT – A BLOB or TEXT column with a maximum length of 255 characters. You do not specify a length with TINYBLOB or TINYTEXT.

MEDIUMBLOB or MEDIUMTEXT − A BLOB or TEXT column with a maximum length of 16777215 characters. You do not specify a length with MEDIUMBLOB or MEDIUMTEXT.

LONGBLOB or LONGTEXT − A BLOB or TEXT column with a maximum length of 4294967295 characters. You do not specify a length with LONGBLOB or LONGTEXT.

ENUM – An enumeration, which is a fancy term for list. When defining an ENUM, you are creating a list of items from which the value must be selected (or it can be NULL). For example, if you wanted your field to contain "A" or "B" or "C", you would define your ENUM as ENUM ('A', 'B', 'C') and only those values (or NULL) could ever populate that field.

Connecting PHP page to MySQL

Before perfoming any database operation we must connect database with php page ,for that we can use **mysqli ()** function and can create an object of connection string.This object is used for database operation

Syntax:
```
 $connectionstring = new mysqli($servername, $username, $password, $databasename);
```
Example:
```
 $con = new mysqli(“localhost”,”macas”,”123”,”MyDatabase”);
```
Con is the connection string

PHP MySQL Create Table

- A database table has its own unique name and consists of columns and rows.
- The CREATE TABLE statement is used to create a table in MySQL.

Syntax:

$querystring=CREATE TABLE tablename(field1 datatype,field2 datatype, ...);

Example:

```
$sql = "CREATE TABLE student( ". "rollno INT, "."name VARCHAR(100) NOT NULL, "."class VARCHAR(40) NOT NULL, . "PRIMARY KEY ( rollno )); ";
mysql_query( $con, $sql);
```

PHP MySQL INSERT Query

- The INSERT INTO statement is used to insert new rows in a database table.
- We can use INSERT INTO statement with appropriate values to insert values in the table , after that we will execute this insert query through passing it to the PHP mysqli_query() function to insert data in table.

Syntax:

```
$query_String=INSERT INTO student(rollno, name,class) VALUES ('value1','value2',value3')";
```

Example:

```
$sql = "INSERT INTO persons (rollno, name,class) VALUES ('1', 'Abhi', 'MCA')";
$sql = "INSERT INTO persons (rollno, name,class) VALUES ('2', 'Priya', 'MTech')";
mysqli_query($con, $sql);
```

Selecting Data From Database Tables

The SQL SELECT statement is used to select the records from database tables
Syntax:

```
SELECT column1_name, column2_name, columnN_name FROM table_name;
```

Example: This example shows selecting rows from database and display the content in a table.

```
$sql = "SELECT * FROM student";
if($result = mysqli_query($con, $sql))
{
  if(mysqli_num_rows($result) > 0)
  {
    echo "<table>";
      echo "<tr>";
        echo "<th>Rollno</th>";
        echo "<th>Name</th>";
        echo "<th>Class</th>";
        echo "</tr>";
    while($row = mysqli_fetch_array($result)
     {
      echo "<tr>";
        echo "<td>" . $row['rollno'] . "</td>";
        echo "<td>" . $row['name'] . "</td>";
        echo "<td>" . $row['class'] . "</td>";
      echo "</tr>";
```

```
        }
        echo "</table>";
    }
```

PHP MySQL WHERE Clause

The WHERE clause is used to extract only those records that fulfill a specified condition.
The basic syntax of the WHERE clause can be given with:

Syntax:

```
SELECT column_name(s) FROM table_name WHERE condition;
```

Example: This example shows details of student whose roll number is 2

```
$sql = "SELECT * FROM student WHERE rollno='1'";
if($result = mysqli_query($con, $sql))
{
    if(mysqli_num_rows($result) > 0)
    {
        echo "<table>";
            echo "<tr>";
                echo "<th>Rollno</th>";
                echo "<th>Name</th>";
                echo "<th>Class</th>";
                echo "</tr>";
        while($row = mysqli_fetch_array($result)
         {
            echo "<tr>";
                echo "<td>" . $row['rollno'] . "</td>";
                echo "<td>" . $row['name'] . "</td>";
                echo "<td>" . $row['class'] . "</td>";
            echo "</tr>";
        }
        echo "</table>";
      }
```

Ordering the Result Set

The ORDER BY clause can be used in conjugation with the SELECT statement to see the data from a table ordered by a specific field. The ORDER BY clause lets you define the field name to sort against and the sort direction either ascending or descending.

Syntax:

```
SELECT column_name(s) FROM table_name ORDER BY column_name(s) ASC|DESC
```

Example: This example shows details of student in ascending order of name

```
$sql = "SELECT * FROM student ORDER BY name ASC";
if($result = mysqli_query($con, $sql))
{
  if(mysqli_num_rows($result) > 0)
  {
    echo "<table>";
      echo "<tr>";
        echo "<th>Rollno</th>";
        echo "<th>Name</th>";
        echo "<th>Class</th>";
        echo "</tr>";
    while($row = mysqli_fetch_array($result)
     {
      echo "<tr>";
        echo "<td>" . $row['rollno'] . "</td>";
        echo "<td>" . $row['name'] . "</td>";
        echo "<td>" . $row['class'] . "</td>";
      echo "</tr>";
    }
    echo "</table>";
  }
```

MySQL LIKE operator

- The LIKE operator is a logical operator that tests whether a string contains a specified pattern or not. Here is the syntax of the LIKE operator:
- The LIKE operator is used in the WHERE clause of the SELECT , DELETE, and UPDATE statements to filter data based on patterns.
- MySQL provides two wildcard characters for constructing patterns: percentage % and underscore _ .
- The percentage (%) wildcard matches any string of zero or more characters.
- The underscore (_) wildcard matches any single character.

For example, s% matches any string starts with the character s such as sun and six. The se_ matches any string starts with se and is followed by any character such as see and sea

Syntax:

```
SELECT column_name(s) FROM table_name LIKE  pattern
```

Example: This example shows details of students whose name starts with letter p

```
$sql = "SELECT * FROM student LIKE ‘a%’";
if($result = mysqli_query($con, $sql))
{
   if(mysqli_num_rows($result) > 0)
   {
      echo "<table>";
         echo "<tr>";
            echo "<th>Rollno</th>";
            echo "<th>Name</th>";
            echo "<th>Class</th>";
            echo "</tr>";
      while($row = mysqli_fetch_array($result)
       {
         echo "<tr>";
            echo "<td>" . $row['rollno'] . "</td>";
            echo "<td>" . $row['name'] . "</td>";
            echo "<td>" . $row['class'] . "</td>";
         echo "</tr>";
      }
      echo "</table>"; }
```

Selecting Multiple tables: using join

MySQL JOINS can be used with SELECT statement. It is used to retrieve data from multiple tables. It is performed whenever you need to fetch records from two or more tables.

There are three types of MySQL joins:

1. MySQL INNER JOIN (or sometimes called simple join)
2. MySQL LEFT OUTER JOIN (or sometimes called LEFT JOIN)
3. MySQL RIGHT OUTER JOIN (or sometimes called RIGHT JOIN)

1. MySQL Inner JOIN (Simple Join)

The MySQL INNER JOIN is used to return all rows from multiple tables where the join condition is satisfied. It is the most common type of join.

Syntax:

```
SELECT columns  FROM table1  INNER JOIN table2  ON table1.column = table2.column;
```

Example:

Consider two tables "officers" and "students", having the following data.

officers

officer_id	**officer_name**	**address**
1	Abhi	TVM
2	Anu	KLM
3	Kiran	EKM
4	Priya	TVM

students

student_id	**student_name**	**course_name**
1	Suma	MCA
2	Ajin	MSc
3	Lallu	BSc

Example of Inner join :

SELECT officers.officer_name, officers.address, students.course_name FROM officers
INNER JOIN students ON officers.officer_id = students.student_id;

officer_name	**address**	**course_name**
Abhi	TVM	MCA
Anu	KLM	MSc
Kiran	EKM	BSc

2. MySQL Left Outer Join

The LEFT OUTER JOIN returns all rows from the left hand table specified in the ON condition and only those rows from the other table where the join condition is fulfilled.

Syntax:

SELECT columns FROM table1 LEFT [OUTER] JOIN table2 ON table1.column = table2.column;

Example of Left join:

SELECT officers.officer_name, officers.address, students.course_name FROM officers
LEFT JOIN students ON officers.officer_id = students.student_id;

officer_name	**address**	**course_name**
Abhi	TVM	MCA
Anu	KLM	MSc
Kiran	EKM	BSc
Priya	TVM	NULL

3. MySQL Right Outer Join

The MySQL Right Outer Join returns all rows from the RIGHT-hand table specified in the ON condition and only those rows from the other table where he join condition is fulfilled.

Syntax:

SELECT columns FROM table1 RIGHT [OUTER] JOIN table2 ON table1.column = table2.column;

Example of Right join:

SELECT officers.officer_name, officers.address, students.course_name, FROM officers RIGHT JOIN students ON officers.officer_id = students.student_id;

officer_name	**address**	**course_name**
Abhi	TVM	MCA
Anu	KLM	MSc
Kiran	EKM	BSc

Modifying Records

1. PHP MySQL Update Command

- The MySQL UPDATE query is used to update existing records in a table in a MySQL database.
- It can be used to update one or more field at the same time.
- It can be used to specify any condition using the WHERE clause

Syntax1 :

UPDATE table_name SET column1=value, column2=value2

Syntax1 :

UPDATE table_name SET column1=value, column2=value2 WHERE some_column=some_value ;

Example1:

$sql = "UPDATE students SET course='BSc' ";

Example1:

$sql = "UPDATE students SET course='BSc' WHERE student_id=2";

2. PHP MySQL Replace Command

The REPLACE() function replaces all occurrences of a substring within a string, with a new substring.

Syntax:

REPLACE(string, from_string, new_string)

Example:

SELECT REPLACE("XYZ FGH XYZ", "X", "M");

It will Replace "X" with "M":

3. PHP MySQL Delete Command

The DELETE statement is used to delete records from a table:

Syntax:

```
DELETE FROM table_name WHERE some_column = some_value;
```

Example:

```
$sql = "DELETE FROM students WHERE student_id=3";
```

Date Time Functions in PHP MySQL

1. CURDATE()

The CURDATE() function returns the current date as a value in the 'YYYY-MM-DD' format if it is used in a string context or YYYMMDD format if it is used in a numeric context.
The following example shows how the CURDATE() function is used in the string context.

Format: SELECT CURDATE();

Output: 11-12-2019

2. NOW()

The MySQL NOW() function returns the current date and time in the configured time zone as a string or a number in the 'YYYY-MM-DD HH:MM:DD' format.
The returned type of the NOW() function depends on the context where it is used.

Format: SELECT NOW();

Output: 11-12-2019 12:30:55

3. DAYOFWEEK ()

The DAYOFWEEK function returns the weekday index for a date i.e., 1 for Sunday, 2 for Monday, … 7 for Saturday. These index values correspond to the ODBC standard.

Format: SELECT DAYOFWEEK('2019-12-08');

Output: Sunday

4. DATEDIFF()

The MySQL DATEDIFF function calculates the number of days between two DATE, DATETIME, or TIMESTAMP values.

Format: SELECT DATEDIFF('2011-08-10','2011-08-08');

Output: 2

5. DATE_ADD()

The DATE_ADD function adds an interval to a DATE

Format: SELECT DATE_ADD('1999-12-31',INTERVAL 1 DAY) ;

Output: 2000-01-01

6. DATE_SUB()

The DATE_SUB() function subtracts a time value (or an interval) from a DATE or DATETIME value.

Format: SELECT DATE_SUB('2017-07-04’,INTERVAL 1 DAY) ;
Output: 2017-07-03’

7. LAST_DAY()

The LAST_DAY() function takes a DATE or DATETIME value and returns the last day of the month for theinput date.

Format: LAST_DAY('2016-02-03')
Output: 2016-02-29

8. SYSDATE()

The SYSDATE() function returns the current date and time as a value in 'YYYY-MM-DD HH:MM:SS' format

Format: SELECT SYSDATE();
Output: 2019-12-09 17:42:37

9. EXTRACT()

The EXTRACT() function extracts part of a date from a format 'YYYY-MM-DD HH:MM:SS'.

Format: SELECT EXTRACT(DAY FROM '2019-07-14 09:04:44') DAY;
Output: 14

10. YEAR()

The YEAR() function takes a date argument and returns the year of the date.

Format: SELECT YEAR('2019-07-14 ') ;
Output: 2019