PHP

PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.

PHP is a widely-used, free, and efficient alternative to competitors such as Microsoft's ASP.

Example

```
<!DOCTYPE html>
<html>
<body>
<?php
echo "My first PHP script!";
?>
</body>
</html>
```

Before you continue you should have a basic understanding of the following:

- HTML
- CSS
- JavaScript

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

PHP is an amazing and popular language!

It is powerful enough to be at the core of the biggest blogging system on the web (WordPress)!
It is deep enough to run the largest social network (Facebook)!
It is also easy enough to be a beginner's first server side language!

What is a PHP File?

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code are 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

Why PHP?

- PHP runs on various platforms (Windows, Linux, UNIX, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases
- PHP is free. Download it from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

To start using PHP, you can:

- Find a web host with PHP and MySQL support
- Install a web server on your own PC, and then install PHP and MySQL

Use a Web Host With PHP Support

If your server has activated support for PHP you do not need to do anything.

Just create some .php files, place them in your web directory, and the server will automatically parse them for you.

You do not need to compile anything or install any extra tools.

Because PHP is free, most web hosts offer PHP support.

Set Up PHP on Your Own PC

However, if your server does not support PHP, you must:

- install a web server
- install PHP
- install a database, such as MySQL
- PHP script is executed on the server, and the plain HTML result is sent back to the browser.

Basic PHP Syntax

- A PHP script can be placed anywhere in the document.
- A PHP script starts with **<?php** and ends with **?>**:

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

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

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

Below, we have an example of a simple PHP file, with a PHP script that uses a built-in PHP function "echo" to output the text "Hello World!" on a web page:

Example

```
<!DOCTYPE html>
<html>
<body>

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>
```

PHP Variables

Variables are "containers" for storing information.

Creating (Declaring) PHP Variables

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

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

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

Rules for PHP variables:

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

PHP echo and print Statements

In PHP there are two basic ways to get output: `echo` and `print`.

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

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

PHP Data Types

Variables can store data of different types, and different data types can do different things.

PHP supports the following data types:

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

Get the Length of a String

The PHP strlen() function returns the length of a string.

The example below returns the length of the string "Hello world!":

```
echo strlen("Hello world!"); // outputs 12
```

Count The Number of Words in a String

The PHP str_word_count() function counts the number of words in a string:

```
echo str_word_count("Hello world!"); // outputs 2
```

PHP Constants

A constant is an identifier (name) for a simple value. The value cannot be changed during the script.

A valid constant name starts with a letter or underscore (no $ sign before the constant name).

Note: Unlike variables, constants are automatically global across the entire script.

Create a PHP Constant

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

Syntax

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

```
define("GREETING", "Welcome to W3Schools.com!");
echo GREETING;
```

PHP Operators

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators

- Increment/Decrement operators
- Logical operators
- String operators
- Array operators

PHP Conditional Statements // similar to C

Very often when you write code, you want to perform different actions for different conditions. You can use conditional statements in your code to do this.

In PHP we have the following conditional statements:

- if statement - executes some code if one condition is true
- if...else statement - executes some code if a condition is true and another code if that condition is false
- if...elseif....else statement - executes different codes for more than two conditions
- switch statement - selects one of many blocks of code to be executed

The switch statement is used to perform different actions based on different conditions.

PHP Loops

Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal code-lines in a script, we can use loops to perform a task like this.

In PHP, we have the following looping statements:

- while - loops through a block of code as long as the specified condition is true
- do...while - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- for - loops through a block of code a specified number of times
- foreach - loops through a block of code for each element in an array

The PHP foreach Loop

The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.

```
<?php
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {
   echo "$value <br>";
}
?>
```

PHP User Defined Functions

Besides the built-in PHP functions, we can create our own functions.

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

A function will not execute immediately when a page loads.

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

Create a User Defined Function in PHP

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

Syntax   // similar to c

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

Create an Array in PHP

In PHP, the array() function is used to create an array:

In PHP, there are three types of arrays:

- **Indexed arrays** - Arrays with a numeric index
- **Associative arrays** - Arrays with named keys
- **Multidimensional arrays** - Arrays containing one or more arrays

PHP Indexed Arrays

There are two ways to create indexed arrays:

The index can be assigned automatically (index always starts at 0), like this:

$cars = array("Volvo", "BMW", "Toyota");

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

Get The Length of an Array - The count() Function

The count() function is used to return the length (the number of elements) of an array:

Loop Through an Indexed Array

To loop through and print all the values of an indexed array, you could use a for loop, like this:

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

for($x = 0; $x < $arrlength; $x++) {
   echo $cars[$x];
   echo "<br>";
}
?>
```

PHP Associative Arrays

Associative arrays are arrays that use named keys that you assign to them.

There are two ways to create an associative array:

$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```

Loop Through an Associative Array

To loop through and print all the values of an associative array, you could use a `foreach` loop, like this:

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $x_value) {
   echo "Key=" . $x . ", Value=" . $x_value;
   echo "<br>";
}
?>
```

PHP Global Variables - Superglobals

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

The PHP superglobal variables are:

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_ENV
- $_COOKIE
- $_SESSION

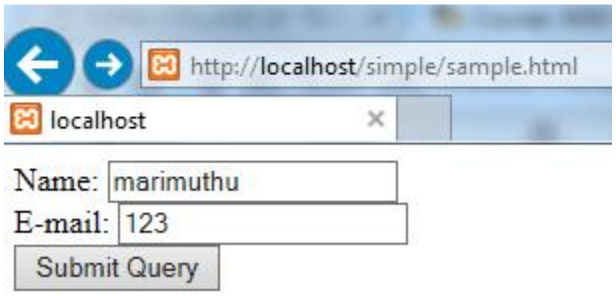
HTML Form Handling

sample.html

```
<html>
<body>
<form action="one.php" method="get">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
</body>
</html>
```

one.php

```
<html>
<body>
Welcome <?php echo $_GET["name"]; ?><br>
Your email address is: <?php echo $_GET["email"]; ?>
</body>
</html>
```



Name: marimuthu
E-mail: 123

```
Welcome marimuthu
Your email address is: 123
```

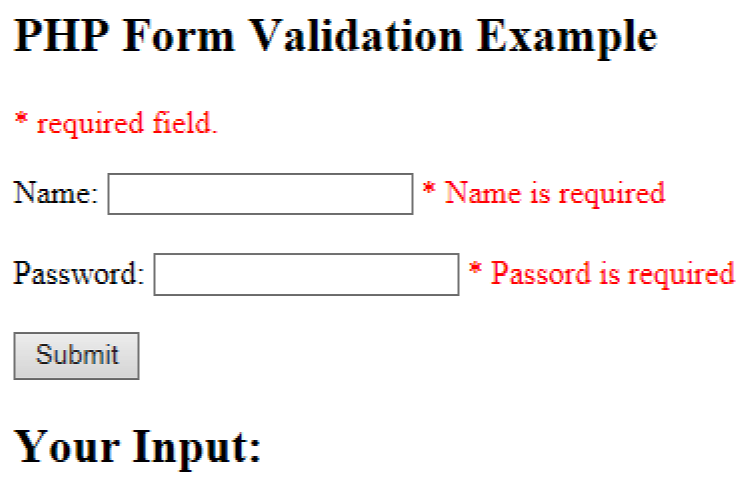
PHP Form Validation

```
<!DOCTYPE HTML>
<html>
<head>
</head>
<body>

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

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = $_POST["name"];
  $email = $_POST["email"];
  $website = $_POST["website"];
  $comment = $_POST["comment"];
  $gender = $_POST["gender"];
}
?>

<h2>PHP Form Validation Example</h2>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  Name: <input type="text" name="name">
  <br><br>
  E-mail: <input type="text" name="email">
  <br><br>
  Website: <input type="text" name="website">
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40"></textarea>
  <br><br>
  Gender:
  <input type="radio" name="gender" value="female">Female
  <input type="radio" name="gender" value="male">Male
  <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>     //output refer complete form validation
```

PHP Form Required

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

<?php
// define variables and set to empty values
$nameErr = $passErr = "";
```

```
$name = $pass = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = $_POST["name"];
  }

  if (empty($_POST["pass"])) {
    $passErr = "Passord is required";
  } else {
    $pass = $_POST["pass"];
  }
}

?>
<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">
  <span class="error">* <?php echo $nameErr;?></span>
  <br><br>
  Password: <input type="text" name="pass">
  <span class="error">* <?php echo $passErr;?></span>
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>
<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $pass;
?>
</body>
</html>
```


PHP Form Validation Example

* required field.

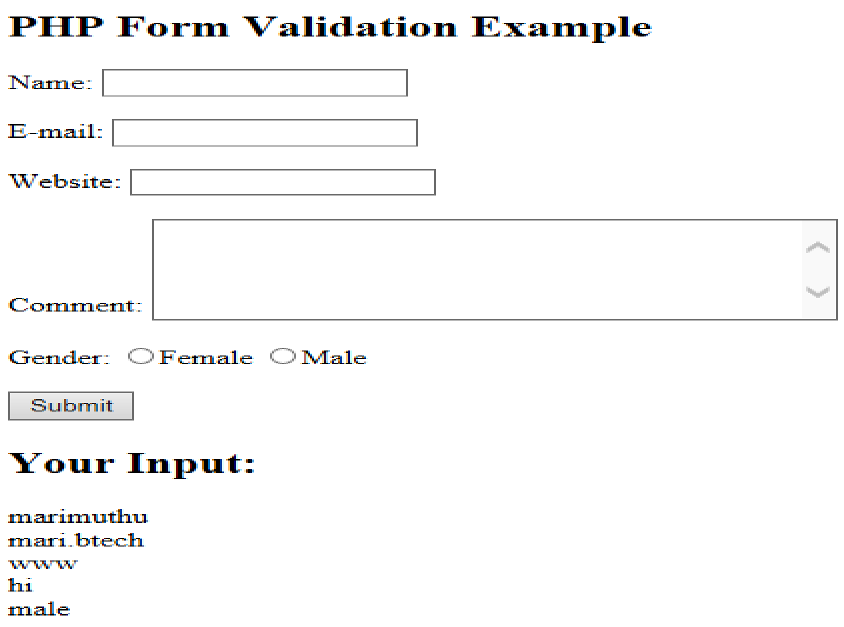
Name: * Name is required

Password: * Passord is required

Your Input:

PHP Complete Form Validation

```
<!DOCTYPE HTML>
<html>
<head>
</head>
<body>

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

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = $_POST["name"];
  $email = $_POST["email"];
  $website = $_POST["website"];
  $comment = $_POST["comment"];
  $gender = $_POST["gender"];
}
?>

<h2>PHP Form Validation Example</h2>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  Name: <input type="text" name="name">
  <br><br>
  E-mail: <input type="text" name="email">
  <br><br>
  Website: <input type="text" name="website">
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40"></textarea>
  <br><br>
  Gender:
  <input type="radio" name="gender" value="female">Female
  <input type="radio" name="gender" value="male">Male
  <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>
```



Your Input:

```
marimuthu
mari.btech
www
hi
male
```

<u>**PHP Database Connectivity**</u>

PHP MySQL Database
With PHP, you can connect to and manipulate databases.
MySQL is the most popular database system used with PHP.
What is MySQL?

- MySQL is a database system used on the web
- 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
- MySQL uses standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use
- MySQL is developed, distributed, and supported by Oracle Corporation
- MySQL is named after co-founder Monty Widenius's daughter: My

The data in a MySQL database are stored in tables. A table is a collection of related data, and it consists of columns and rows.

<u>**PHP Connect to MySQL**</u>

- **MySQLi extension** (the "i" stands for improved)
- **PDO (PHP Data Objects)**

<u>**Should I Use MySQLi or PDO?**</u>
If you need a short answer, it would be "Whatever you like".
Both MySQLi and PDO have their advantages:
PDO will work on 12 different database systems, whereas MySQLi will only work with MySQL databases.
So, if you have to switch your project to use another database, PDO makes the process easy. You only have to change the connection string and a few queries. With MySQLi, you will need to rewrite the entire code - queries included.
Both are object-oriented, but MySQLi also offers a procedural API.
Both support Prepared Statements. Prepared Statements protect from SQL injection, and are very important for web application security.

PHP Database Connectivity

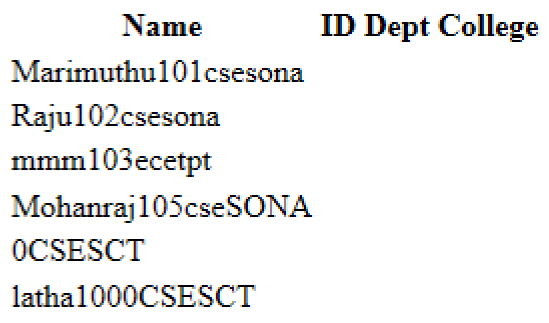
```
<?php
$servername = "localhost:3306";
$username = "root";
$password = "";

// Create connection
$conn = new mysqli($servername, $username, $password);

// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

PHP Database Create & Sample Query for [insert,delete & Update]

```
<?php
$servername = "localhost:3306";
$username = "root";
$password = "";
// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}
// Create database
$sql = "CREATE DATABASE Employee";
if ($conn->query($sql) === TRUE) {
   echo "Database created successfully";
} else {
   echo "Error creating database: " . $conn->error;
}
$conn->close(); // Connection close
?>

/*
Create Table
Insert Quesry
$sql = "INSERT INTO MyGuests (firstname, lastname, email) VALUES ('John', 'Doe',
'john@example.com')";
Insert Multiple Quesry
$sql = "INSERT INTO MyGuests (firstname, lastname, email) VALUES ('John', 'Doe',
'john@example.com');";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email) VALUES ('Mary', 'Moe',
'mary@example.com');";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email) VALUES ('Julie', 'Dooley',
'julie@example.com')";
multi_query($sql)
Delete Query & Update Query
*/
```

PHP – Database select query

```
<!DOCTYPE html>
<html>
<?php
$servername = "localhost:3306";
$username = "root";
$password = "";
$dbname = "studentdata";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT * FROM student";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    echo "<table><tr><th>Name</th><th>ID</th><th>Dept</th><th>College</th></tr>";
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "<tr><td>".$row["sname"].$row["sno"].$row["sdept"].$row["scollege"]."</td><td>";
    }
    echo "</table>";
} else {
    echo "0 results";
}
$conn->close();
?>
```

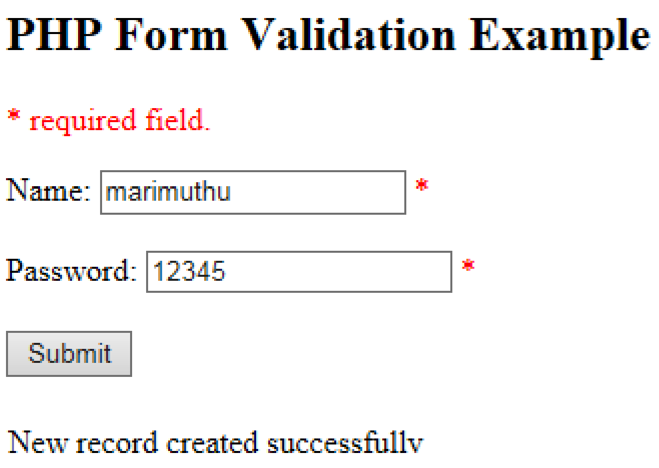


```
          Name          ID Dept College
 Marimuthu101csesona
 Raju102csesona
 mmm103ecetpt
 Mohanraj105cseSONA
 0CSESCT
 latha1000CSESCT
```

PHP – Multi tier Application using Mysqli[object oriented]

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

```
<?php
// define variables and set to empty values
$nameErr = $passErr = "";
$name = $pass = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = $_POST["name"];
  }

  if (empty($_POST["pass"])) {
    $passErr = "Passord is required";
  } else {
    $pass = $_POST["pass"];
  }

}
?>

<?php
$servername = "localhost:3306";
$username = "root";
$password = "";
$dbname = "studentdata";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO student VALUES ('$name','$pass','CSE','SCT')";

if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>


<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">
  <span class="error">* <?php echo $nameErr;?></span>
  <br><br>
  Password: <input type="text" name="pass">
  <span class="error">* <?php echo $passErr;?></span>
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $pass;
?>

</body>
</html>
```



PHP Form Validation Example

* required field.

Name: marimuthu *

Password: 12345 *

New record created successfullv