
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 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.
- PHP supports a wide range of databases
- PHP is free. Download it from the official PHP resource
- PHP runs efficiently on the server side

Basic PHP Syntax

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

Example

```
<!DOCTYPE html>
<html>
    <body>
        <h1>My first PHP page</h1>
        <?php
                echo "Hello World!";
        ?>
    </body>
</html>
```

Comments in PHP

//	- This is a single-line comment
#	- This is also a single-line comment
/* ——————— */	- This is a multiple-lines comment block

PHP Case Sensitivity

- In PHP, NO keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are case-sensitive.
- Eg:
  - *ECHO "Hello Manipal*
  - *echo "Hello Manipal";*
  - *EcHo "Hello Manipal";*

PHP Variables

- Variables are "containers" for storing information.
- Creating (Declaring) PHP Variables
  - *All variables in PHP are denoted with a leading dollar sign ($).*
  - *Eg:* <?php

```
$txt = "Hello world!";
$x = 5;
$y = 10.5;

?>
```

- **Note:** When you assign a text value to a variable, put quotes around the value.
- **Note:** Unlike other programming languages, PHP has no command for declaring a variable. It is created the moment you first assign a value to it.

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 Variables(Cont..)

- Output Variables
  - *The PHP echo statement is often used to output data to the screen.*
  - *Eg: <?php*

```
$txt = “Manipal";
echo “Welcome to $txt !";
?>
```

- The below example will produce the same output as the example above:
  - *Eg: <?php*

```
$txt = “Manipal";
echo “Welcome to” . $txt . “!";
?>
```

PHP Variables Scope

- In PHP, variables can be declared anywhere in the script.
- The scope of a variable is the part of the script where the variable can be referenced/used.
- PHP has three different variable scopes:
  - *local*
  - *global*
  - *static*

PHP Variables Scope (cont..)

- A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function
  - *The global keyword is used to access a global variable from within a function.*
  - *To do this, use the global keyword before the variables (inside the function)*
  - PHP also stores all global variables in an array called $GLOBALS[*index*].
  - The *index* holds the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.
- A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function
- Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. To do this, use the static keyword when you first declare the variable

PHP echo and print Statements

- echo and print are more or less the same. They are both used to output data to the screen.
- The differences are small:
  - *echo has no return value*
  - *echo can take multiple parameters (although such usage is rare)*
  - *echo is marginally faster than print.*
  - *print has a return value of 1 so it can be used in expressions.*
  - *print can take one argument.*

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*

PHP Operators

■ Operators are used to operate on values. There are four classifications of operators:

- *Arithmetic*
- *Assignment*
- *Comparison*
- *Logical*

PHP Operators

Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 x+2	4
-	Subtraction	x=2 5-x	3
*	Multiplication	x=4 x*5	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

PHP Operators

Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
.=	x.=y	x=x.y
%=	x%=y	x=x%y

PHP Operators

Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
!=	is not equal	5!=8 returns true
<>	is not equal	5<>8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

PHP Operators

Logical Operators

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
||	or	x=6 y=3 (x==5 || y==5) returns false
!	not	x=6 y=3 !(x==y) returns true

PHP Conditional Statements

- Very often when you write code, you want to perform different actions for different decisions.
- You can use conditional statements in your code to do this.
- In PHP we have the following conditional statements…

PHP Conditional Statements

- **if** statement - use this statement to execute some code only if a specified condition is true
- **if...else** statement - use this statement to execute some code if a condition is true and another code if the condition is false
- **if...elseif....else** statement - use this statement to select one of several blocks of code to be executed
- **switch** statement - use this statement to select one of many blocks of code to be executed

Cont..

- The if statement executes some code if one condition is true.
  - if (*condition*) {
    code to be executed if condition is true;
    }
- The if....else statement executes some code if a condition is true and another code if that condition is false.
  - if (*condition*) {
    code to be executed if condition is true;
    } else {
    code to be executed if condition is false;
    }

Cont..

- The if….elseif…else statement executes different codes for more than two conditions.
  - if (*condition*) {
    code to be executed if this condition is true;
    } elseif (*condition*) {
    code to be executed if this condition is true;
    } else {
    code to be executed if all conditions are false;
    }

Cont..

- Use the switch statement to select one of many blocks of code to be executed.
  - switch (*n*) {
    case *label1:*
    code to be executed if n=label1;
    break;
    case *label2:*
    code to be executed if n=label2;
    break;
    case *label3:*
    code to be executed if n=label3;
    break;
    ...
    default:
    code to be executed if n is different from all labels;
    }

PHP Loops

- 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*

PHP While and do while loops

- The while loop executes a block of code as long as the specified condition is true.
  - while (*condition is true*) {
    code to be executed;
    }
- The do…while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.
  - do {
    code to be executed;
    } while (*condition is true*);

PHP for and foreach Loops

- The for loop is used when you know in advance how many times the script should run.
  - for (*init counter; test counter; increment counter*) {
    code to be executed;
    }
    - *init counter*: Initialize the loop counter value
    - *test counter*: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
    - *increment counter*: Increases the loop counter value
- The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.
  - foreach ($*array* as $*value*) {
    code to be executed;
    }

PHP functions

- PHP has more than 1000 built-in 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.

  - function *functionName*() {
    code to be executed;
    }

Cont..

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

    <?php
        function writeMsg() {
        echo "Hello world!";
        }

        writeMsg();
    ?>

</body>
</html>
```

PHP Function Arguments

- 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.

```
<?php
    function firstName($fname) {
        echo "$fname.<br>";
    }
    firstName(“Manjunath");
    firstName(“Hegde");
?>
```

Function with two arguments

```
<?php
    function myName($fname, $year)
    {
        echo "$fname. Born in $year <br>";
    }
    myName(“Manjunath", "1990");
    myName(“Manju", "1990");
?>
```

Default Argument Value

```
<?php
    function setHeight($minheight = 50)
    {
        echo "The height is : $minheight <br>";
    }

    setHeight(350);
    setHeight(); // will use the default value of 50
    setHeight(135);
    setHeight(80);
?>
```

Functions - Returning values

```
<?php
    function sum($x, $y)
    {
        $z = $x + $y;
        return $z;
    }
    echo "5 + 10 = " . sum(5, 10) . "<br>";
    echo "7 + 13 = " . sum(7, 13) . "<br>";
    echo "2 + 4 = " . sum(2, 4);
?>
```

Arrays

- An array is a special variable, which can hold more than one value at a time.

```
<?php
    $myName = array(“fname", “mname", “lname");
    echo “My Name is" . $ fname[0] . ", " . $ mname[1] . " and " . $ lname[2] . ".";
?>
```

Create an Array in PHP

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

Array Type	Syntax	Example
Indexed arrays	$cars = array("Volvo", "BMW", "Toyota");	$cars = array("Volvo", "BMW", "Toyota"); the index can be assigned manually: Eg: $cars[0] = "Volvo"; $cars[1] = "BMW"; $cars[2] = "Toyota";
Associative arrays	$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");	$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43"); or: $age['Peter'] = "35"; $age['Ben'] = "37"; $age['Joe'] = "43";
Multidimensional arrays	$cars = array (array("Volvo",22,18), array("BMW",15,13), array("Saab",5,2), array("Land Rover",17,15));	echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].; echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].;

Sorting Arrays

- The elements in an array can be sorted in alphabetical or numerical order, descending or ascending.
  - *sort() - sort arrays in ascending order*
  - *rsort() - sort arrays in descending order*
  - *asort() - sort associative arrays in ascending order, according to the value*
  - *ksort() - sort associative arrays in ascending order, according to the key*
  - *arsort() - sort associative arrays in descending order, according to the value*
  - *krsort() - sort associative arrays in descending order, according to the key*

Objects

- An object is a data type which stores data and information on how to process that data.
- In PHP, an object must be explicitly declared.
- First we must declare a class of object. For this, we use the class keyword. A class is a structure that can contain properties and methods:

Eg:

```
<?php
class Car
{
    function Car() {
        $this->model = "VW";
    }
}
$herbie = new Car();

// show object properties
echo $herbie->model;
?>
```

Note:
$this is mainly used to refer properties of a class.

Cont..

```
class Student {
    // constructor
    public function __construct($first_name, $last_name) {
        $this->first_name = $first_name;
        $this->last_name = $last_name;
    }
    public function say_name() {
        echo "My name is " . $this->first_name . " " . $this->last_name . ".\n";
    }
}
$alex = new Student("Alex", "Jones");
$alex->say_name();
```

Cont..

- Here are some important definitions related to objects:
  - ***Classes*** *define how objects behave. Classes do not contain data.*
  - ***Objects*** *are instances of classes, which contain data.*
  - ***Members*** *are variables that belong to an object.*
  - ***Methods*** *are functions that belong to an object, and have access to its members.*
  - ***Constructor*** *is a special method that is executed when an object is created.*

Inheritance

- The most important feature of object oriented programming is inheritance. This feature allows us to reuse code we've written and extend it. For example, let's say we want to be able to define a math student, which also knows how to sum two numbers.

```
class Parent {
  // The parent’s class code
}

class Child extends Parent {
  // The  child can use the parent's class code
}
```

Form Handling

- The PHP superglobals $_GET and $_POST are used to collect form-data.

GET vs. POST

- Both GET and POST create an array (e.g. array(key => value, key2 => value2, key3 => value3, ...)). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.
- Both GET and POST are treated as $_GET and $_POST. These 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.
- $_GET is an array of variables passed to the current script via the URL parameters.
- $_POST is an array of variables passed to the current script via the HTTP POST method.

When to use GET?

- Information sent from a form with the GET method is **visible to everyone** (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send. The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.
- GET may be used for sending non-sensitive data.
- **Note:** GET should NEVER be used for sending passwords or other sensitive information!

When to use POST?

- Information sent from a form with the POST method is **invisible to others** (all names/values are embedded within the body of the HTTP request) and has **no limits** on the amount of information to send.
- Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server.
- However, because the variables are not displayed in the URL, it is not possible to bookmark the page.