**Introduction**

The PHP Hypertext Preprocessor (PHP) is a programming language that allows web developers to create dynamic content that interacts with databases. It’s a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites. It integrates with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.

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

**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

**Installation**

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

A simple way is to use WAMP/XAMPP and make your own local SERVER.

You can follow the [XAMPP Server Setup Here.](https://www.apachefriends.org/index.html)

### Syntax

A PHP script is executed on the server, and the plain HTML result is sent back to the browser.  It can be placed anywhere in the document. It 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:

<!DOCTYPE html>

<html>

<body>

<h1>My first PHP page</h1>

<?php

       echo "Hello World!";

?>

</body>

</html>

**Note:** PHP statements end with a semicolon (;).

**Variables**

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

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

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

**PHP has a total of eight data types which we use to construct our variables:**

* **Integers** − are whole numbers, without a decimal point, like 4195.
* **Doubles** − are floating-point numbers, like 3.14159 or 49.1.
* **Booleans** − have only two possible values either true or false.
* **NULL** − is a special type that only has one value: NULL.
* **Strings** − are sequences of characters, like ‘PHP supports string operations.’
* **Arrays** − are named and indexed collections of other values.
* **Objects** − are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
* **Resources** − are special variables that hold references to resources external to PHP (such as database connections).

The first five are *simple types*, and the next two (arrays and objects) are compound – the compound types can package up other arbitrary values of arbitrary type, whereas the simple types cannot.

<?php

//integer variable

$int\_var = 12345;

//double variable

$many = 2.2888800;

//Boolean

$x = true;

$y = false;

//string

$string\_1 = "This is a string in double quotes";

//array

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

//object

class Car {

function Car() {

$this->model = "VW";

}

}

// create an object

$herbie = new Car();

// show object properties

echo $herbie->model;

?>

### Comments

A *comment* is the portion of a program that exists only for the human reader and stripped out before displaying the programs result. There are two commenting formats in PHP:

**Single-line comments** − They are generally used for short explanations or notes relevant to the local code. Here are the examples of single line comments.

<?php

# This is a comment, and

# This is the second line of the comment

// This is a comment too. Each style comments only

print "An example with single line comments";

?>

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

<?php

/\* This is a comment with multiline

Subject: PHP

\*/

print "An example with multi line comments";

?>

### Output

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.

Here are the examples for the output statements:

<?php

$txt1 = "Learn PHP";

echo "Hello world!<br>";

echo "<h2>" . $txt1 . "</h2>";

print "<h2>PHP is Fun!</h2>";

print "<h2>" . $txt1 . "</h2>";

?>

### 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
* Logical operators
* Conditional (or ternary) Operators

##### PHP Arithmetic Operators

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

|  |  |  |
| --- | --- | --- |
| **Operator** | **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 |

##### PHP Assignment Operators

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

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

|  |  |  |
| --- | --- | --- |
| **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 |

##### PHP Comparison Operators

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

|  |  |  |
| --- | --- | --- |
| **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 |

##### PHP Logical Operators

The PHP logical operators are used to combine conditional statements.

|  |  |  |
| --- | --- | --- |
| **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. |

##### 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.

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

### Conditions

Conditional statements are used to perform different actions based on different conditions.

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

##### PHP – The if Statement

The if statement executes some code if one condition is true.

**Syntax**

if (condition) {

    code to be executed if condition is true;

}

**Example:**

<?php

$int = 2;

if ($int < 20) {

echo "This integer is less than 20";

}

?>

##### PHP – The if…else Statement

The if….else statement executes some code if a condition is true and another code if that condition is false.

**Syntax**

if (condition) {

code to be executed if condition is true;

} else {

code to be executed if condition is false;

}

**Example:**

<?php

$int = 2;

if ($int < 20) {

echo "This integer is less than 20";

} else {

echo "This integer is greater than 20";

}

?>

##### PHP – The if…elseif….else Statement

The if….elseif…else statement executes different codes for more than two conditions.

**Syntax**

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;

}

**Example:**

<?php

$int = 2;

if ($int > 0 && $int <= 5 ) {

echo "This integer is less than 5";

}

else if ($int > 5 && $int < 20 ) {

echo "This integer is less than 20";

}

else {

echo "This integer is greater than 20";

}

?>

##### PHP – The switch Statement

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

**Syntax**

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;

}

**Example:**

$favcolor = "red";

switch ($favcolor) {

case "red":

echo "Your favorite color is red!";

break;

case "blue":

echo "Your favorite color is blue!";

break;

case "green":

echo "Your favorite color is green!";

break;

default:

echo "Your favorite color is neither red, blue, nor green!";}

### Loop

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.

##### For 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 following example makes five iterations and changes the assigned value of two variables on each pass of the loop −

<?php

$a = 0;

$b = 0;

for( $i = 0; $i<5; $i++ ) {

   $a += 10;

   $b += 5;

}

echo ("At the end of the loop a = $a and b = $b" );

?>

##### The while loop statement

The while statement will execute a block of code if and as long as a test expression is true.

**Syntax**

while (condition) {

code to be executed;

}

**Example**  
This example decrements a variable value on each iteration of the loop and the counter increments until it reaches 10 when the evaluation is false and the loop ends.

<?php

$i = 0;

$num = 50;

while( $i < 10) {

$num--;

$i++;

}

echo ("Loop stopped at i = $i and num = $num" );

?>

The 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**

The following example will increment the value of i at least once, and it will continue incrementing the variable i as long as it has a value of less than 10.

<?php

$i = 0;

$num = 0;

do {

$i++;

}

while( $i < 10 );

echo ("Loop stopped at i = $i" );

?>

##### The 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**

<?php

$array = array( 1, 2, 3, 4, 5);

foreach( $array as $value ) {

echo "Value is $value <br />";

}

?>

##### The break statement

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

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

<?php

$i = 0;

while( $i < 10) {

$i++;

if( $i == 3 )break;

}

echo ("Loop stopped at i = $i" );

?>

This will produce the following result:  
Loop stopped at i = 3

##### The continue statement

The PHP continue keyword is used to halt the current iteration of a loop but it does not terminate the loop.

**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

$array = array( 1, 2, 3, 4, 5);

foreach( $array as $value ) {

if( $value == 3 )continue;

echo "Value is $value <br />";

}

?>

**This will produce the following result:**

Value is 1  
Value is 2  
Value is 4  
Value is 5

### Function

PHP functions are similar to other programming languages. A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value.

##### Creating PHP Function

Its very easy to create your own PHP function. Suppose you want to create a PHP function which will simply write a simple message on your browser when you will call it. Following example creates a function called writeMessage() and then calls it just after creating it.

<?php

/\* Defining a PHP Function \*/

**function** writeMessage() {

echo "You are really a nice person, Have a nice time!";

}

/\* Calling a PHP Function \*/

writeMessage();

?>

##### PHP Functions with Parameters

PHP gives you option to pass your parameters inside a function. You can pass as many as parameters your like. These parameters work like variables inside your function. Following example takes two integer parameters and add them together and then print them.

<?php

**function** addFunction($num1, $num2) {

$sum = $num1 + $num2;

echo "Sum of the two numbers is : $sum";

}

addFunction(10, 20);

?>



